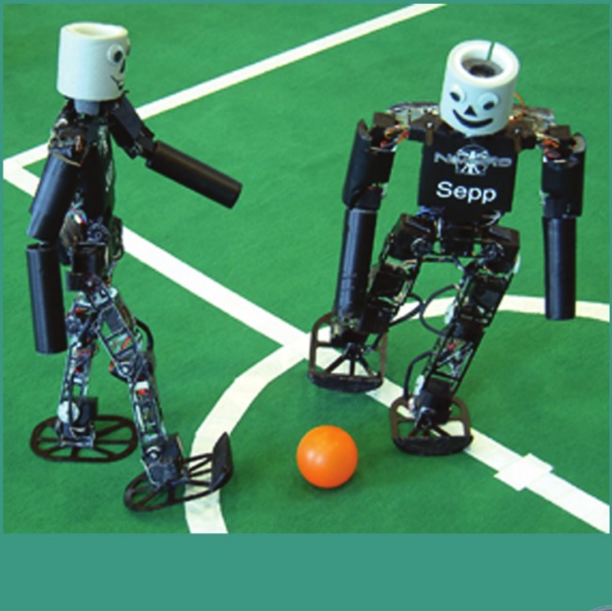
Ansgar Bredenfeld
Adam Jacoff
Itsuki Noda
Yasutake Takahashi (Eds.)

LNAI 4020

RoboCup 2005: Robot Soccer World Cup IX



Springer

RoboCup

Lecture Notes in Artificial Intelligence 4020

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Ansgar Bredenfeld   Adam Jacoff
Itsuki Noda   Yasutake Takahashi (Eds.)

RoboCup 2005: Robot Soccer World Cup IX

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Ansgar Bredenfeld
Fraunhofer Institute for Autonomous Intelligent Systems (AIS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
E-mail: ansgar.bredenfeld@ais.fraunhofer.de

Adam Jacoff
National Institute of Standards and Technology, Intelligent Systems Division
100 Bureau Drive, Mail Stop 8230, Gaithersburg, MD 20899-8230, USA
E-mail: adam.jacoff@nist.gov

Itsuki Noda
National Institute of Advanced Industrial Science and Technology
Information Technology Research Institute
2-41-6 Aomi, Koto-ku, Tokyo 135-0064, Japan
E-mail: i.noda@aist.go.jp

Yasutake Takahashi
Osaka University, Graduate School of Engineering
Department of Adaptive Machine Systems
Yamadaoka 2-1, Suita, Osaka 565-0871, Japan
E-mail: yasutake@er.ams.eng.osaka-u.ac.jp

Library of Congress Control Number: 2006927344

CR Subject Classification (1998): I.2, C.2.4, D.2.7, H.5, I.5.4, J.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-35437-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-35437-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11780519 06/3142 5 4 3 2 1 0

Preface

The 9th RoboCup International Symposium was held at INTEX Osaka, Japan, immediately after the RoboCup 2005 Soccer, Rescue and Junior Competitions. The researchers behind the RoboCup teams met for presentation of scientific and technological contributions in areas relevant to RoboCup.

The number of submissions to the symposium increased again and totalled 131. The International Program Committee including both RoboCup researchers from around the world and experts from communities outside RoboCup selected 36 submissions as full papers and 38 as posters. These contributions came from 21 countries and were presented in 6 oral sessions and 2 poster sessions.

Two of the full papers were selected as awards. The Scientific Award went to the paper authored by Jelle Kok and Nikos Vlassis on “Using the Max-Plus Algorithm for Multiagent Decision Making in Coordination Graphs.” The Engineering Award was given to the paper written by Gerald Steinbauer, Martin Mörth and Franz Wotawa with the title “Real-Time Diagnosis and Repair of Faults of Robot Control Software.”

The Fourth IEEE International Conference on Development and Learning (ICDL 2005) took place at the same venue as RoboCup 2005 during July 19-21, 2005. ICDL 2005 brought together researchers from the field of learning in humans, animals, and automatons to discuss and exchange ideas between robotics and cognitive science. As a second co-located academic event, the 5th International Workshop on Epigenetic Robotics (EpiRob 2005) was also held in Nara.

The RoboCup International Symposium started with an invited talk by Hiroshi Ishiguro of Osaka University and ATR under the title “Research Issues for Personal and Social Robots.” We had another two invited talks in joint sessions with ICDL 2005, delivered by Pat Langley of Stanford University, on “An Adaptive Architecture for Physical Agents,” and by Giogio Metta, University of Genova, on “Developmental Robotics and Robot Development.”

We like to thank the RoboCup Organizing Committee for the local organization of the Symposium at INTEX Osaka. The RoboCup competitions and the scientific symposium celebrated their 10th anniversary in Bremen, Germany, on June 14-20, 2006.

December 2005

Ansgar Bredenfeld
Adam Jacoff
Itsuki Noda
Yasutake Takahashi

Organization

The RoboCup Federation

The RoboCup Federation is an international organization, registered in Switzerland, aiming to promote science and technology using robot and simulation competition.

President
Minoru Asada (Osaka University, Japan)

Vice-Presidents
Manuela Veloso (Carnegie Mellon University, USA)
Hans-Dieter Burkhard (Humboldt University, Germany)

Founding President
Hiroaki Kitano (Kitano Symbiotic Systems Project, JST, Japan)

Board of Trustees
Tucker Balch (Georgia Institute of Technology, USA)
Silvia Coradeschi (Örebro University, Sweden)
Gerhard K. Kraetzschmar (University Bonn-Rhein-Sieg, Germany)
Pedro U. Lima (Instituto Superior Técnico, Portugal)
Daniele Nardi (University of Rome “La Sapienza”, Italy)
Itsuki Noda (AIST, Japan)
Enrico Pagello (University of Padova, Italy)
Elizabeth Sklar (Brooklyn College, City University of New York, USA)
Peter Stone (The University of Texas at Austin, USA)
Satoshi Tadokoro (Tohoku University, Japan)

Executive Committee

Simulation League
Daniel Polani (University of Hertfordshire, UK)
Martin Riedmiller (University of Osnabrück, Germany)
Ubbo Visser (University of Bremen, Germany)

Small-Size Robot (F-180) League
Tadashi Naruse (Aichi Prefectural University, Japan)
Ng BengKiat (Ngee Ann Polytechnic, Singapore)
Brett Browning (Carnegie Mellon University, USA)

Middle-Size Robot (F-2000) League
Yasutake Takahashi (Osaka University, Japan)
Andrea Bonarini (Politecnico di Milano, Italy)
Fernando Ribeiro (University of Minho, Portugal)

4-Legged Robot League
Claude Sammut (University of New South Wales, Australia)
Thomas Röfer (University of Bremen, Germany)

Humanoid League
Changjiu Zhou (Singapore Polytechnic, Singapore)
Norbert Mayer (Osaka University, Japan)

RoboCupRescue Robot League
Adam Jacoff (National Institute of Standards and Technology, USA)
Andreas Birk (International University of Bremen, Germany)

RoboCupRescue Simulation League
Tomoichi Takahashi (Meijo University, Japan)
H. Levent Akın (Boğaziçi University, Turkey)

RoboCupJunior
Jeffrey Johnson (Open University, UK)
Luis Almeida (University of Aveiro, Portugal)
Brian Thomas (Australia)

Web Presentation
Ansgar Bredenfeld (Fraunhofer AIS, Germany)

RoboCup 2005 Organization

Chairs
Hitoshi Matsubara (Future University - Hakodate, Japan), *General Chair*
Satoshi Tadokoro (Tohoku University, Japan), *Organizing Chair*
Itsuki Noda (AIST, Japan), *Vice Organizing Chair*
Takeshi Ohashi (Kyushu Inst. of Tech., Japan), *Vice Organizing Chair*

Symposium
Ansgar Bredenfeld (Fraunhofer AIS, Germany)
Adam Jacoff (NIST, USA)
Itsuki Noda (AIST, Japan)
Yasutake Takahashi (Osaka University, Japan)

RoboCup Soccer - Humanoid League
Norbert Mayer (Osaka University, Japan), *Chair and Local Chair*
Changjiu Zhou (Singapore Polytechnic, Singapore)

Jacky Baltes (University of Manitoba, Canada)
Sven Behnke (University of Freiburg, Germany)

RoboCup Soccer - Middle-Size Robot League (MSL)

Dominico Giorgio Sorrenti (MRT, Italy), *Chair*
Paul Gerhard Plöger (Fraunhofer AIS, Germany)
Alireza Fadaei Tehrani, (Isfahan University of Technology, Iran)
Hikari Fujii (Keio University, Japan)
Yasutake Takahashi (Osaka University, Japan), *Local Chair*

RoboCup Soccer - Small-Size Robot League (SSL)

Tadashi Naruse (Aichi Prefectural University, Japan), *Chair*
Tim Laue (Bremen University, Germany)
David Ball (Queensland University, Australia)
Yuki Nakagawa (iXs Research Corp., Japan), *Local Chair*

RoboCup Soccer - 4-Legged Robot League

Vincent Hugel (LRV, France), *Chair*
Thomas Röfer (University of Bremen, Germany)
Noriaki Mitsunaga (ATR, Japan), *Local Chair*

RoboCup Soccer - Simulation League

Jelle Kok (University of Amsterdam, The Netherlands), *Chair*
Joschka Boedecker (University of Koblenz, Germany)
Mohammad Nejad Sedaghat (Iran Univ. of Science and Technology, Iran)
Junji Nishino (University of Electro-Communications, Japan)
Tomoharu Nakashima (Osaka Prefecture University, Japan), *Local Chair*
Hidehisa Akiyama (Tokyo Institute of Technology, Japan), *Local Chair*

RoboCupRescue - Rescue Robot League

Adam Jacoff (NIST, USA), *Chair*
Andreas Birk (International University of Bremen, Germany)
Satoshi Tadokoro (Kobe University, Japan)
Tetsuya Kimura (Nagaoka University of Technology, Japan), *Local Chair*

RoboCupRescue - Rescue Simulation League

Alexander Kleiner (University of Freiburg, Germany), *Chair*
Tomoichi Takahashi (Meijo University, Japan), *Local Chair*

RoboCup Junior

Luis Almeida (University of Aveiro, Portugal), *Chair*
Martin Bader (IGV, Germany)
Amy Eguchi (University of Cambridge, UK)
Tairo Nomura (Saitama University, Japan), *Local Chair*
Carlos Cardeira (IST, Portugal)
Tadahiro Kaneda (Osaka Prefectural College of Technology, Japan)

International Symposium Program Committee

Tamio Arai, Japan
Ronald Arkin, USA
Minoru Asada, Japan
Tucker Balch, USA
Jacky Baltes, Canada
Wolfgang Banzhaf, Canada
Andreas Birk, Germany
Andrea Bonarini, Italy
Ansgar Bredenfeld, Germany
Hans-Dieter Burkhard, Germany
Lola Cañamero, UK
Thomas Christaller, Germany
Brad Clement, USA
Silvia Coradeschi, Sweden
Kosei Demura, Japan
Jorge Dias, Portugal
Claudia Goldman, Israel
Andrew Howard, USA
Huosheng Hu, UK
Hiroshi Ishiguro, Japan
Nobuhiro Ito, Japan
Adam Jacoff, USA
Mansour Jamzad, Iran
Hyuckchul Jung, USA
Nate Kohl, USA
Gerhard Kraetzschmar, Germany
Kostas Kyriakopoulos, Greece
Pedro Lima, Portugal
Yoichiro Maeda, Japan
Hitoshi Matsubara, Japan
Akihiro Matsumoto, Japan
Noriaki Mitsunaga, Japan
Tomoharu Nakashima, Japan
Daniele Nardi, Italy
Tadashi Naruse, Japan
Katsumi Nitta, Japan
Itsuki Noda, Japan
Takeshi Ohashi, Japan
Masayuki Ohta, Japan
Enrico Pagello, Italy
Paul Plöger, Germany
Daniel Polani, UK
David Pynadath, USA
Martin Riedmiller, Germany
Thomas Röfer, Germany
Raúl Rojas, Germany
Paul Rybski, USA
Claude Sammut, Australia
Paul Scerri, USA
Sandip Sen, USA
Onn Shehory, Israel
Kosuke Shinoda, Japan
Elizabeth Sklar, USA
Bill Smart, USA
Peter Stone, USA
Katya Sycara, USA
Tomoichi Takahashi, Japan
Yasutake Takahashi, Japan
Ikuo Takeuchi, Japan
Jean-Philippe Tarel, France
Ashley Tews, Australia
Takashi Tubouchi, Japan
Ubbo Visser, Germany
Felix von Hundelshausen, USA
Marco Wiering, The Netherlands
Laura R. Winer, Canada
Florentin Wörgötter, Germany
Gordon Wyeth, Australia
Changjiu Zhou, Singapore

Table of Contents

Award Winning Papers

Scientific Award

Engineering Award

Full Papers

Humanoid

Learning, Analysis, Sensor-Control

Localization, Device, Tools, Other

Multi Agents, Robots

Rescue

Vision

Posters

Using the Max-Plus Algorithm for Multiagent Decision Making in Coordination Graphs

Jelle R. Kok and Nikos Vlassis

Informatics Institute, University of Amsterdam, The Netherlands
{jellekok, vlassis}@science.uva.nl

Abstract. Coordination graphs offer a tractable framework for cooperative multiagent decision making by decomposing the global payoff function into a sum of local terms. Each agent can in principle select an optimal individual action based on a variable elimination algorithm performed on this graph. This results in optimal behavior for the group, but its worst-case time complexity is exponential in the number of agents, and it can be slow in densely connected graphs. Moreover, variable elimination is not appropriate for real-time systems as it requires that the complete algorithm terminates before a solution can be reported. In this paper, we investigate the max-plus algorithm, an instance of the belief propagation algorithm in Bayesian networks, as an approximate alternative to variable elimination. In this method the agents exchange appropriate payoff messages over the coordination graph, and based on these messages compute their individual actions. We provide empirical evidence that this method converges to the optimal solution for tree-structured graphs (as shown by theory), and that it finds near optimal solutions in graphs with cycles, while being much faster than variable elimination.

1 Introduction

A multiagent system (MAS) consists of a group of agents that interact which each other [1, 2]. In such systems agents act individually, but the outcome can differ based on the behavior of the other agents. In this paper, we concentrate on cooperative MASs in which the agents try to optimize a shared performance measure and have to ensure that their selected individual actions result in good team behavior. RoboCup [3] is a good example of a cooperative (or team) MAS in which the agents also have to deal with time constraints since the soccer-playing robots have to coordinate their actions in real-time in order to win.

A recently introduced framework for multiagent coordination is the concept of coordination graphs (CG) [4], which allows for a tractable representation of the coordination problem. In this framework, payoff functions between subsets of the agents are specified that represent local coordination dependencies between the agents. In order to determine the optimal joint action that maximizes the sum of the local payoffs, a variable elimination (VE) algorithm was proposed in [4]. We applied CGs and VE to our UvA Trilearn RoboCup Simulation team,

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 1–12, 2006.

which won the RoboCup-2003 World Championship [5]. However, although VE is exact and always produces the optimal joint action, it can be slow in certain cases and in the worst case scales exponentially in the number of agents for densely connected graphs. In previous work [6], we compared two different alternatives to VE. In this paper we will concentrate further on the max-plus algorithm, which is analogous to the belief propagation algorithm [7, 8, 9] for Bayesian networks, as an approximate alternative to VE. In this method, the agents repeatedly exchange payoff messages on which they base their individual action selection. In this paper, we provide empirical evidence that this method converges to the optimal solution for tree-structured graphs and also show that it finds near optimal solutions in densely connected graphs with cycles, while being much faster than VE. These results make this framework interesting for domains as RoboCup where real-time decision making in a group of distributed cooperative agents is of great importance.

2 Coordination Graphs and Variable Elimination

In this section we will review the problem of computing a coordinated action for a group of n agents using the *variable elimination* (VE) algorithm [4]. Each agent chooses an individual action a_i from a set A_i, and the resulting *joint* action $a = (a_1, \ldots, a_n)$ generates a payoff $u(a)$ for the team. The coordination problem is to find the optimal joint action a^* that maximizes $u(a)$, i.e., $a^* = \arg\max_a u(a)$. An obvious approach is to enumerate all possible joint actions and select the one that maximizes $u(a)$. However, this approach quickly becomes impractical, since the joint action space $\times_i A_i$ grows exponentially with the number of agents n.

Fortunately, many problems exhibit the property that the payoff matrix $u(a)$ is sparse. Each agent only depends on a small subset of the other agents, e.g., in robotic soccer only robots that are close to each other have to coordinate their actions. A recent approach to exploit such dependencies involves the use of a coordination graph [4], which decomposes the global payoff function $u(a)$ into a linear combination of local payoff functions, each involving only a few agents. For example, a payoff function involving four agents can be decomposed as follows:

$$u(a) = f_{12}(a_1, a_2) + f_{13}(a_1, a_3) + f_{34}(a_3, a_4). \tag{1}$$

The functions f_{ij} specify the local coordination dependencies between the actions of the agents and can be mapped to a graph $G = (V, E)$ in which each node in V represents an agent, while an edge in E defines a coordination dependency between two agents. Only interconnected agents have to coordinate their actions at any particular instance. The global coordination problem is thus replaced by a number of local coordination problems each involving fewer agents.

In order to find the optimal joint action a^* we can apply VE, which is essentially identical to variable elimination in a Bayesian network [10]. The algorithm operates as follows. One agent is selected and collects all payoff functions in which it is involved from its neighbors. Next, it optimizes its decision conditionally on the possible action combinations of its neighbors and communicates the resulting ‘conditional’ payoff function to one of its neighbors. Thereafter, this agent is eliminated from the graph. When only one agent remains, this agent selects an action that maximizes the final conditional strategy. A second pass in the reverse order is then performed in which every agent computes its optimal action based on its conditional strategy and the fixed actions of its neighbors.

We will illustrate VE on the aforementioned decomposition (1). We first eliminate agent 1. This agent depends on the local payoff functions f_{12} and f_{13} and therefore the maximization of $u(a)$ in (1) can be written as

$$\max_a u(a) = \max_{a_2,a_3,a_4} \Big\{ f_{34}(a_3,a_4) + \max_{a_1}[f_{12}(a_1,a_2) + f_{13}(a_1,a_3)] \Big\}. \qquad (2)$$

Agent 1 defines the function $\phi_{23}(a_2,a_3) = \max_{a_1}[f_{12}(a_1,a_2) + f_{13}(a_1,a_3)]$ and the best-response (conditional strategy) function $B_1(a_2,a_3) = \arg\max_{a_1} [f_{12}(a_1,a_2) + f_{13}(a_1,a_3)]$ which respectively return the maximal value and the associated best action for agent 1 given the actions of agent 2 and 3. The function $\phi_{23}(a_2,a_3)$ is independent of agent 1, which can now be eliminated from the graph, simplifying (2) to $\max_a u(a) = \max_{a_2,a_3,a_4}[f_{34}(a_3,a_4) + \phi_{23}(a_2,a_3)]$. Note that the elimination of agent 1 induces a new dependency between agent 2 and 3 and thus a change in the graph's topology.

Next, we apply the same procedure to eliminate agent 2. Since only ϕ_{23} depends on agent 2, we define $B_2(a_3) = \arg\max_{a_2} \phi_{23}(a_2,a_3)$ and replace ϕ_{23} by $\phi_3(a_3) = \max_{a_2} \phi_{23}(a_2,a_3)$ producing $\max_a u(a) = \max_{a_3,a_4}[f_{34}(a_3,a_4) + \phi_3(a_3)]$, which is independent of a_2. Next, we eliminate agent 3 giving $\max_a u(a) = \max_{a_4} \phi_4(a_4)$ with $\phi_4(a_4) = \max_{a_3}[f_{34}(a_3,a_4) + \phi_3(a_3)]$. Agent 4 is the last remaining agent and fixes its optimal action $a_4^* = \arg\max_{a_4} \phi_4(a_4)$. Thereafter, a second pass in the reverse elimination order is performed in which each agent computes its optimal (unconditional) action from its best-response function and the fixed actions from its neighbors. In our example, agent 3 first selects $a_3^* = B_3(a_4^*)$. Similarly, we get $a_2^* = B_2(a_3^*)$ and $a_1^* = B_1(a_2^*, a_3^*)$. In the case that one agent has more than one maximizing best-response action, it can select one randomly, since it always communicates its choice to its neighbors.

The outcome of VE is independent of the elimination order and always results in the optimal joint action. On the other hand, the execution time of the algorithm does depend on the elimination order[1] and is exponential in the induced width of the graph (the size of the largest clique computed during node elimination). This can be slow in certain cases and in the worst case scales exponentially in n. Furthermore, VE will only produce its final result after the end of the second pass. This is not always appropriate for real-time multiagent systems where decision making must be done under time constraints. For example, in the RoboCup 2D Simulation League, each agent has to sent an action to the server within 100ms. In these cases, an anytime algorithm that improves the quality of the solution over time would be more appropriate.

[1] Computing the optimal order for minimizing the runtime costs is known to be NP-complete, but good heuristics exists, e.g., minimum deficiency search which first eliminates the agent with the minimum number of neighbors [11].

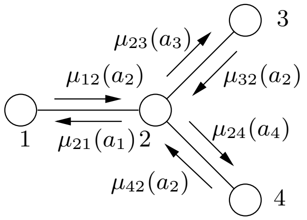


Fig. 1. Graphical representation of different messages μ_{ij} in a graph with four agents

3 The Max-Plus Algorithm

In this section, we describe the *max-plus algorithm* [7, 8, 9, 6] as an approximate alternative to VE. The max-plus algorithm is a popular method for computing the *maximum a posteriori* (MAP) configuration in an (unnormalized) undirected graphical model. This method, analogous to the belief propagation (BP) or sum-product algorithm in Bayesian networks, operates by iteratively sending messages $\mu_{ij}(a_j)$, which can be regarded as locally optimized payoff functions, between agent i and j over the edges of the graph. For tree-structured graphs, the message updates converge to a fixed point after a finite number of iterations [7].

We can translate the above result to our multiagent decision making problem, since computing the MAP configuration is equivalent to finding the optimal joint action in a CG [6]. Suppose that we have a coordination graph $G = (V, E)$ with $|V|$ vertexes and $|E|$ edges. Instead of variables, the nodes in V represent agents, while the global payoff function can be decomposed as follows[2]

$$u(a) = \sum_{i \in V} f_i(a_i) + \sum_{(i,j) \in E} f_{ij}(a_i, a_j). \tag{3}$$

Here f_i denotes a local payoff function for agent i and is only based on its individual action a_i. Furthermore, $(i, j) \in E$ denotes a pair of neighboring agents (an edge in G), and f_{ij} is a local payoff function that maps two actions (a_i, a_j) to a real number $f_{ij}(a_i, a_j)$. A function f_i thus represents the payoff an agent contributes to the system when it acts individually, e.g., dribbling with the ball, and f_{ij} represents the payoff of a coordinated action, e.g., a coordinated pass.

Again, the goal is to find the optimal joint action a^* that maximizes (3). Each agent i (node in G) repeatedly sends a message μ_{ij} to its neighbors $j \in \Gamma(i)$, where μ_{ij} maps an action a_j of agent j to a real number as follows:

$$\mu_{ij}(a_j) = \max_{a_i} \big\{ f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \big\} + c_{ij}, \tag{4}$$

where $\Gamma(i) \setminus j$ represents all neighbors of i except j and c_{ij} is a normalization vector. This message can be understood as an approximation of the maximum

[2] Note that this function $u(a)$ is analogous to the log-transform of an (unnormalized) factorized probability distribution for which the MAP state is sought.

Algorithm 1. Pseudo-code of the centralized max-plus algorithm

```
centralized max-plus algorithm for $CG = (V, E)$
intialize $\mu_{ji} = \mu_{ij} = 0$ for $(i, j) \in E$, $g_i = 0$ for $i \in V$ and $m = -\infty$
while fixed-point = false and deadline to sent action has not yet arrived do
  // run one iteration
  fixed-point = true
  for every agent $i$ do
    for all neighbors $j = \Gamma(i)$ do
      send $j$ message $\mu_{ij}(a_j) = \max_{a_i} \left\{ f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \right\} + c_{ij}$
      if $\mu_{ij}(a_j)$ differs from previous message by a small threshold then
        fixed-point = false
    determine $g_i(a_i) = f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)$ and $a'_i = \arg\max_{a_i} g_i(a_i)$
  if use anytime extension then
    if $u((a'_i)) > m$ then
      $(a^*_i) = (a'_i)$ and $m = u((a'_i))$
  else
    $(a^*_i) = (a'_i)$
return $(a^*_i)$
```

payoff i can achieve for a given action of j, and is computed by maximizing (over the actions of i) the sum of the payoff functions f_i and f_{ij} and all incoming messages to i except that from j. The agents keep exchanging messages until they converge. Fig. 1 shows a CG with four agents and the corresponding messages.

A message μ_{ij} in max-plus has three important differences with respect to the conditional strategies in VE. First, before convergence each message is an approximation of the exact value (conditional payoff) since it depends on the incoming (still unconverged) messages. Second, an agent i only has to sum over the received messages from its neighbors defined over its individual actions, instead of enumerating over all possible action combinations of its neighbors (this makes the algorithm tractable). Finally, in VE the elimination of an agent often causes a change in the graph topology. In the max-plus algorithm, messages are always sent over the edges of the original graph.

For trees the messages converge to a fixed-point within a finite number of steps [7, 9]. A message $\mu_{ij}(a_j)$ then equals the payoff the subtree with agent i as root can produce when agent j performs action a_j. If, at convergence, we define

$$g_i(a_i) = f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i), \tag{5}$$

then we can show that $g_i(a_i) = \max_{\{a' | a'_i = a_i\}} u(a')$ holds [6]. Each agent i now individually selects its optimal action $a^*_i = \arg\max_{a_i} g_i(a_i)$. If there is only one maximizing action for every agent i, the globally optimal joint action $a^* = \arg\max_a u(a)$ is unique and has elements $a^* = (a^*_i)$. Note that this global optimal joint action is computed by only local optimizations (each node maximizes $g_i(a_i)$ separately). In case the local maximizers are not unique, an optimal joint action can be computed by a dynamic programming technique [9, sec. 3.1].

Algorithm 2. Pseudo-code of a distributed max-plus implementation

```
distributed max-plus for agent $i$, $CG = (V, E)$, spanning tree $ST = (V, S)$
initialize $\mu_{ji} = \mu_{ij} = 0$ for $j \in \Gamma(i)$, $g_i = 0$, $p_i = 0$ and $m = -\infty$
while deadline to sent action has not yet arrived do
  wait for message $msg$
  if $msg = \mu_{ji}(a_i)$ // max-plus message then
    for all neighbors $j = \Gamma(i)$ do
      compute $\mu_{ij}(a_j) = \max_{a_i} \left\{ f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in \Gamma(i) \setminus j} \mu_{ki}(a_i) \right\} + c_{ij}$
      send message $\mu_{ij}(a_j)$ to agent $j$ (if it differs from last sent message)
    if use anytime extension then
      if heuristic indicates global payoff should be evaluated then
        send evaluate( $i$ ) to agent $i$ (me) // initiate computation global payoff
    else
      $a_i^* = \arg\max_{a_i} [f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]$
  if $msg =$ evaluate( $j$ ) // receive request for evaluation from agent $j$ then
    lock $a_i' = \arg\max_{a_i} [f_i(a_i) + \sum_{j \in \Gamma(i)} \mu_{ji}(a_i)]$ and set $p_i = 0$ if $a_i'$ not locked
    send evaluate( $i$ ) to all neighbors (parent and children) in $ST \neq j$
    if $i =$ leaf in $ST$ then
      send accumulate_payoff( 0 ) to agent $i$ (me) // initiate accumulation payoffs
  if $msg =$ accumulate_payoff( $p_j$ ) from agent $j$ then
    $p_i = p_i + p_j$ // add payoff child $j$
    if received accumulated payoff from all children in $ST$ then
      get actions $a_j'$ from $j = \Gamma(i)$ in CG and set $g_i = f_i(a_i') + \frac{1}{2} \sum_{j \in \Gamma(i)} f_{ij}(a_i', a_j')$
      if $i =$ root of $ST$ then
        send global_payoff( $g_i + p_i$ ) to all children in $ST$
      else
        send accumulate_payoff( $g_i + p_i$ ) to parent in $ST$
  if $msg =$ global_payoff( $g$ ) then
    if $g > m$ then
      $a_i^* = a_i'$ and $m = g$
    send global_payoff( $g$ ) to all children in $ST$ and unlock action $a_i'$
return $a_i^*$
```

Unfortunately, in graphs with cycles there are no guarantees that either max-plus converges or that the local maximizers $a_i^* = \arg\max_{a_i} g_i(a_i)$ correspond to the global optimum. It has been shown that a fixed point of message passings exists [9], but there is no algorithm yet that can provably converge to such a solution. However, bounds are available that characterize the quality of the solution if the algorithm converges [12]. In practice, the max-product algorithm has been successfully applied in graphs with cycles [8, 13, 14].

The max-plus algorithm can both be implemented in a centralized and distributed version. In the distributed implementation, each agent computes and sends updated messages after it has received a new (and different) message from one of its neighbors. In this case, messages are sent in parallel, resulting in a computational advantage over the sequential execution of the centralized algorithm. However, an additional complexity arises since each agent has to individually determine whether the system has converged or when it should report its action. In general we can assume that each agent receives a ‘deadline’ signal (either from an external source or from an internal synchronized timing signal) after which it must report its action. This, in turn, necessitates the development of an anytime algorithm in which each (local) action is only updated when the corresponding global payoff improves upon the best one found so far. In the centralized version of Alg. 1, we therefore compute the global payoff after every iteration by inserting the current computed joint action into (3). For the distributed case in Alg. 2 the evaluation of the (distributed) joint action is much more complex and is only initiated by an agent when it believes it is worthwhile to do so, e.g., after a big increase in the values of the received messages. This agent starts the propagation of an 'evaluation' over a spanning tree ST of the nodes in G. This tree is fixed beforehand and common knowledge among all agents. An agent receiving an evaluation message fixes its individual action. When an agent is a leaf of ST it also computes its local contribution to the global payoff and sends it to its parent in ST. Each parent accumulates all payoffs of its children and after adding its own contribution sends the result to its parent. Finally, when the root of ST has received all accumulated payoffs from its children, the sum of these payoffs (global payoff) is distributed to all nodes in ST. An agent updates its best individual action a_i^* only when this payoff improves upon the best one found so far. When the 'deadline' signal arrives, each agent thus does not report the action corresponding to the current messages, but the action related to the highest found global payoff. Alg. 1 and Alg. 2 respectively show a centralized and a distributed version in pseudo-code.

4 Experiments

In this section, we describe our experiments with the max-plus algorithm on differently shaped graphs. Since our focus in this paper is on the resulting policies and the corresponding (global) payoff, we used the centralized algorithm from Alg. 1. We first tested it on trees with $|V| = 100$ agents, each having $|A_i| = 4$ actions and a fixed number of neighbors. We created 24 trees in which the number of neighbors per node ranged between $[2, 25]$. Since we fixed the number of agents, each tree had 99 edges but a different depth. Each edge $(i, j) \in E$ was associated with a payoff function f_{ij} where each action combination was assigned a payoff $f_{ij}(a_i, a_j)$ generated from a normal distribution $\mathcal{N}(0, 1)$.

We applied both the VE and max-plus algorithm to compute the joint action. In VE we always eliminated the agent with the minimum number of neighbors, such that each local maximization step involved at most two agents. For the max-plus algorithm, we applied both a random order and the same order as VE to select the agent to sent its messages. Note that in the latter case, the second iteration is the reverse order of the first (comparable to the reversed pass in VE).

Fig. 2 shows the relative payoff found with max-plus with respect to the optimal payoff after each iteration. Results are averaged over all 24 graphs. The policy converges to the optimal solution after a few iterations. When using the elimination order of VE to select the next agent, it always converges after two iterations. For this order, each message only has to be computed once

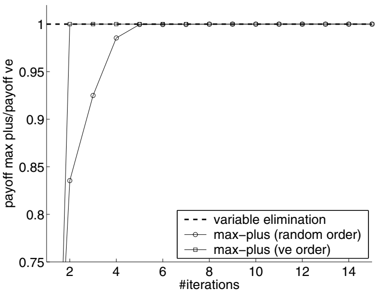


Fig. 2. Average payoff for the max-plus algorithm after each iteration

(see [15]) and the two methods become equivalent. When using a random order, it takes a few iterations before the same information is propagated through the graph.

We also tested max-plus on graphs with cycles. Now, because an outgoing message from agent i can eventually become part of its incoming messages, the values of the messages can become extremely large. Therefore, as in [9], we normalize each sent message by subtracting the average of all values in μ_{ik} using $c_{ij} = \frac{1}{|A_k|}\sum_k \mu_{ik}(a_k)$ in (4). Furthermore, we stopped max-plus after 100 iterations when the messages did not converge (as we will see later in Fig. 4 the policy has stabilized at this point).

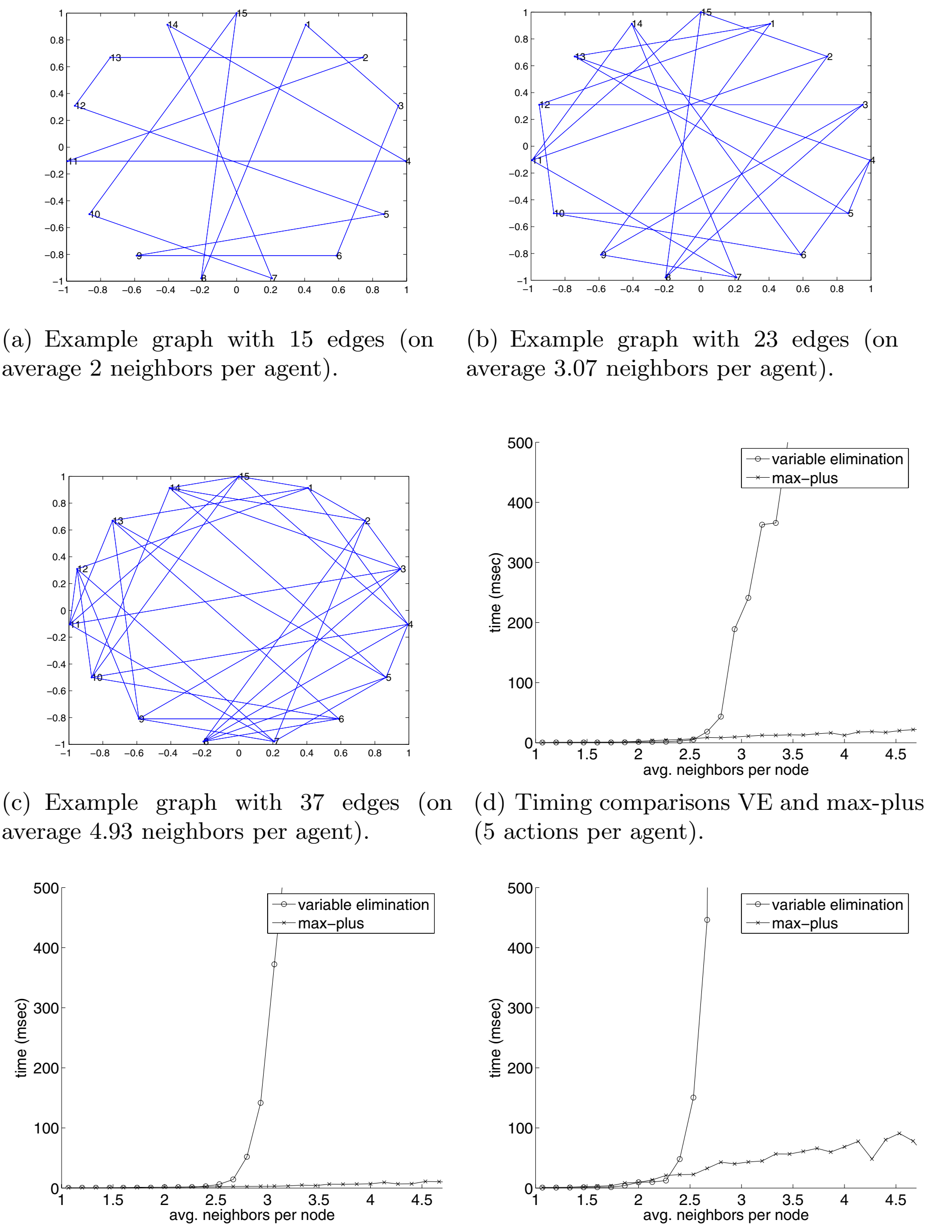
For our experiments, we created graphs with 15 agents, and a varying number of edges. In order to get a balanced graph in which each agent approximately had the same number of neighbors, we randomly added edges between the agents with the minimum number of edges. We generated 100 graphs for each $|E| \in [8, 37]$, resulting in 3000 graphs. Fig. 3(a)-3(c) depict example graphs with respectively 15, 23 and 37 edges (on average 2, 3.07 and 4.93 neighbors per node).

We applied the above procedure to create three test sets. In the first set, each edge $(i,j) \in E$ was associated with a payoff function f_{ij} defined over five actions per agent and each action combination was assigned a random payoff $f_{ij}(a_i, a_j) \in \mathcal{N}(0,1)$. In the second set, we added one outlier to each payoff function: for a randomly picked joint action, the corresponding payoff value was set to $10 * \mathcal{N}(0,1)$. For the third test set, we specified a payoff function using 10 actions per agent and the same method as in the first set to generate the values.

The timing results[3] for the three different test sets[4] are plotted in Fig. 3(d)-3(f). They show that the time for the max-plus algorithm grows linearly as the complexity of the graphs increases (the number of messages is related to the number of edges in the graph). The time for VE grows exponentially since it has to enumerate over an increasing number of neighboring agents in each local maximization step. Furthermore, the elimination of an agent often causes

[3] All results are generated on a 3.4GHz / 2GB machine using a C++ implementation.

[4] For the graphs with ten actions per agent and more than four neighbors per node, VE was not always able to compute a policy since the intermediate computed tables grew too large for the available memory. These graphs were removed from the set.



(a) Example graph with 15 edges (on average 2 neighbors per agent).

(b) Example graph with 23 edges (on average 3.07 neighbors per agent).

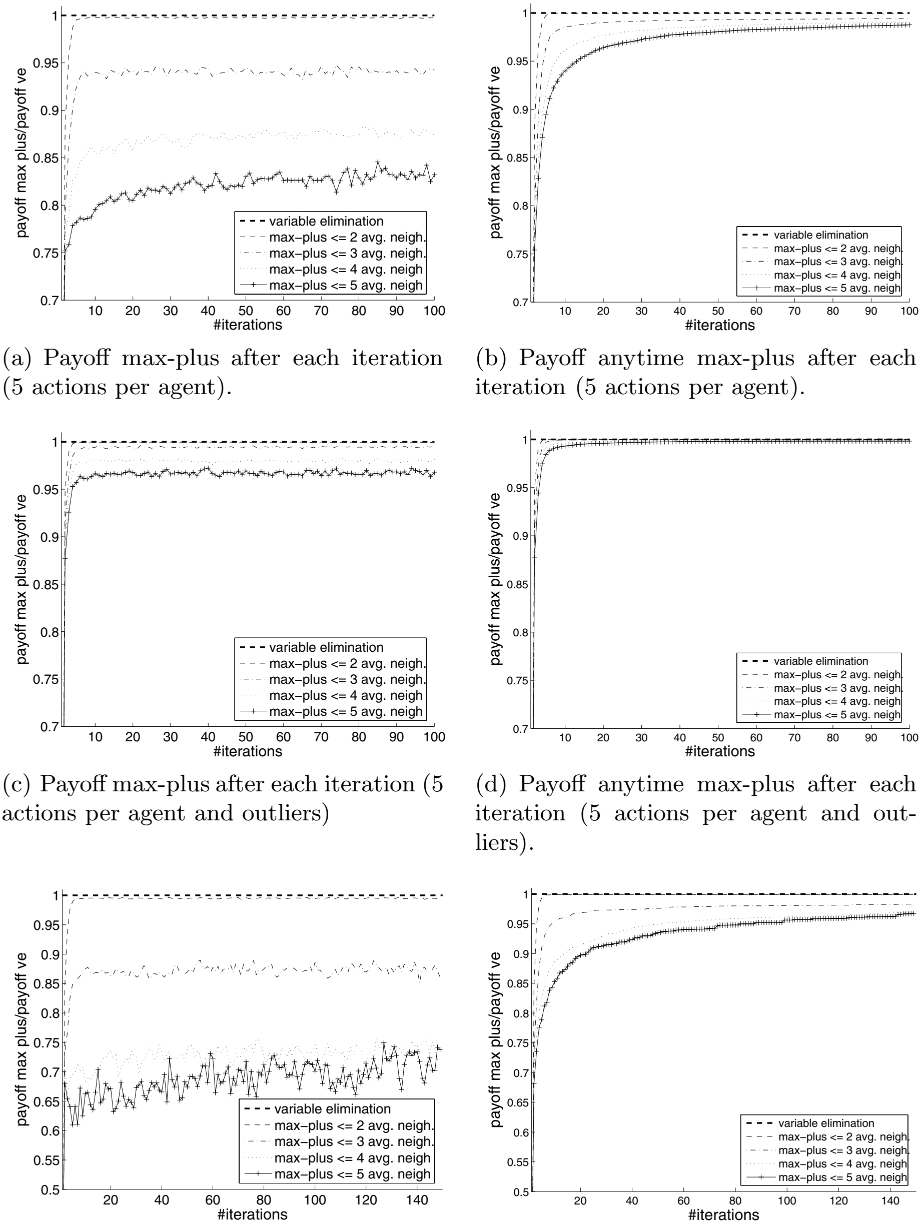
(c) Example graph with 37 edges (on average 4.93 neighbors per agent).

(d) Timing comparisons VE and max-plus (5 actions per agent).

(e) Timing comparisons VE and max-plus (5 actions per agent and outliers).

(f) Timing comparisons VE and max-plus (10 actions per agent).

Fig. 3. Example graphs and (average) timing results for both VE and max-plus for different graphs with 15 agents and cycles



(a) Payoff max-plus after each iteration (5 actions per agent).

(b) Payoff anytime max-plus after each iteration (5 actions per agent).

(c) Payoff max-plus after each iteration (5 actions per agent and outliers)

(d) Payoff anytime max-plus after each iteration (5 actions per agent and outliers).

(e) Payoff max-plus after each iteration (10 actions per agent).

(f) Payoff anytime max-plus after each iteration (10 actions per agent).

Fig. 4. Relative payoff with respect to VE for both max-plus with (graphs on the right) and without (graphs on the left) the anytime extension on different graphs with 15 agents and cycles

a neighboring agent to receive a conditional strategy involving agents it did not had to coordinate before, changing the graph topology to an even denser graph.

Fig. 4 shows the relative payoff found with the max-plus algorithm with respect to the optimal payoff after each iteration for graphs with different average numbers of neighbors. For the loosely connected graphs (less than two neighbors) the result is similar to the optimal result after a few iterations only. As the number of neighbors increases, the resulting policy becomes worse. This effect is less evident in the graphs with outliers (Fig. 4(c)) since certain action combinations are clearly preferred lowering the number of oscillations. Increasing the number of actions per agents (Fig. 4(e)) has a negative influence on the result because of the increase in the total number of action combinations.

Applying the anytime version as discussed in Section 3, improves the results for all graphs indicating that the failing convergence of the messages causes the algorithm to oscillate between different joint actions and 'forget' good joint actions. Fig. 4 shows that for all sets near-optimal policies are found, although it takes more iterations for the graphs with ten actions per agent to find them.

5 Conclusion and Future Directions

In this paper, we continued the work started in [6] and investigated further the usage of the max-plus algorithm as an alternative action selection method to variable elimination (VE) in coordination graphs (CG). VE is an exact method that will always report the optimal joint action, but is slow for densely connected graphs with cycles as its worst-case complexity is exponential in the number of agents. Furthermore, this method is only able to report a solution after the complete algorithm has ended. The max-plus algorithm operates by repeatedly sending local payoff messages over the edges in the CG. By performing a local computation based on its incoming messages, each agent is able to select its individual action. We provided empirical evidence that this method converges to the optimal joint action for tree-structured graphs (as shown by theory), and that it finds near optimal solutions in large, highly connected graphs with cycles an order of magnitude faster than VE. Another advantage of the max-plus algorithm is that it can be implemented fully distributed using asynchronous and parallel message passing. For these reasons, we believe max-plus is an appropriate action selection technique for cooperative real-time systems such as used in RoboCup.

As future research, we are planning to implement the max-plus algorithm in our UvA Trilearn 2D simulation team. In previous years, we used VE for cooperative action selection[5], but computational constraints restricted us in the number of coordination dependencies (see [5]). Using the max-plus algorithm we hope to be able to introduce more specialized fine-grained coordination.

Finally, we like to apply max-plus to sequential decision making. In [16, 4] CGs are used in combination with VE to learn coordinated policies of the agents using

[5] Since the agents in the 2D simulator cannot communicate directly, each agent models the complete algorithm separately using common knowledge assumptions (see [5]).

reinforcement learning. We like to investigate whether the usage of max-plus can help to learn the coordinated behavior for larger groups of agents.

Acknowledgments

This research is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414.

References

1. Weiss, G., ed.: Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
2. Vlassis, N.: A concise introduction to multiagent systems and distributed AI, Informatics Institute, University of Amsterdam (2003)
3. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. In: Proc. of the IJCAI-95 Workshop on Entertainment and AI/Alife. (1995)
4. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: Advances in Neural Information Processing Systems 14, The MIT Press (2002)
5. Kok, J.R., Spaan, M.T.J., Vlassis, N.: Non-communicative multi-robot coordination in dynamic environments. Robotics and Autonomous Systems **50** (2005) 99–114
6. Vlassis, N., Elhorst, R., Kok, J.R.: Anytime algorithms for multiagent decision making using coordination graphs. In: Proc. of the International Conference on Systems, Man and Cybernetics, The Hague, The Netherlands (2004)
7. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufman (1988)
8. Yedidia, J., Freeman, W., Weiss, Y.: Understanding belief propagation and its generalizations. In: Exploring Artificial Intelligence in the New Millennium. Morgan Kaufmann Publishers Inc. (2003) 239–269
9. Wainwright, M., Jaakkola, T., Willsky, A.: Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. Statistics and Computing **14** (2004) 143–166
10. Zhang, N.L., Poole, D.: Exploiting causal independence in bayesian network inference. Journal of Artificial Intelligence Research **5** (1996) 301–328
11. Bertelé, U., Brioschir, F.: Nonserial dynamic programming. Academic Press (1972)
12. Wainwright, M., Jaakkola, T., Willsky, A.: Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. Technical report, P-2554, LIDS-MIT (2002)
13. Crick, C., Pfeffer, A.: Loopy belief propagation as a basis for communication in sensor networks. In: Proc. of the 19th Conference on Uncertainty in AI. (2003)
14. Murphy, K., Weiss, Y., Jordan, M.: Loopy belief propagation for approximate inference: An empirical study. In: Proc. 15th Conf. on Uncertainty in Artificial Intelligence, Stockholm, Sweden (1999)
15. Loeliger, H.A.: An introduction to factor graphs. In: IEEE Signal Proc. Mag. (2004) 28–41
16. Kok, J.R., Vlassis, N.: Sparse Cooperative Q-learning. In Greiner, R., Schuurmans, D., eds.: Proc. of the 21st Int. Conf. on Machine Learning, ACM (2004) 481–488

Real-Time Diagnosis and Repair of Faults of Robot Control Software*

Gerald Steinbauer, Martin Mörth, and Franz Wotawa

Institute for Software Technology, Graz University of Technology
Inffeldgasse 16b/II, A-8010 Graz, Austria
{steinbauer, moerth, wotawa}@ist.tugraz.at

Abstract. Faults in hardware and software are not totally avoidable not even if the components are carefully designed, implemented and tested. In this paper we present a solution for detection, localization and repair of faults in the control software for autonomous mobile robots. The presented diagnosis system uses model-based diagnosis for fault detection and localization. Furthermore, we present a method which enables the robot control software to recover from located faults. The novelty of our approach is that fault localization and repair takes place at runtime. Moreover, we present experimental results of the proposed diagnosis system obtained in the RoboCup Middle-Size scenario.

1 Introduction

Even if the control software of a mobile robot is carefully designed, implemented and tested, there is always the possibility of faults in the system. Generally, faults are the deviation of the current behavior of a system from its desired behavior. For instance, we know very well situations in RoboCup Middle Size League (MSL) games where frequently robots had to be removed from and inserted into the field because some hardware or software components crashed or showed an undesired behavior. Carlson and Murphy [1] presented a quantitative evaluation of failures on mobile robots. The situation gets even worse if one thinks about autonomous robots, which operate for a long time without the possibility of human intervention, e.g. nuclear inspection robots, space probes or planetary rovers. Therefore, robustness and fault-tolerance are crucial for truly autonomous robots.

Because faults are not totally avoidable it is desirable that mobile robots are able to autonomously detect and repair such faults. If a permanent fault, e.g. broken hardware, is identified the robot should at least provide basic functionality or should be able to switch to a safe state. These requirements could be fulfilled if a dedicated diagnosis system is attached to the robot control software. Usually, a diagnosis system comprises three modules: (1) a monitoring module, (2) a fault detection and localization module and (3) a repair module. The first

* This research has been funded in part by the Austrian Science Fund (FWF) under grant P17963-N04 and Land Steiermark under grant 40Ro03-PE "RoboCup 2004".

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 13–23, 2006.

module observes the actual behavior of the hardware and software of the robot system. The fault detection uses observations and a model of the system's desired behavior to detect deviations between them. A deviation is equivalent to a detected fault. However, in practice the detection of a fault is not enough. The module should also identify the hardware or software component which caused the fault. If a fault and its location is identified the repair module tries to resolve the fault. This could happen by a restart or reconfiguration of the affected components.

There are many proposed and implemented approaches for fault detection and repair in autonomous systems. The Livingstone architecture by Williams and colleagues [2] was used on the space probe *Deep Space One* to detect failures in the probe's hardware and to recover from them. The fault detection and recovery is based on model-based reasoning. Model-based reasoning uses a logic-based formulation of the system model and the observations. The advantage is that well understood reasoning algorithms can be used. Model-based diagnosis has also been successfully applied to fault detection in digital circuits, car electronics and software debugging of VHDL programs [3]. Verma and colleagues [4] used particle filter techniques to estimate the state of the robot and its environment. These estimations together with a dynamic model of the robot were used to detect faults. The advantage of this approach is that it accounts for uncertainties of the robot and its environment as it derives the most probable state. But so far it was only applied to fault detection in the robots hardware and no repair actions were derived. Rule-based approaches were proposed by Murphy and Hershberger [5] to detect failures in sensing and to recover from them. Additional sensor information was used to generate and test hypotheses to explain symptoms resulting from sensing failures. Roumeliotis et. al. [6] used a bank of Kalman filters to track specific failure models and the nominal model. The filter residuals were post-processed to produce a probabilistic interpretation of the system's operation. Such methods are popular for linear systems affected by Gaussian noise.

Up to now solutions for fault detection and repair of robot control software at runtime are rare. Most previous research has dealt either with hardware diagnosis or diagnosis of software as part of the software engineering cycle. In [7] Melchior and Smart summarized ideas and requirements for robots that are aware of failures at runtime. In this paper we present a solution for real-time fault detection and repair of control software of autonomous robots. The fault diagnosis follows the model-based diagnosis paradigm [8]. It is based on observations of the current behavior of the control system's components, a model of the desired behavior of the control system's components and the dependencies between them. A monitoring module constantly observes the behavior of the control software. If a deviation of the desired behavior is observed a diagnosis kernel derives a diagnosis, i.e., a set of malfunctioning software components explaining the deviation. Based on this diagnosis and a model of the software components and their connections a repair module executes an appropriate repair action to recover the system from the fault. The proposed diagnosis system has been implemented and tested on our RoboCup MSL robots within the robotic soccer scenario. In the next section we describe the control software of our mobile robots. In section 3 we present the diagnosis system in more detail. Section 4 reports the results of experiments we conducted for the diagnosis system on our MSL robots. Finally, we draw some conclusions and give an outlook on future research.

2 Robot Control Software

The control software of our robots comprises different separated modules, called services. Each service runs as an independent process and implements a specific task, e.g., image processing, world modeling, planning. The control software is based on the MIRO framework [9]. MIRO is a CORBA-based software framework for robotics applications and provides a wide range of mechanisms for implementing services and for the communication between services. It is a very flexible framework which can be easily adapted to different robot platforms and applications [10].

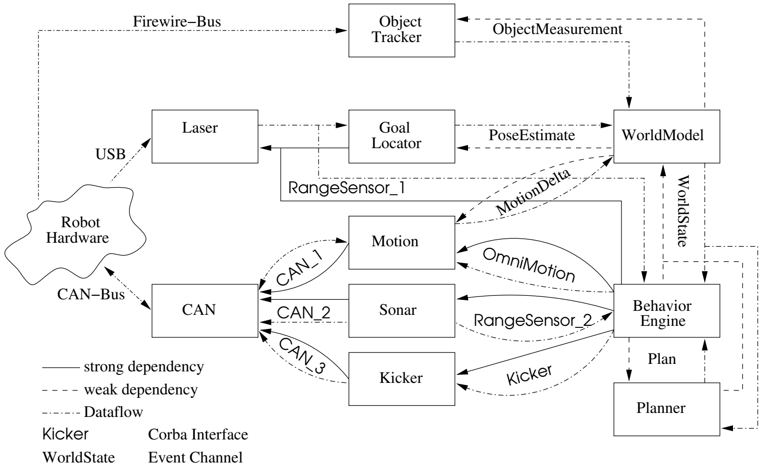


Fig. 1. Dependencies and data-flow within the robot control software

Figure 1 shows an overview of our robot control software. The software is organized in three levels with increasing abstraction. The Laser and CAN services provide low-level connection to the hardware of the robot. The connections are based on raw sensor data and low-level commands for actuators, e.g., laser scanner and the hardware modules on the CAN-Bus. The planner is located on top of the hierarchy and implements an abstract symbolic representation of the knowledge of the robot and its decision making process. The remaining services form the continuous level. Herein, all the computation of sensor inputs and the control of actuators on a continuous level are implemented, e.g., image processing, sensor fusion, execution of actions.

We use two different methods for the interactions and connections between the services: (1) remote CORBA method calls and (2) an event channel. Remote CORBA calls follow the client/server paradigm. One service, called the server, implements a specific function and exports a corresponding interface, e.g., the control interface for the omni-directional drive. The client, the service which uses the exported function, remotely invokes a method call on the server. The return value of the call may contain queried data. In Figure 1 remote CORBA calls are shown as solid lines directed to the server. The data-flow between server and client is shown as chain dotted lines. The Figure also shows the dependencies between services. Remote CORBA calls are called *strongly dependent* because a fault in the server directly affects the client.

The latter communication mechanism is the event channel. A server posts data via an event. All clients which are subscribed to this specific event are automatically informed if a new event is available. This connection type is shown as dashed lines and the data-flow direction is always directed from the server to the client. The event channel is called *weakly dependent*. The distinction between strong and weak dependencies is later important for the diagnosis and repair process.

The above described structure of the control software, the different types of connections and the dependencies between the services are used to build a model of the desired behavior of the control software. In the next section we describe how we use this model together with various observations of the behavior of services and connections to form a diagnosis system for the control software of our robots.

3 Diagnosis System

3.1 Monitoring

The task of the monitoring module of the diagnosis system is to observe the actual behavior of the control system. For this purpose we introduce the concept of observers. An observer monitors the behavior of a service or the communication between services. The observer determines a misbehavior of the control system if the observed behavior deviates from the specified behavior.

In the current implementation we use the following observers:

- *Periodic event production*: This observer checks whether a specific event e is at least produced every n milliseconds. An example for this observer is the event *MotionDelta* containing odometry data which is produced every 50 ms by the *Motion* service.
- *Conditional event production*: This observer checks whether an event e_1 is produced within n milliseconds after an event e_2 occurred. An example for this observer is the event *WorldState* which is produced by the *WorldModel* after an event *ObjectMeasurement* occurs.
- *Periodic method calls*: This observer checks whether a service calls a remote method m at least every n milliseconds. An example for this observer is the

RangeSensor interface of the *Sonar* service which is regularly called by the *BehaviorEngine*.

– *Spawn processes*: This observer checks whether a service spawns at least n threads. We know for instance if the *Motion* service works correctly it spawns six threads.

There are several requirements for the monitoring module. First, if observers are used there should be no or at least only a minimum necessity for changes in the control system. Furthermore, the monitoring component should not reduce the overall performance of the control system. Both requirements can be fulfilled easily by using mechanisms provided by CORBA [11] and the Linux OS. The first two observers are implemented using the CORBA event channel. The third observer is implemented using the CORBA portable interceptor pattern. The last observer is implemented using the information provided by the *proc* file-system of the Linux OS. For all these observers no changes are necessary in the control system. Furthermore, the computational power requirements for all the observers are negligible.

3.2 Diagnosis

A fault is detected if a observer belonging to the monitoring module recognizes a deviation between the actual and the specified behavior of the system. But so far we do not know which misbehaving service causes this fault. We use the model-based diagnosis (MDB) paradigm [8, 12] to locate the faulty service.

We will explain the principles of MDB with a simple example.

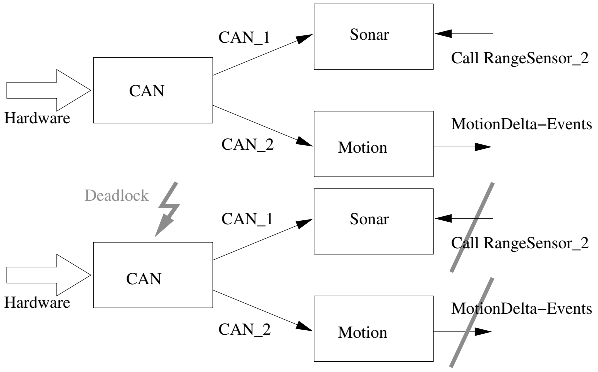


Fig. 2. Diagnosis of a fault in the Can-Service. The upper figure shows the desired behavior. The lower figure shows the behavior after a deadlock in the Can-Service.

Figure 2 shows an example for the diagnosis process in case of a malfunctioning CAN-Service. First, we build an abstract model of the correct behavior of the CAN-, Sonar and Motion Service. Therefore, we introduce two predicates: $AB(x)$ becomes true if a service x is abnormal, meaning x is malfunctioning.

$ok(y)$ becomes true if a connection y, either a remote call or an event, shows a correct behavior. The model for the correct behavior of the example could be described in the following clauses:

1. $\neg AB(CAN) \rightarrow ok(CAN\_1)$
2. $\neg AB(CAN) \rightarrow ok(CAN\_2)$
3. $\neg AB(Sonar) \wedge ok(CAN\_1) \rightarrow ok(RangeSensor\_2)$
4. $\neg AB(Motion) \wedge ok(CAN\_2) \rightarrow ok(MotionDelta)$

Lines 1 and 2 specify that if the CAN-Service works correctly also the connections *CAN_1* and *CAN_2* work correctly. Line 3 specifies that if the Sonar Service and its input connection *CAN_1* work correctly also the connection *RangeSensor_2* has to show a correct behavior. Line 4 specifies similar facts for the Motion Service.

If there is a deadlock in the CAN-Service, the Motion and Sonar services can not provide new events or calls as they get no more data from CAN. This fact is recognized by the corresponding observers and can be expressed by the clause: $\neg ok(RangeSensor\_2) \wedge \neg ok(MotionDelta)$. If we assume a correct behavior of the system expressed by the clause $\neg AB(CAN) \wedge \neg AB(Sonar) \wedge \neg AB(Motion)$ we get a contradiction. This means we have detected a fault.

Finding the service which caused the fault is equivalent to finding the set of predicates $AB(x)$ with $x \in \{CAN, Motion, Sonar\}$ that resolves the contradiction. These sets are called *diagnoses*, Δ. We are interested in finding a set with minimal cardinality, e.g., a single faulty service. These diagnoses are in general sufficient as multiple faults are unlikely. In this example the set $\{AB(CAN)\}$ with only one element is able to resolve the contradiction. Therefore, the faulty CAN-Service is located.

3.3 Repair

Once the diagnosis system has found one or more malfunctioning services responsible for a detected fault it should be able to recover the control system from this fault. Therefore, the repair module determines an appropriate repair action based on the described diagnosis and the dependencies between the services.

The repair action comprises a stop and a restart of the malfunctioning services. But we have to be careful, because restarting a specific service may also cause a restart of other services depending the restarted service. Therefore, the repair module takes the *strong dependencies* between services into account.

The appropriate repair action is derived in the following way: Put all members of the diagnosis Δ in a set R. The members of this set R are scheduled for restart. In the next step insert all services into R, which strongly depend on a member of R. Repeat this step until no more services are added to R. R now contains all services which have to be restarted. But first of all the scheduled services have to be stopped in an ordered way. This means to first stop all services which no other service strongly depend on. Afterwards, stop all services for which no more services are running which depend on them. This process is necessary to avoid additional crashes of services caused by a sudden stop of a service another service depends on. Hereafter, start all affected services in the reverse order. Services which were restarted because of a strong dependency on the malfunctioning services should be able to retain data which it had gained so far or they should be able to recover such data after a restart. Otherwise the control system may become inconsistent.

After this repair action took place, the robots control system is again in the desired state.

4 Experiments

The proposed diagnosis system has been implemented and tested on the robots of our RoboCup MSL team. The robot control system runs on an embedded Pentium III PC with 850 MHz clock rate and 256 MB of RAM. The operating system on the PC is an ordinary Linux system.

The diagnosis system itself is implemented as a separate process to minimize the interference with the existing control system. The diagnosis system implements the four types of observers described in Section 3. The use of Corba and OS services allows monitoring of the robot control system without any impact to it.

The model of the robot control system (software components, dependencies, observers) is specified in a XML file. Therefore, changes in the model or adaptation to other software systems are simple and straight forward. Table 1 shows a section of the specification of the used model describing the behavior of the Motion service.

Table 1. Model description for the motion service

```
<component id="MotionService">
  <rule class="SPAWNS-PROCESS-RULE">
    <property name="min-process-count">6</property>
  </rule>
  <rule class="DIRECT-DEPENDENCY-RULE">
    <property name="component-id">CanService</property>
  </rule>
  <rule class="PERIODIC-EVENT-PRODUCTION-RULE">
    <property name="max-sleep-time">150</property>
    <property name="event-name">MotionDelta</property>
  </rule>
</component>
```

The diagnosis system is divided into three modules: (1) a monitoring module, (2) a diagnosis kernel and (3) a repair module. The monitoring module starts all necessary observers according to the model description and regularly checks for violations of the observers. If such a violation is detected the diagnosis kernel is informed. The diagnosis kernel derives a diagnosis based on the model of the control system and the violated observations. The derivation of a diagnosis is started after a certain amount of time, i.e. 5 s, within no more changes in the states of the observers are detected. This is done for stability reasons as it takes a certain amount of time for all observers to recognize an improper behavior. The diagnosis will be communicated to the repair module. It executes the appropriate repair action to recover the control system. During the repair action no new diagnoses are derived. We do this for stability reasons as the repair action temporally may violate additional observers. After the repair action is completed the observers and the diagnosis kernel are started again.

For the evaluation of the proposed diagnosis system and its implementation we did several experiments on our mobile robots. We introduced artificial faults into the robot control system and analyzed if the diagnosis system detected and located the fault and recovered the control system. We used two different fault scenarios:

- *Killing a Service*: A certain software service is explicitly killed. This is equivalent to a crash of a certain service.
- *Deadlock a Service*: A deadlock is introduced to a certain software service. This is equivalent to a malfunctioning software service.

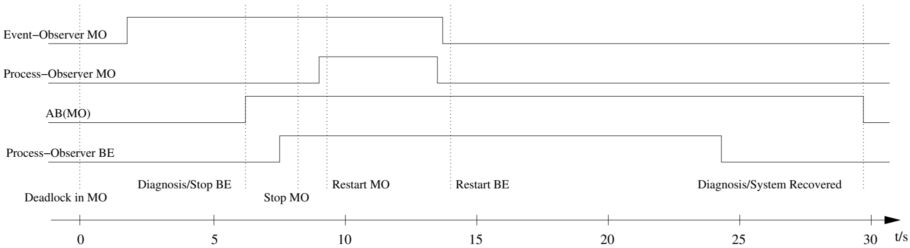


Fig. 3. Timing diagram for diagnosis and repair of a deadlock in the motion service

Figure 3 shows the timing diagram for the diagnosis and repair of an introduced deadlock in the motion service (MO). After introducing the deadlock in MO the Periodic Event Observer for the event *MotionDelta* detects that no more events are produced. After the waiting time the diagnosis kernel derived that MO is malfunctioning, *AB(MO)*. Instantly the repair process starts. The repair action comprises a stop of the Behavior Engine (BE), a stop of MO, and a restart of MO and BE. The restart of BE is necessary because BE strongly depends on MO. Again after the waiting time the diagnosis kernel derives the diagnosis that all services work properly now. Please note that no other services were affected by the repair process. The Figure also shows the fact that suspending the diagnosis kernel during the repair is necessary as observers report additional improper observations, e.g. Process Observer. The relatively long time for the recovery can be explained by the fact that stopping and starting of services can take a while because of the required starting, stopping and re-configuration of hardware components. Furthermore, stopping a service may consists of various

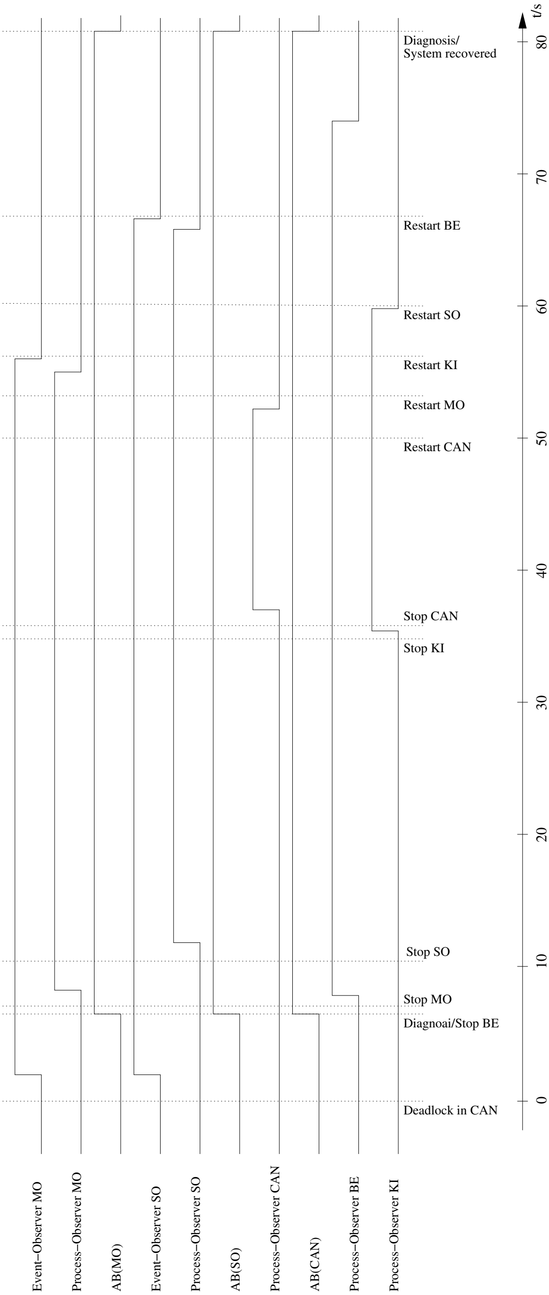


Fig. 4. Timing diagram for diagnosis and repair of a deadlock in the CAN service

steps. First, we send the service a interrupt signal (SIGINT). This allows a service to shutdown properly. If this does not stop the service within 5 seconds we send the service a termination signal (SIGTERM). Finally, we send a killing signal (SIGKILL) if after another 5 seconds still processes of the services run. The time for computing the diagnosis is negligible because it is less than 10 ms.

Figure 4 shows a more complex scenario. Here we introduce a deadlock in the CAN-Service. After introducing the deadlock, MO and the Sonar Service SO produce no more data because they get no more data from CAN. This fact is detected by the appropriate observers. Using the model of the observations, the components and its connections the diagnosis kernel recognizes the malfunctioning CAN. The repair action is similar to the example above except that more services are involved. After repair, the control system is again in the desired state.

We conducted also two experiments in which we killed a service. In the first experiment we killed the Laser Service. The diagnosis system successfully detected the fault and recovered the control system by restarting BE, Goal Locator and Laser. The recovery took 68 s. In a second experiment we killed the World Model. The diagnosis system successfully detected and repaired the fault. During this experiment it was important that the whole process took only 20 s because the system located the fault in the WM and no other service was affected.

We also tested the diagnosis system during games of a RoboCup MSL exhibition at our university. During the games the diagnosis system performed well. The image processing of one of our robots crashed twice during the games because of problems with a new camera. But the diagnosis system successfully detected, localized and repaired the fault at runtime.

The affect of the diagnosis system on the runtime performance of the robot control system is negligible. The diagnosis system uses less than 1 % of the CPU time and less than 5 % of the memory.

5 Conclusion and Future Work

In this paper we presented a diagnosis system capable of real-time fault detection, localization and repair for the control software of autonomous mobile robots. The proposed system follows the model-based diagnosis paradigm. It uses a general abstract model of the correct behavior of the control system together with observations of the actual behavior of the system to detect and localize faults in the software. Furthermore, we presented a repair method which is able to recover the software from a fault. Because of its general methods the proposed system is also applicable to other software than robot control software.

The proposed diagnosis system has been successfully implemented and tested on our RoboCup MSL robots. Experiments show that the system is able to detect and localize faults like crashed or deadlocked services at runtime. Furthermore, the system is able to recover the control software from such faults on the fly. Moreover, the proposed diagnosis system can be deployed on a robot platform with no changes in the existing control software and with nearly no effect on the runtime performance of the control system.

For future research it would be interesting to improve the used models by using more knowledge about the robot and its environment. Therefore, diagnosis of more complex systems and also the robots hardware would be possible. Another interesting issue would be the automatic reconfiguration of the robots hardware and software to recover from permanent faults, like broken hardware.

References

1. Jennifer Carlson and Robin R. Murphy. Reliability Analysis of Mobile Robot. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, ICRA 2003, September 14-19, 2003, Taipei, Taiwan.* IEEE, 2003.
2. B. C. Williams, P. Nayak, and N. Muscettola. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–48, August 1998.
3. Gerhard Friedrich, Markus Stumptner, and Franz Wotawa. Model-based diagnosis of hardware designs. *Artificial Intelligence*, 111(2):3–39, 1999.
4. V. Verma, G. Gordon, R. Simmons, and S. Thrun. Real-time fault diagnosis. *IEEE Robotics & Automation Magazine*, 11(2):56 – 66, June 2004.
5. Robin R. Murphy and David Hershberger. Classifying and recovering from sensing failures in autonomous mobile robots. In *AAAI/IAAI, Vol. 2*, pages 922–929, 1996.
6. S.I. Roumeliotis, G.S. Sukhatme, and G.A. Bekey. Sensor fault detection and identification in a mobile robot. In *IEEE Conf on Intelligent Robots and Systems*, pages 1383 – 1388, Victoria, Canada, 1998.
7. Nik A. Melchior and William D. Smart. Autonomic systems for mobile robots. In *Proceedings of the International Conference on Autonomic Computing (ICAC 2004)*, pages 280–281, 2004.
8. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
9. Hans Utz, Stefan Sablatng, Stefan Enderle, and Gerhard K. Kraetzschmar. Miro – middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, 18(4):493–497, August 2002.
10. Gordon Fraser, Gerald Steinbauer, Arndt Mühlenfeld, and Franz Wotawa. A modular architecture for a multi-purpose mobile robot. In *Proc. of the 17th Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 3029 of *Lecture Notes in Artificial Intelligence*. Springer, 2004.
11. Michi Henning and Steve Vinoski. *Advanced CORBA©Programming with C++*. Addison Wesley Professional, 1st edition, 1999.
12. Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

Exploiting the Unexpected: Negative Evidence Modeling and Proprioceptive Motion Modeling for Improved Markov Localization

Jan Hoffmann, Michael Spranger, Daniel Göhring, and Matthias Jüngel

Institut für Informatik
LFG Künstliche Intelligenz
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany
http://www.aiboteamhumboldt.com

Abstract. This paper explores how sensor and motion modeling can be improved to better Markov localization by exploiting deviations from expected sensor readings. Proprioception is achieved by monitoring target and actual motions of robot joints. This provides information about whether or not an action was executed as desired, yielding a quality measure of the current odometry. Odometry is usually extremely prone to errors for legged robots, especially in dynamic environments where collisions are often unavoidable, due to the many degrees of freedom of the robot and the numerous possibilities of motion hindrance. A quality measure helps differentiate the periods of unhindered motion from periods where robot motion was impaired for whatever reason. Negative evidence is collected when a robot fails to detect a landmark that it expects to see. Therefore the gaze direction of the camera has to be modeled accordingly. This enables the robot to localize where it could not when only using landmarks. In the general localization task, the probability distribution converges more quickly when negative information is taken into account.

1 Introduction

Selflocalization, the estimation of position and orientation of a mobile robot, remains an important and valuable task for mobile robotics. One of the most successfully applied approaches is called *Monte-Carlo-Localization.* This method is used in numerous robot navigation problem domains, such as office navigation [1], museum tour guides [13], RoboCup [7], as well as outdoor or less structured environments [9]. We propose 2 extensions affecting the sensor model as well as the motion model.

1. We show how *negative information* can be incorporated into Monte Carlo localization. The sensor model is extended by modeling the probability of non-detection events.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 24–35, 2006.

2. The motion model is improved through careful modeling of proprioceptive information. The resulting model is incorporated into the action update of the particle filter.

The adjustments and changes presented improve the general ability to localize and also allow the robot to localize in areas where it was previously unable. They enable the robot to quickly recover its belief after collision events and to adjust quickly to large displacements (kidnapped robot).

Negative information denotes the ascertained absence of expected sensor readings. This is incorporated into the current belief much like an additional sensor. Proprioception is based on the comparison of actual motion to intended motion. This information is used to enhance the influence of action commands onto the belief. The extensions each prove to be a useful addition to the particle filter used, but are not particle filter specific approaches and can be used in other Bayes Filters. The positive impact on localization is shown in real world experiments using the Sony Aibo ERS-7 robot.

2 Monte Carlo Localization

The Monte Carlo Localization method is a probabilistic method, utilizing Bayes law and the Markov assumption. The robot maintains a set of samples, called particles. The particles approximate the belief of the robot's position, a probability distribution over the possible positions of the robot. The current belief of the robot's position is modeled as particle density, allowing for multi-modal probability distributions and beliefs. Each particle represents a hypothetical position of the robot. Belief $Bel(s_t)$, the localization estimate at time t, to be at position s_t is determined by *all* previous robot actions u_t and observations z_t. Using Bayes law and the Markov assumption, $Bel(s_t)$ can be written as a function that only depends on the previous belief $Bel(s_{t-1})$, the last robot action u_{t-1}, and the current observation z_t:

$$Bel^{-}(s_t) \longleftarrow \int \underbrace{p(s_t|s_{t-1}, u_{t-1})}_{\text{motion model}} Bel(s_{t-1}) ds_{t-1} \tag{1}$$

$$Bel(s_t) \longleftarrow \eta \underbrace{p(z_t|s_t)}_{\text{sensor model}} Bel^{-}(s_t) \tag{2}$$

with normalizing constant η. Equation 1 shows the *a priori* belief $Bel^{-}(s_t)$ which takes into account the previous belief and propagates it using the motion model of the robot. It is the belief prior to the measurement. The measurement is then incorporated into the belief as described in (2) using the sensor model ('sensor updating'). In Markov localization, given an initial belief $Bel(s_0)$ at $t = t_0$, the robot updates its belief using odometry and then incorporates new sensor information. Each time new information arrives the robot updates its particle distribution using the previous motion command, the resulting distribution is updated using the gathered sensor information. This 2 step operation requires 2 models. The *motion model* $p(s_t|s_{t-1}, u_{t-1})$ tries to model the effect of motion commands on the hypothetical positions. The *sensor model* incorporates environment and sensor information regarding this environment into the current belief. The particle filter employed for our work is based on the method described in [10]. Here particles consist of a robot pose and a probability. The robot pose (x, y, θ) represents the position and orientation of the robot (x,y coordinates on the field in mm and orientation in radians). The likelihood p is a measure of the plausibility of the hypothesis being at the specified robot pose. The approach first moves all particles according to the motion model of the action chosen. Afterwards the probabilities of the particles are adjusted using the sensory input and the sensor model. In a third step, called *resampling* particles are moved, deleted from the particle set or injected from observation, based on their probability.

3 Proprioceptive Motion Modeling

If obstacle avoidance fails robots are often unable to detect collisions since many designs, like the robot used in this work, lack touch sensors or bumpers. Apart from the current action failing, collisions (and subsequently being stuck) have severe impact on the robot's localization if odometry is used to any degree in the localization process. For these approaches to be robust against collisions, they tend to not trust odometry much. This paper is based on work dealing with collisions detection for a Sony Aibo using the walking engine and software framework described in [3]. The approach uses the servo motor's direction sensors for the task of estimating the quality of the odometry data gathered by the walking engine. In analogy to biology we call it proprioception because intrinsic data of the leg sensors is used.

3.1 Motion Model

The *motion model* consists of consecutive acquired odometry data incorporated into the Belief, as well as a random error Δ_{error}, which is related to the distance traveled and the angle rotated. Every particle is updated using the odometry offset accumulated since the last update.

$$pose_{\text{new}} = pose_{\text{old}} + \Delta_{\text{odometry}} + \Delta_{\text{error}} \tag{3}$$

Where Δ_{error} is defined as

$$\Delta_{\text{error}} = \begin{pmatrix} 0.1d \times \text{random}(-1\ldots1) \\ 0.02d \times \text{random}(-1\ldots1) \\ (0.002d + 0.2\alpha) \times \text{random}(-1\ldots1) \end{pmatrix} \tag{4}$$

3.2 Collision Detection

The Aibo is not equipped with sensors to directly perceive the contact with obstacles. We have shown ways of detecting collisions using the sensor readings

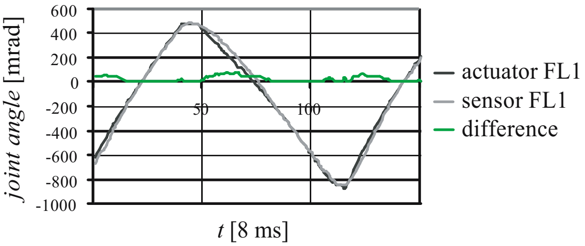


Fig. 1. Sensor and actuator data (shoulder joint FL1) for a freely walking robot. The corresponding difference function shows discrepancies between actuator and sensor data, caused by walking motions (peaks in the curve).

from the servo motors of the robot's legs in [3]. The comparison of motor commands and actual movement (as sensed by the servo's position sensor) can be used to detect collisions (see fig.1). This comparison has to compensate for the phase shift between the two signals and has to cope with arbitrary movements and accelerations produced by the behavioral layers of the robot. The method provides a *virtual collision sensor* that can be used to improve the motion model.

3.3 Extended Motion Model

The extended motion model accounts for the supplementary information provided by the collision detection module, by changing Δ_{error} as well as affecting the accumulated odometry update data in a random way. The binary decision of the collision sensor has a static impact on the motion noise. This means that Δ_{error} is no longer dependent on the distance traveled and the angle rotated, but rather is a uniform noise, within an interval expected to be a possible outcome of collisions. But also odometry data can not be fully relied upon, which is accounted for by randomly updating particles through the gathered odometry information, with the assumption that the robot most probably ends up somewhere between the requested destination and the starting point. The noise tries to account for the severe and unforeseeable impact of the collision. If collisions *are detected*, every particle is updated by:

$$pose_{\text{new}} = pose_{\text{old}} + \text{random}(0...1) \cdot \Delta_{\text{odometry}} + \Delta_{\text{error}}$$

Where Δ_{error} is

$$\Delta_{\text{error}} = \begin{pmatrix} 40 \times \text{random}(-1 \ldots 1) \\ 40 \times \text{random}(-1 \ldots 1) \\ 0.5 \times \text{random}(-1 \ldots 1) \end{pmatrix} \tag{5}$$

Otherwise, when no collision was detected, the motion model is not extended and the update is performed as usual(3). The effect of the changes are illustrated in fig.2 and 3.

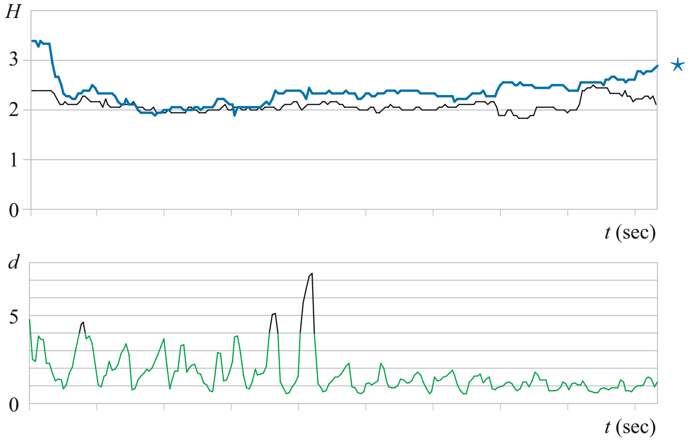


Fig. 2. Collision Experiment: A robot starts walking from the center circle in the direction of the goal and turns left before reaching the penalty area. The distributions entropy with (*) and without the extended motion model (*top*) and the corresponding collision "sensor" (*bottom*, values greater than 4 are interpreted as a collision).

Entropy. We use the expected entropy H as an information theoretical quality measure of the position estimate $Bel(s_t)$ [2]:

$$H_p(s_t) = -\sum_{s_t} Bel(s_t)\log(Bel(s_t)) \tag{6}$$

The sum runs over all possible states. The entropy of the particle distribution becomes zero if the robot is perfectly localized in one position. Maximal values of H mean that $Bel(s_t)$ is uniformly distributed.

Fig. 3 illustrates the effect of the described motion modeling on the particle distribution. A robot is walking from the center circle in the direction of the goal when a collision occurs. It then continues towards the goal and turns left before reaching the penalty area. When the collision is modeled, the uncertainty in the belief is clearly visible and can be used to trigger appropriate robot behavior.

4 Negative Evidence Modeling

The used localization algorithm uses field lines and landmarks to perform the sensor updating of the current belief. We extend this approach through the use of *negative information.* Two main reasons make it hard to actually implement a system that integrates negative evidence: the target (landmark) may not be there or the sensor may simply be unable to detect the target (due to occlusions, sensor imperfections, imperfect image processing, etc.). Differentiating the two cases requires careful sensor modeling. We address this problem by considering the field of view of the robot and by using obstacle detection to estimate occlusions. Negative information has been used in object

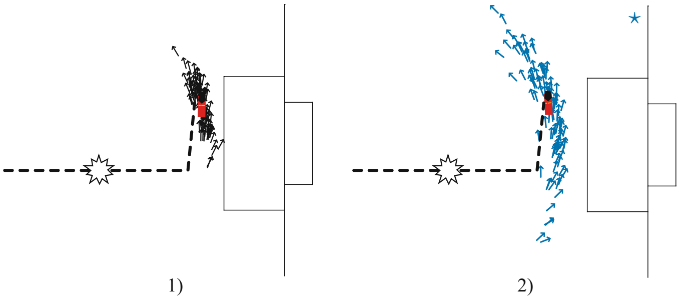


Fig. 3. Belief distribution without (2) and with (3) odometry quality used after a collision (marked by the star on the robot's path)

tracking (see [12] for an introduction and [5] for an overview). Not seeing the ball on the RoboCup field causes Monte Carlo particles to be deleted in that particular region [6]; occlusions are considered by explicitly modeling other robots' positions.

4.1 Sensor Model

Whenever the robot senses a landmark, the localization estimate is updated using the sensor model. This sensor model is acquired before the actual run. It describes the probability of the measurement z given a state s (position, orientation, etc.) of the robot. If no landmark is detected, the state estimation is updated using (only) the motion model of the robot. Sensory input in our case is measured bearings to landmark. The approach has been extended by accounting for field lines and edges [11] which require a slightly different model and method. The probability of the particles is derived from the measured bearings to landmarks.

4.2 Negative Information

Negative information describes the absence of a sensor reading in a situation where a sensor reading is expected given the current position estimate. To integrate negative information, imagine a binary sensor being added that fires whenever the primary sensor *does not* detect a particular landmark l. Its probability of it firing is given by:

$$p(z^{\star}_{l,t}|s_t) \tag{7}$$

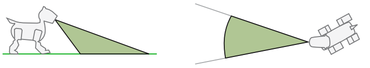
This sensor model can be used to update the robot's belief whenever it fails to detect a landmark, i.e. when negative evidence is acquired. This rather coarse way of incorporating negative information can be refined by taking into account the range r_t of the robot's sensors and possible occlusions o_t of landmarks. The sensing range is the physical volume that the sensor is monitoring. In case of a stationary robot, $r_t = r_0$ is constant, for a mobile robot with a pan-tilt camera it is not. By o_t we denote a means of detecting the occurrence of occlusions. In practice, this can be calculated from a map of the environment, directly sensed by a sensor such as a laser range finder, or derived from a model of moving objects in the environment. Combining the two yields the probability of not sensing an expected landmark l is:

$$p(z^\star_{t,l}|s_t, r_t, o_t) \tag{8}$$

The absence of the detection of a landmark can be used in the sensor update step of the Iterative Bayesian Updating (see Algorithm 1).

4.3 Extended Sensor Model

Field of View. Here we show the camera of the Sony Aibo ERS-7. The ERS-7 is a legged robot with a camera mounted in its head. The camera has a horizontal opening angle of 55° and the robot's head has 3 degrees of freedom (neck tilt, head pan, head tilt). We abbreviate gaze direction by $\varphi = (\varphi_{\text{tilt1}}, \varphi_{\text{pan}}, \varphi_{\text{tilt2}})$. The sensing range is calculated by considering the field of view (FOV) of the robot:



Occlusion. In order to account for occlusions, we opted for an approach that has been used successful for detecting obstacles, referred to as 'visual sonar' [4, 8]: The camera image is scanned in vertical scan lines and unoccupied space in the plane of the field is detected, since it can only be of green or white color (field lines). Scanning for these colors tells the robot where obstacles are and where there is free space, which in turn can be used to determine whether the visibility of the landmark was impaired, i.e. if it was occluded by another robot or some other obstacle. More specifically, if the expected landmark lies in an area where the robot has detected free space, the likelihood of the corresponding pose estimate is decreased. If it lies outside of the detected free space, no information can be inferred. Taking FOV and occlusion into account, the sensor model for not perceiving an expected landmark is given by:

$$p(z^\star_t|s_t, z_{t,\text{obstacle}}) \tag{9}$$

Where $s_t = (x_t, y_t, \vartheta_t, \varphi_t)$ describes the robot state that consists of the *robot pose* (position x_t, y_t, and orientation ϑ_t) and the current gaze direction φ_t. These are used to calculate the field of view of the camera. The sensor updating part of the localization code was adjusted to account for FOV and occlusion as described above. This means that sensor updating is triggered whenever there is a new camera image regardless of whether or not there was a percept. Before re-sampling, the weight of an individual particle is calculated as follows: Of all landmarks L, a subset of landmarks L' is detected, the subset $L^\star$ is expected but not detected, and lastly the subset $L^\diamond$ is not detected, but was also not expected,

Algorithm 1. Iterative Bayesian updating incorporating negative evidence

```
$Bel^{-}(s_t) \longleftarrow \int p(s_t|s_{t-1}, u_{t-1})Bel(s_{t-1})ds_{t-1}$
if (landmark $l$ was detected) then
    $Bel(s_t) \longleftarrow \eta p(z_t|s_t)Bel^{-}(s_t)$
else
    $Bel(s_t) \longleftarrow \eta p(z^{\star}_{t,l}|s_t, r_t, o_t)Bel^{-}(s_t)$
end if
```

$L = L' \cup L^{\star} \cup L^{\diamond}$ and $L^{\star} \cap L' = \emptyset$. The probability of a particle p_i is calculated by multiplying all the gathered evidences:

$$p_i = \underbrace{\prod_{l \in L'} s_l(\alpha_{\text{measd}}, \alpha_{\text{expd}}) \cdot}_{\text{detected}} \underbrace{\prod_{l \in L^{\star}} s^{\star}_l(\varphi, \alpha_{\text{expd}})}_{\text{expected and not detected}} \tag{10}$$

The function s_l is an approximation of the sensor model and returns the likelihood of sensing the landmark l at angle α_{measd} for a particle p_i that expects this landmark to be at α_{expd}. Function $s^{\star}_l$ models the probability of not sensing the expected landmark $l^{\star}$ given the current sensing range as determined by φ, the robot pose associated with p_i, and the obstacle percept z_{obstacle}.

4.4 Experimental Results

Localization Experiment. The following experiment is a localization task on the real robot. The robot is placed on the field in front of a landmark facing outwards. The robot performs a scanning motion with its head (pan range $[-45°, 45°]$) but does not move otherwise. From where it is standing, it can only see one landmark. A panorama composed of actual robot camera images is shown in fig. 4. The *a priori* belief is assumed uniform. This position was chosen because it is a particulary difficult spot for the robot to localize given the limited sensor information. The bearing α_l to a landmark is used for localization because the distance measurement is prone to errors. Using just the bearing, only the orientation of the robot can be inferred.

In the following paragraphs, the basic localization excluding the use of negative information and localization incorporating negative information are compared. We will first give a more qualitative analysis of the particle distribution and then show how the entropy of the distribution decreases when negative information is considered.

Particle Distribution. The basic experiment was conducted using 100 particles for Monte Carlo localization. It was repeated on a log file (containing camera images and robot joint angles) using an increased particle count of 2000. This was done to reveal artifacts due to the small number of particles used in the standard implementation.

Not using negative information. Without using negative information, the robot is unable to localize (fig. 5). Only the orientation of the particles is adjusted

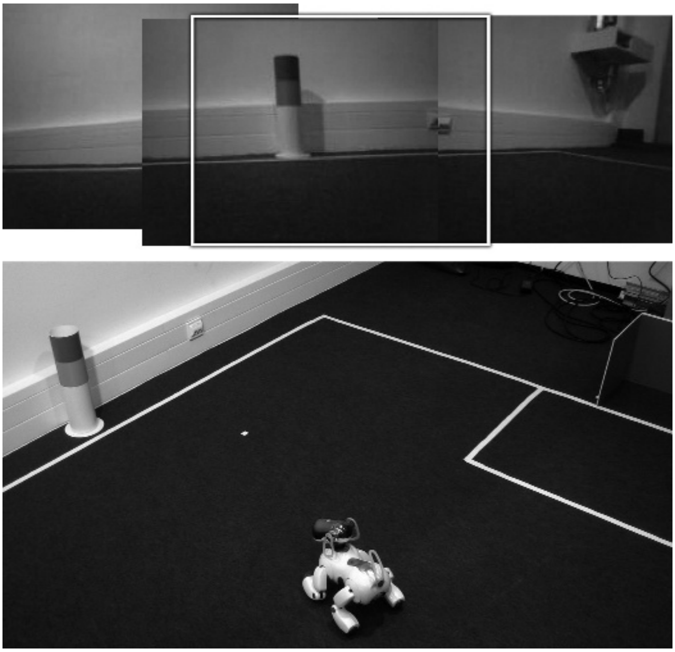


Fig. 4. *Top:* A panoramic view generated from actual camera images, single camera image highlighted. The robot can only see *one* landmark. *Below:* Photo of the experimental setup in our lab.

according to the sensor readings. The apparent clustering in fig. 5 is not stable and even after considerable time, the particles do not converge. The distribution for the larger sample set is uniform (w.r.t. position). Note that the distribution is not circular because the distance to the landmark was not used. Instead, only the bearing to the landmark was used. This results in a radial distribution resembling magnetic field lines.

Incorporating negative information. The negative information gained in this experiment is only seeing one landmark within the pan range. Incorporating this information, the robot is able to localize quickly. On average, the robot is reasonably well localized after 5-10 secs with a pose error of less than $\varDelta p = (35 \text{ cm}, 35 \text{ cm}, 20^o)$.

Entropy. We now consider the entropy of the particle distribution as defined earlier. Fig. 6 shows the progression of the distribution's entropy over time for the above localization experiment calculated from the 100 particle distribution.

Not using negative information. The experiments starts with a uniform particle distribution which equals to maximum entropy. When the landmark comes into view, a decrease in entropy is observed. This information gain is caused by the robot now knowing its relative orientation w.r.t. the landmark. Since there are

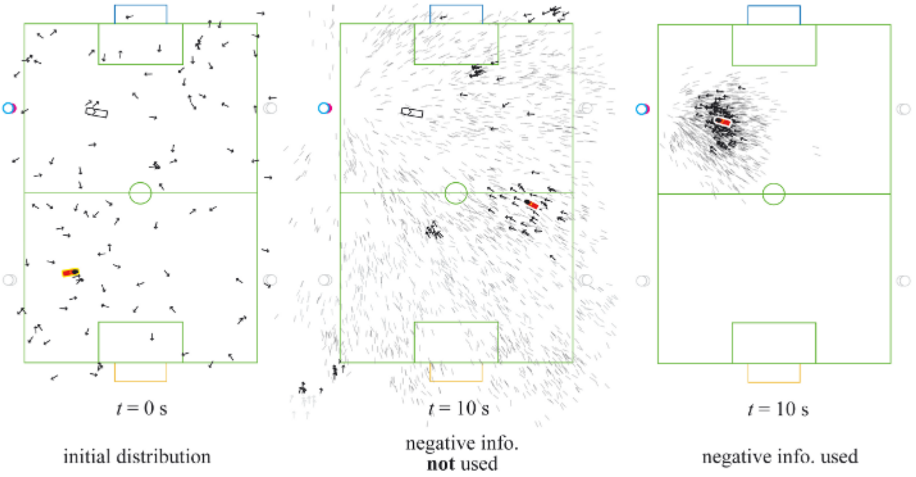


Fig. 5. Particle distribution not using negative information, initial uniform distribution and distribution after 10s. Solid arrows indicate Monte Carlo particles (100). The experiment was repeated using 2000 particles (shaded lines) to better represent the actual probability distribution. The actual robot position is indicated by the white symbol, the calculated robot pose by the solid symbol. Not using negative information and only using the bearing to the landmark the robot is unable to localize. Some clusters of particles form but they do not converge. As one would expect, the position distribution is almost uniform but the relative angle is quite distinct. *Right.* Particle distribution when negative information is incorporated, enabling the robot to localize quickly.

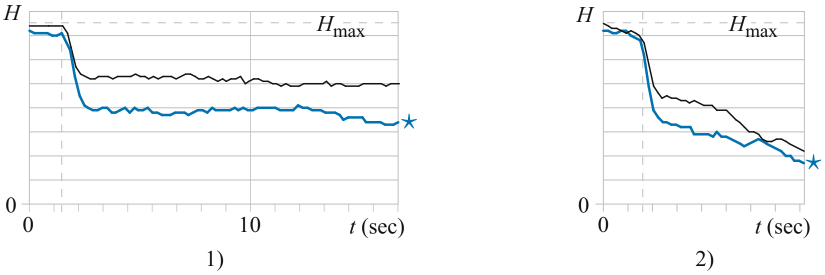


Fig. 6. Expected entropy of the belief in the localization task with (*) and without (thin line) using negative information. 1) At first the robot does not see the landmark. As soon as the landmark comes into the robot's view (indicated by the dashed vertical line), the entropy drops. Using negative information, the quality of the localization is greatly improved and the entropy continues to decrease over time. Note that the entropy decreases even before the landmark has been seen for the first time. 2) Additionally using field lines for localization enables the robot to localize. Incorporating negative information, the rate of convergence is higher and the entropy is significantly lower than without using it.

no constraints on the robot's position, the entropy remains at a relatively high level. Note that even though there is a drop in entropy, the localization estimate itself is still highly uncertain.

Incorporating negative information. When using negative information, the entropy decreases even before the first sensor reading. The information gain is much smaller than that caused by perceiving a landmark, but nevertheless noticeable. As soon as there is a percept, the negative information in combination with the knowledge of the robot's orientation results in a quick convergence of the particle distribution towards the actual robot pose. Remember that without using negative information no localization was observed.

Using field lines for localization. In our last experiment, field lines were used for localization in addition to landmarks. This enables the robot to localize quickly at the actual robot pose even when using the basic localization. Adding negative information, however, greatly increases the rate of convergence and the overall level of entropy is reduced even further. It is noteworthy that the decrease of entropy when incorporating negative information is not obscured by the usage of lines for localization (which offer a much higher information content than negative information).

5 Conclusion

We have demonstrated the power of integrating negative information as well as information about collisions into Markov localization.

We have shown how an odometry-based motion model can be improved using the knowledge about collisions with obstacles. This knowledge has been obtained by comparing the motor commands and the sensor readings of the leg joints. In the case of a collision the influence of the odometry on the motion model was reduced and extra noise was added that models the impact of an obstacle.

Incorporating negative information into the sensor model makes localization more stable even in areas where landmarks are rarely visible. The usage of negative information does, however, require very careful modeling. To avoid false negatives, the model needs to take into account the sensor's sensing range and possible occlusions of landmarks. We have presented how such modeling can be achieved for a Sony Aibo robot in the RoboCup environment. In real robot experiments, we have shown that using negative information, a robot is able to localize in positions where it otherwise would not. The entropy of the distribution is greatly reduced when negative information is incorporated and the rate of convergence towards the estimated position is increased. Future work will focus on how negative information can be used for other types of landmarks (e.g. field lines) and other sensors. Performance evaluation will be continued in more complex situations and the possibilities of reducing the number of particles necessary for robust Monte Carlo localization will be investigated.

Acknowledgments

Program code used was developed by the GermanTeam, a joint effort of the Humboldt University of Berlin, University of Bremen, University of Dortmund, and the Technical University of Darmstadt. Source code is available for download at http://www.germanteam.org.

References

1. D. Fox, W. Burgard, F. Dellart, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proc. of AAAI*, 1999.
2. D. Fox, W. Burgard, and S. Thrun. Active Markov Localization for Mobile Robots. In *Robotics and Autonomous Systems*, 1998.
3. J. Hoffmann and D. Göhring. Sensor-Actuator-Comparison as a Basis for Collision Detection for a Quadruped Robot. In *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2005.
4. J. Hoffmann, M. Jüngel, and M. Lötzsch. A Vision Based System for Goal-Directed Obstacle Avoidance. In *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2005.
5. W. Koch. On Negative Information in Tracking and Sensor Data Fusion. In *Proceedings of the Seventh International Conference on Information Fusion*, pages 91–98, 2004.
6. C. Kwok and D. Fox. Map-based Multiple Model Tracking of a Moving Object. In *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2005.
7. S. Lenser, J. Bruce, and M. Veloso. CMPack: A Complete Software System for Autonomous Legged Soccer Robots. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 204–211. ACM Press, 2001.
8. S. Lenser and M. Veloso. Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision. In *Proceedings of IROS'03*, 2003.
9. M. Montemerlo and S. Thrun. Simultaneous Localization and Mapping with Unknown Data Association Using FastSLAM. 2003.
10. T. Röfer and M. Jüngel. Vision-Based Fast and Reactive Monte-Carlo Localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-2003), Taipei, Taiwan*, pages 856–861, 2003.
11. T. Röfer and M. Jüngel. Fast and robust edge-based localization in the sony four-legged robot league. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004.
12. S. Särkkä, T. Tamminen, A. Vehtari, and J. Lampinen. Probabilistic Methods in Multiple Target Tracking, Research Report B36. Technical report, Laboratory of Computational Engineering Helsinki University of Technology, 2004.
13. S. Thrun, D. Fox, and W. Burgard. Monte Carlo Localization with Mixture Proposal Distribution. In *Proc. of the National Conference on Artificial Intelligence*, pages 859–865, 2000.

Playing Soccer with RoboSapien

Sven Behnke, Jürgen Müller, and Michael Schreiber

Albert-Ludwigs-University of Freiburg, Computer Science Institute
Georges-Köhler-Allee 52, 79110 Freiburg, Germany
{behnke, jmuller, schreibe}@informatik.uni-freiburg.de

Abstract. Due to limited availability of humanoid robots and the high costs involved, multi-agent experiments with humanoid robots have been at least difficult so far. With the introduction of RoboSapien, a low-cost humanoid robot developed for the toy market, this situation has changed.

This paper describes how we augmented multiple RoboSapiens to obtain a team of soccer playing humanoid robots. We added a Pocket PC and a color camera to the robot base to make it autonomous.

For a team of these augmented RoboSapiens, we implemented computer vision and self localization. We designed basic soccer skills, such as approaching the ball, dribbling the ball towards the goal, and defending the goal. We set up a soccer field and played test games in our lab to evaluate the system.

The paper reports experiences made during these soccer matches as well as results on a scoring task. We also tested this system at RoboCup German Open 2005, where we played soccer matches against the Brainstormers Osnabrück, who also used augmented RoboSapiens.

1 Introduction

To work towards the long-term goal of winning against the FIFA world champion, the RoboCup Federation added in 2002 a league for humanoid robots to their annual soccer championships. The RoboCup Humanoid League competition rules [14] require the participating robots to have a human-like body plan. They must consist of a trunk, two legs, two arms, and a head. The only allowed mode of locomotion is bipedal walking. The robots must be fully autonomous. No external power, computing power, or remote control is allowed.

Because the humanoid robots have not been ready for playing soccer games so far, the robots had to demonstrate their capabilities by solving a number of subtasks. In the Humanoid Walk they had to walk towards a pole, to turn around it, and to come back to the start. Scoring was based on walking speed and stability. In the Penalty Kick competition two robots faced each other. While one robot tried to score a goal, the other defended. In the Freestyle competition, the robots had five minutes to show a performance to a jury. Each year, there is also a new technical challenge. In 2004, it consisted of an obstacle walk, a passing task, and balancing across a sloped ramp.

The teams which participated in the Humanoid League chose very different robot platforms. Most teams constructed their own robots (e.g. Robo-Erectus [30]). A few teams used expensive humanoid robots developed by the

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 36–48, 2006.

Japanese industry, e.g. Hoap-2 [16] or Honda Asimo [8]. Some teams purchased servo-driven commercial robots or robot kits, e.g. from iXs [10] or Vstone [26].

The performance of the robots in the Humanoid League improved over the three competitions. In 2004, Team Osaka won the competition with the robot VisiON [21]. This robot used an omnidirectional camera as head. As a goalie, it could defend against a shot by jumping to the ground. Afterwards, VisiON got up without help. Another highlight of the 2004 competition was the passing demonstration between two Hoap-2 robots of team Senchans A [16].

Despite these impressive achievements, the overall performance of the RoboCup humanoids is still far from perfect. Basic soccer skills, such as robust dynamic walking and kicking without loosing balance are not possessed by all robots. Moreover, due to the high price involved, it exceeds the resources of most research groups to buy or construct more than one robot. To play soccer, however, a group needs a number of players in order to field a team.

Fortunately, RoboSapien, a low cost commercial humanoid robot, hit the market in 2004. In its original version, it is controlled by a human operator. We found a way to make it autonomous by augmenting it with a Pocket PC and a camera. Due to the low cost of this solution, it is not hard to obtain multiple augmented RoboSapiens. Since basic problems, such as dynamic walking, are solved when using this robot base, one can focus on higher-level issues, such as visual perception, self localization, behavior control, and communication.

The paper is organized as follows. The next section reviews some of the related work. Section 3 presents the original RoboSapien. In Section 4, we describe how we augmented it with a Pocket PC and a camera. Section 5 covers visual perception and self localization on the soccer field. Behavior control for making it play soccer is detailed in Section 6. In Section 7, we describe some infrastructure components needed to support a team of soccer playing robots. Section 8 reports experiences made during test games and presents experimental results on a scoring test. The paper concludes with a discussion of the feasibility of using RoboSapien for soccer competitions.

2 Related Work

Humanoid robots are not only used to play soccer. The human-like body has advantages when the robots are used in environments designed for humans. It facilitates multimodal communication with humans and imitation learning. Consequently, a number of research groups, especially in Japan, are constructing humanoid robots. A list of projects is maintained by Willis [28].

Among the most advanced humanoid robots developed so far, is the 58cm tall Sony Qrio [19]. It contains three CPUs and has 38 degrees of freedom (DOF). Qrio is able to walk and dance. Research on map building and navigation, as well as on human-robot interaction is carried out inside Sony. Currently, it is unclear if and when this robot will be available to a larger research community, but the costs of Qrio have been compared to the price of a luxury car.

Unlike Qrio, Hoap-2 (25 DOF, 50cm tall), developed by Fujitsu [7], has been sold to some labs for about USD 50,000. A taller humanoid, Asimo, has been developed by Honda [8]. Its most recent research version has 34 DOFs and a height of 130cm. Approximately the same size of Asimo has a trumpet playing humanoid robot which has been announced recently by Toyota [24].

While the humanoid robots developed by large companies are impressive, they are not available to researchers outside the industry labs or are too expensive for academic research. Some universities built their own robots, but due to limited resources, usually only one prototype has been constructed. Hence, multi-robot experiments with humanoid robots are currently not feasible in academic environments and are likely to be at least difficult in the near future.

Faced with similar problems, researchers working with wheeled robots came up with creative low-cost solutions. One example of a low-cost robot kit is the Lego Mindstorms system. It has been used e.g. for robotic soccer [13], education [29], and communication with people [11]. Other low-cost robotic platforms include the Tetrixx kit [5], the Trikebot [9], and the VolksBot [1].

To avoid the development of custom processing boards, some researchers used off-the-shelf PDAs to control their robots [27, 15]. One of the best know PDA projects is the Palm Pilot Robot Kit [4, 17], developed at CMU. A PDA has also been used to control the Robota and DB humanoid robots [3].

While RoboSapien is certainly the most frequently sold humanoid robot today, it is not the only option for researchers. Kondo developed the servo-driven KHR-1 robot kit [12]. We augmented it with a Pocket PC and a camera as well and use it for gait optimization and to play soccer. Compared to RoboSapien, it is more expensive (≈ EUR 1,000), less robust, and less stable. On the other hand, it has more degrees of freedom (17) than RoboSapien and can move in a more flexible way.

Similar servo-driven robots are offered from Vstone (Robovie-M/MS [26]), Tribotix (Cycloid [25]), and Speecys [20]. Vstone also offers the VisiON robot (Robovie-V) for a price of approximately EUR 7,200. A question for further research would be to find out if the higher number of DOFs of these robots translates to better performance in soccer games. One possible danger could be that walking stability is compromised in these more complex designs.

3 Original RoboSapien

RoboSapien, shown in Fig. 1, is a low-cost humanoid robot, which has been designed by Mark W. Tilden [23] and is marketed with great success by WowWee for the toy market. It measures approximately 34cm in height and its weight is about 2.1kg, including four mono (D) type batteries. These batteries are located in its feet. The low center of mass makes RoboSapien very stable.

The robot is driven by seven small DC motors. One motor per leg moves two joints in the hip and the knee in the sagittal plane, keeping the foot orthogonal to the trunk. A trunk motor tilts the upper body laterally. One motor in each shoulder raises and lowers the arm and one motor in each elbow twists the lower

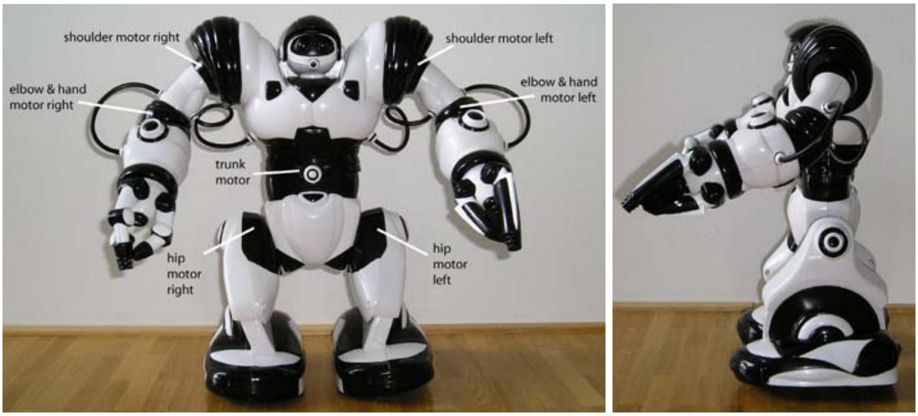


Fig. 1. Robo Sapien. Left: frontal view, seven motors move the robot. Right: side view.

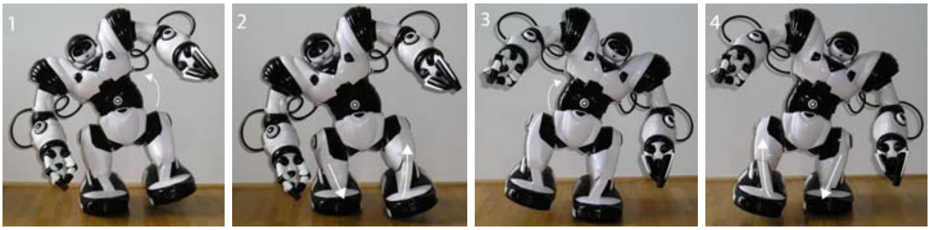


Fig. 2. Dynamic walking gait of RoboSapien. (1) The trunk motor tilts the upper body to the right. The center of mass shifts over the right foot. The left foot lifts from the ground. (2) The leg motors move into opposite directions, resulting in a forward motion of the robot. As the upper body swings back, the left foot regains contact with the ground. (3,4) Symmetrical to (1,2).

arm and opens its grippers. RoboSapien has two gripper hands consisting of three fingers each.

Unlike more complex bipedal robots, RoboSapien uses only three motors for locomotion. This is possible because its gait patterns utilize the dynamics of the robot. For dynamic walking, RoboSapien swings its upper body laterally to achieve a periodic displacement of the center of mass projection from one foot to the other. The resulting walking pattern is illustrated in Fig. 2. The robot moves approximately 4cm per step on a laminate floor. With a step frequency of about 2Hz, this corresponds to a speed of 8cm/s. In the second gait mode the step frequency is increased to about 2.7Hz, but the step length decreases to approximately 2cm. This results in a speed of about 5.2cm/s. RoboSapien walks backwards in a similar way. It can also turn on the spot.

The original RoboSapien is controlled by a human operator who pushes buttons on a remote control. 67 motion commands can be issued to the robot. The motion primitives can be combined, e.g. to have the robot walk a curve.

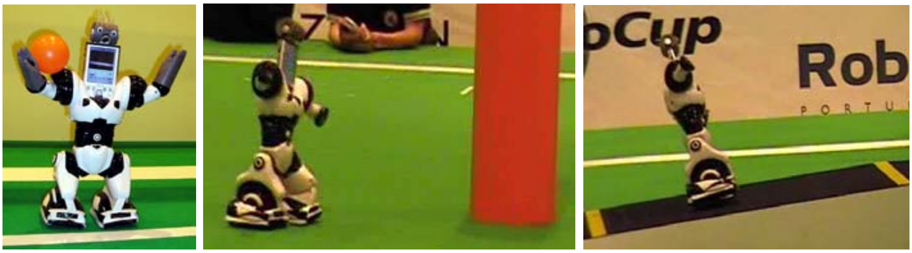


Fig. 3. The augmented RoboSapien competed as NimbRo RS at RoboCup 2004: Humanoid Walk and Balancing Challenge

Preprogrammed motion chains can be triggered by touch sensors, located in its feet and at its finger tips, as well as by a sonic sensor which reacts to clapping sounds.

4 Augmented RoboSapien

In order to make RoboSapien autonomous, we augmented it with computing power and a camera [2]. As NimbRo RS, this augmented RoboSapien took part in some of the RoboCup 2004 Humanoid League competitions (see Fig. 3). It performed the Humanoid Walk and was one of only two robots which mastered the Balancing Challenge, resulting in an overall third place in the Technical Challenges. In order to play soccer, we augmented four more RoboSapiens with an updated Pocket PC and a wide-angle camera as follows (conf. to Fig. 6).

For the Pocket PC, we selected the FSC Pocket Loox 720. It has a weight of only 170g, including the battery, and features a 520MHz XScale processor PXA-272, 128MB RAM, 64MB flash memory, a touch-sensitive display with VGA resolution, Bluetooth, wireless LAN, an infrared (IR) interface, and an integrated 1.3 MPixel camera.

In order to place this Pocket PC between the shoulders of the robot, we removed RoboSapien's head (keeping the IR receiver) and cut a rectangular opening into its chest. The Pocket PC can easily be removed to charge the battery and to download programs. Software for it can be conveniently developed on a PC using e.g. Microsoft (Embedded) Visual Studio.

The Pocket PC needs to interface the robot base. We implemented an unidirectional IR interface via a learning remote program (UltraMote) and Windows messages. The Pocket PC can send multiple motion commands per second to the robot base.

Since in RoboCupSoccer key objects, such as the ball and the goals, are color-coded, visual perception provides a rich source of information about the robot's environment. For this reason, we added a miniature color camera to the Pocket PC. From the few available models, we selected the Lifeview FlyCam-CF 1.3M. An SDK is provided by Lifeview that allows user programs to capture uncompressed live images in the RGB color space. The camera supports resolutions

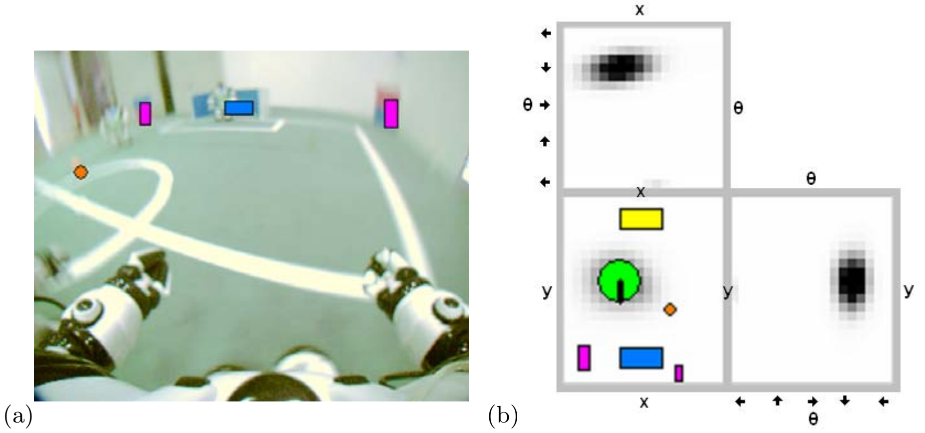


Fig. 4. (a) Image captured from RoboSapien's perspective while it was walking. Detected objects: goal (blue horizontal rectangle), ball (orange circle), and field markers (magenta vertical rectangles); (b) Three two-dimenensional projections of the grid representing the probability distribution of robot poses (x, y, θ). The green circle is drawn at the estimated robot location (x, y). The black line represents its estimated orientation θ. The detected objects are drawn relative to the robot.

from 160×120 up to 1280×1024 pixels. At 320×240 pixels it delivers 5fps. We replaced the original camera lens with a ultra-wide angle lens. The field of view of this camera is now about 150° horizontally × 112° vertically. This allows the augmented RoboSapien to see at the same time its own feet and objects above the horizon (conf. to Fig. 4(a)).

The described modifications make the augmented RoboSapien fully autonomous. The Pocket PC runs computer vision, behavior control, and wireless communication. The total costs for the parts (robot base, Pocket PC, camera, lens, UltraMote) are currently about 700 Euros + tax per robot.

5 Computer Vision and Self Localization

The images captured by the CF camera are the only source of information about the state of the world that our robots use. In order to control their behavior, this data must be analyzed.

Our computer vision software converts the captured RGB images into the YUV color space to decrease the influence of different lighting conditions. The colors of pixels are classified with the pie-slice method [22]. In a multistage process insignificant colored pixels are discarded and the colored objects ball, goals, and field markers are detected. Their coordinates are estimated in an egocentric frame (distance to the robot and angle to its orientation). These estimates are based on image positions and object sizes. The robot-centered coordinates suffice for many relative behaviors, like positioning behind the ball while facing the goal, and dribbling the ball.

To implement global team behaviors, such as kick-off, we need the robot coordinates in an allocentric frame (position on the field and orientation). We estimate these using a probabilistic Markov localization method that integrates egocentric observations and motion commands over time. As proposed by Fox, Burgard, and Thrun [6] this method uses a three-dimensional grid (x, y, θ), shown in Figure 4(b).

We use the localization to compute relative coordinates for the goals if they are currently not visible in the image. Robot localization is also needed for the fusion of local robot views to a global team view. We integrate ball observations from multiple robots using a particle filter to obtain a better estimate of the ball position. The fused ball position is used by the robots that do not see the ball themselves. It is also the basis for the assignment of roles to players.

6 Behavior Control

Since we cannot change the gaits of RoboSapien, behavior control for it needs to focus on higher-level issues, like ball handling, positioning on the field, and team play.

To simplify the behavior control interface to the robot base, we implemented a set of parameterized motion functions, like walking straight for a distance or turning for a certain angle. The motion functions initiate the movement and wait according to the desired distance or the desired turning angle.

Another possibility is to initiate a movement and to keep moving until a desired state change is reported by our computer vision software. This feedback-control is more robust than feed-forward motion macros, but it relies on the visibility of objects.

We try to move the robots in a way that the key objects (ball, goals, and markers) are in the robot's field of view, but this is not always possible. For example, if the robot wants to move around the ball, the robot has to pass the ball first. The robot can only turn towards the ball again when the ball is lying behind the robot, outside its field-of-view. Another example is the situation where the robot faces the goal and wants to move laterally in order to align itself with the ball and the goal. In this case, the robot has to turn (potentially loosing ball sight), to walk towards the goal-ball line, and to turn back. For such cases, we implemented motion macros that chain up to four motion commands. They are triggered when the ball is leaving the robot's field of view. The robot executes the macros and regains ball sight at the end of a macro.

The rate of behavior decisions is limited by the frame rate of the camera and the rate at which RoboSapien accepts new motion commands. Currently, the entire cycle (image capture, computer vision, self localization, behavior control, and motion command) is executed at about 4Hz. This is more than sufficient, compared to the speed of the robots and the ball.

Using the described combination of feed-forward and feedback control, we implemented a number of basic soccer skills. If the robots do not see the ball, they wander on the field to search for it. They can approach the ball, such that they face the goal. They can dribble it towards the goal. If the ball is between a robot and our own goal the robot can move around the ball. We also implemented some defensive behaviors for the goal keeper.

On an external PC, we implemented team behaviors, such as kick-off and normal play. These can assign roles like primary attacker and secondary attacker to the robots. They also can send the robots to an arbitrary position on the field.

7 Infrastructure

In addition to the robots themselves, some infrastructure components are needed to support a team of soccer playing robots.

The most obvious of these are the ball and the field. Our robots play with the small orange plastic ball used in the RoboCup Humanoid League. It has a diameter of 8.4cm and a weight of 26g.

The field size of 3.2m×2.6m also complies to the most recent Humanoid League rules proposal [14]. As playing surface, we use green carpet. The field is marked with 4.8cm wide white lines, as sketched in Fig. 5. In addition to the outer field border, we mark a line separating the two field halves, a center circle of 90cm diameter, and goal areas of size 120cm×40cm. The goals are 80cm wide, 30cm deep, and 30cm high. They are colored in sky-blue and yellow.

We added markers around the field to aid robot localization. The markers are placed at a 50cm distance to the field line. Our first attempt was to use four poles, borrowed from the Aibo-League, placed at the long side of the field, 1m from the half line. It turned out that the 10cm×10cm color patches of these poles were hard to see from a larger distance. For this reason, we switched to six rectangular markers, with color patches of size 29,7cm×21cm (A4 paper). Two of these patches are placed on top of each other. One is always magenta. The other is white for markers placed in the middle of the long field side. It is yellow or sky-blue for markers placed at the corners. Markers on the left side of the field (when facing the yellow goal) have magenta as upper color. For the right field side, magenta is below the other color.

The Pocket PCs used in the augmented RoboSapiens are equipped with wireless network adapters. We use the wireless UDP communication to transmit debug information to an external computer where it is logged and visualized. This computer is also used to fuse local views to a team view and executes team behaviors.

In order to be able to design behaviors without access to the real hardware, we implemented a physics-based simulation for two teams of RoboSapiens. This simulation is based on the Open Dynamics Engine [18].

8 Experimental Results

In addition to the computer vision, self localization, and behavior control components described above, an alternative set of these components has been developed in a lab course by a group of six students.

Both systems are able to play soccer. To evaluate them, we played test games in our lab with an increasing number of players. A similar lab project was done by the Brainstormers at University of Osnabrück. In April 2005, both teams met at the German Open in Paderborn to show three demonstration games. The games lasted 2×10min each and attracted many spectators. During these games, the robots knew most of the time where the ball was. The scored goals were not accidental, but the results of intentional actions. The robots were playing without human help. Only the referee was allowed to touch the robots in order to untangle them in the case of entanglements. Whenever the ball went outside the field, the referee would put it back to the field line at the position where it left the field. We also learned that the presence of more than two robots in the goal box (one attacker and the goalie) must be prevented by the rules to avoid overcrowding.

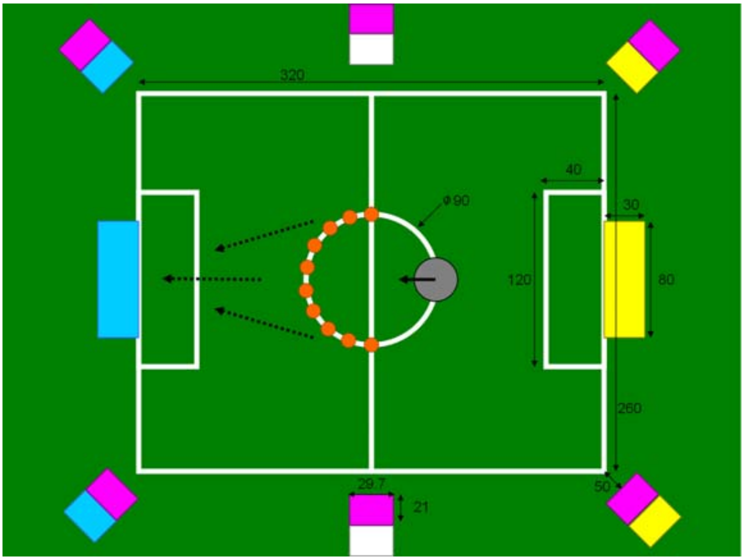


Fig. 5. Scoring Test. Sketch of the field setup, robot and ball positions used.

In order to produce a performance estimate that is less noisy and easier to obtain than the score of entire soccer games, we designed a scoring test, illustrated in Fig. 5. In this test, one robot stands on the most distant point of the center circle, facing the empty goal, which is 2.05m away. The ball is placed at ten different positions on the half of the center circle which is closer to the goal (20° steps). The chosen ball position is not communicated to the robot. Its task is now to bring the ball into the goal as quickly as possible.

Our system managed to score in all ten trials. The augmented RoboSapien needed on average 170s to score. The robot was operating continuously during the scoring test. The only human intervention was to place the ball and the robot at their initial positions.

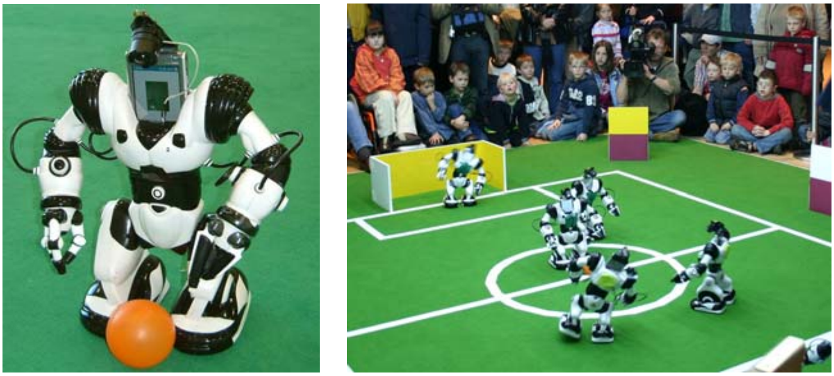


Fig. 6. Augmented RoboSapiens playing soccer at RoboCup German Open 2005

For comparison, a human-controlled (via the original remote control unit) augmented RoboSapien (perfect perception, almost perfect behavior control) took on average 97s for scoring. It had a 100% success rate as well.

9 Conclusions

In this paper, we described a way to augment low-cost commercial off-the-shelf humanoid robots in order to convert them into a soccer team.

For programmable autonomy, we attached Pocket PCs to the RoboSapiens. They provide ample computing power and have many interfaces. To allow for visual perception of the game situation, we added a color CMOS camera.

We implemented computer vision, probabilistic self-localization, and behavior control on the Pocket PC. In addition, we set up a soccer field, wireless communication, and a physics-based simulation.

This system was able to play test games in our lab and at German Open 2005. The augmented RoboSapien also performed well in a scoring test. The soccer experiments revealed some limitations of the augmented RoboSapien. They include low precision, unidirectional IR communication, and mechanical limitations. The low precision of walking makes it unfeasible to rely on path integration for navigation. It is necessary to compensate for the quickly accumulating deviations by visual feedback. The unidirectional IR communication from the Pocket PC to the robot base prevents the use of proprioceptive information, touch sensors, and sonic sensors for behavior control. The low number of DOFs as well as the low center of mass limit the possible movements. For instance, while it is possible to dribble a ball with the robot, RoboSapien is unable to perform the powerful kick needed for penalties. It is also unable to walk laterally, but must turn towards the target before walking.

Despite these limitations, we think that augmented RoboSapiens are a suitable platform for performing multi-robot experiments with humanoid robots. When working with such a platform, one does not need to deal with the difficulties of bipedal walking and balance. Since these issues are solved by the RoboSapien base, the researchers can focus on visual perception and higher-level behavior control.

As can be seen in the RoboCup Four-legged (Aibo) League, the use of standardized hardware has certain advantages and disadvantages. On the positive side, there is no need to develop and build robots for researchers interested in perception and behavior control. One can start with an off-the-shelf robot to develop software. Standardized hardware also facilitates the exchange of software components and the comparison of experimental results between research groups.

On the other hand, commercial robots are usually not fully open. The developer has to use the API provided by the manufacturer and cannot modify the software and hardware layers below the API. These changes are done exclusively by the manufacturer, which might limit the exploration of new ideas by researchers. While it is in many cases possible to find a work around a limitation, this approach might lead to the use of the hardware in a way not intended by the manufacturer. One example for this is the walking on the knees (instead of the paws) adopted by most of the participants of the RoboCup Four-legged League.

For the reasons above, we think that the availability of capable standard hardware would facilitate empirical multi-agent research on humanoid robots. If such robots were open and well documented, they could be used as a starting point for researchers. One step towards this goal was to augment RoboSapien.

Since the costs for this programmable autonomous humanoid robot are only about EUR 700, it is feasible to perform experiments with more than one robot even for research groups lacking huge resources. This could be interesting not only for university groups and industry labs, but also for RoboCup Junior, education, and enthusiasts.

We made the API for sending motion commands to the robot base and capturing images, as well as a tutorial how to augment RoboSapien publicly available. Other research groups adopted the augmented RoboSapien already. We delivered one robot to Microsoft Research Cambridge. The Brainstormers of University of Osnabrück also augmented a number of RoboSapiens. Together, we played three soccer demonstration games at the RoboCup German Open (April 2005) in Paderborn. At RoboCup 2005, the team Hiro used augmented RoboSapiens in the Humanoid League competitions.

Videos of RoboSapiens playing soccer and more images can be found on our website: http://www.NimbRo.net [/rs].

Acknowledgements

This project is funded by grant BE 2556/2-1 of Deutsche Forschungsgemeinschaft (German Research Foundation, DFG). We would like to thank Mark W. Tilden and WowWee for providing two RoboSapien samples.

Jörg Stückler helped us with the simulator. Tobias Langner worked on the RoboSapien API. Dr. Maren Bennewitz provided advice on probabilistic localization. Andreas Maunz, Kai M. Wurm, Jens Singler, Markus Ihmsen, Marc Gißler, and Jan Kurrus developed the alternative system in a lab project.

References

1. Fraunhofer AIS. VolksBot. http://www.ais.fraunhofer.de/be/volksbot/.
2. Sven Behnke, Tobias Langner, Jürgen Müller, Holger Neub, and Michael Schreiber. NimbRo RS: A low-cost autonomous humanoid robot for multi-agent research. In *Proc. of WS on Methods and Technology for Empirical Evaluation of Multi-Agent Systems and Multi-robot Teams (MTEE) at KI2004, Ulm, Germany*, 2004.
3. Sylvain Calinon and Aude Billard. PDA interface for humanoid robots. In *Third IEEE International Conference on Humanoid Robots (Humanoids 2003)*, 2003.
4. CMU. Palm Pilot Robot Kit. http://www-2.cs.cmu.edu/∼reshko/pilot/.
5. Stefan Enderle, Stefan Sablatnög, Steffen Simon, and Gerhard K. Kraetzschmar. Tetrixx – A Robot Development Kit. In *Proc. of Edutainment Robots WS*, 2000.
6. Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.
7. Fujitsu. HOAP-2. http://www.automation.fujitsu.com/en/products/products09.html.
8. Honda. ASIMO. http://world.honda.com/asimo/.
9. Thomas Hsiu, Steve Richards, Ajinkya Bhave, Andres Perez-Bergquist, and Illah Nourbakhsh. Designing a low-cost, expressive educational robot. In *Proc. of Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas*, 2003.
10. iXs Research Corp. http://www.ixs.co.jp.
11. Alexander Koller and Geert-Jan Kruijff. Talking robots with LEGO mindstorms. In *Proc. of 20th Int. Conf. on Computational Linguistics (COLING)*, Geneva, 2004.
12. Kondo Kagaku Co., Ltd. KHR-1. http://www.kondo-robot.com.
13. Henrik Hautop Lund and Luigi Pagliarini. RoboCup Jr. with LEGO Mindstorms. In *Proc. of Int. Conf. on Robotics and Automation, San Francisco, CA*, 2000.
14. Norbert M. Mayer. Humanoid Kid Size League and Medium Size League rules and setup. http://er04.ams.eng.osaka-u.ac.jp/humanoid_webpage/humanoid.pdf.
15. Kevin Mukhar, Dave Johnson, Kevin Mukhar, and Dave Johnson. *The Ultimate Palm Robot*. McGraw-Hill, 2003.
16. Masaki Ogino, Masaaki Kikuchi, Junichiro Ooga, Masahiro Aono, and Minoru Asada. Optic flow based skill learning for a humanoid to trap, approach to, and pass a ball. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 323–334, 2005.
17. Greg Reshko, Matthew Mason, and Illah Nourbakhsh. Rapid prototyping of small robots. Technical Report CMU-RI-TR-02-11, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2002.
18. Russel Smith. Open Dynamics Engine. http://opende.sourceforge.net.
19. Sony. Dream Robot QRIO. http://www.sony.net/qrio.
20. Speecys Corp. Speecys robot kit. http://www.speecys.com.
21. Team Osaka. VisiON. http://www.sansokan.jp/robot/info/vision_en.html.
22. Peter J. Thomas, Russel J. Stonier, and Peter J. Wolfs. Robustness of colour detection for robot soccer. In *Proc. of 7th Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, volume 3, pages 1245–1249, 2002.
23. Mark W. Tilden. Neuromorphic robot humanoid to step into the market. *The Neuromorphic Engineer*, 1(1):12, 2004.
24. Toyota. Partner Robot. http://www.toyota.co.jp/en/special/robot/.
25. Tribotix Pty Ltd. Cycloid. http://www.tribotix.com/products/robotis/cycloid.htm.
26. Vstone Co., Ltd. http://www.vstone.co.jp.
27. Doug Williams. *PDA Robotics*. McGraw-Hill, 2003.
28. Chris Willis. World's greatest android projects. http://www.androidworld.com.

29. Xudong Yu and Jerry B. Weinberg. Robotics in education: New platforms and environments. *IEEE Robotics & Automation Magazine*, 10(3), 2003.
30. Changjiu Zhou and Pik Kong Yue. Robo-Erectus: a low-cost autonomous humanoid soccer robot. *Advanced Robotics*, 18(7):717–720, 2004.

Reliable and Precise Gait Modeling for a Quadruped Robot

Uwe Düffert and Jan Hoffmann

Institut für Informatik
LFG Künstliche Intelligenz
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany
http://www.aiboteamhumboldt.com

Abstract. We present a parametric walk model for a four-legged robot. The walk model is improved using a genetic algorithm, but unlike previous approaches, the fitness is determined in a run that closely resembles the later application. We thus not only achieve high speeds, but also a high degree of flexibility. In addition to the walking model being flexible, we present a means of automatically calibrating the walking engine. This allows for highly precise robot control and greatly improved odometry accuracy. Lastly, we show how the motion model can be extended to integrate specialized motions to further increase locomotion speed without compromising flexibility.

1 Introduction

Legged robots operate in areas that wheeled robots have trouble accessing. On the other hand, creation and optimization of quadruped robot locomotion is a challenging, highly complex task. The main reason for this is the difficulty to cope with the many degrees of freedom of a legged robot even in a relatively simple robot design. Such designs (may it be software and/or hardware) are often inspired by animal locomotion in terms of anatomy (number of limbs and joints, proportions, etc.), gait patterns (Central Pattern Generators) and concepts such as reflexes [5, 1, 13, 16].

Inverse kinematics is commonly used to calculate the motor commands necessary for robot motions: trajectories of the robot's paws are defined and then the corresponding limb movements and thus joint angles as a functions of time are calculated. Inverse kinematics is usually based purely on geometry, neglecting physical properties such as friction, weight of the robot as a whole and of individual components, moments of inertia, motor strengths, etc. These simplifications in the modeling tend to impair performance in real world environments.

Experience can help to come up with experimental setups which take these pitfalls into consideration while not explicitly modeling them, e. g. by conducting experiments on especially difficult surfaces [11]. Limiting possible motions to statically stable ones, robust robot gaits can be developed. These motions

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 49–58, 2006.

are reversible at any given point in time and require three of the robot's four legs to touch the ground at any time [6]. These motions trade off speed for robustness whereas dynamically stable motions can produce faster locomotion [15, 4, 8]. Gaits can be found by trying to model all physical aspects of the robot, but bridging the gap between simulation and the real world remains challenging [7, 3]. An alternative gait representation in frequency space is described in [10] yielding smooth motions and gait transitions.

Evolutionary approaches include the evolution of the controller alone (with fixed robot morphology [11, 17]) and simultaneous evolution of robot morphology and controller [19, 18]. Other approaches include teaching, learning, and inverse kinematics [12]. Many machine learning approaches focus on optimizing a single criterion such as forward or turning speed which results in highly specialized motions. From these specialized motions it tends to be difficult to generalize towards gait patterns that can handle 'mixed' motions, e. g. moving forward and sideways at the same time. These mixed motions are of great importance in real robot control in dynamic environments where it is desirable to be able to adjust the robot's position and orientation quickly ('omnidirectional' movements).

Outline. This paper describes a combination of different strategies to improve the overall performance of four-legged walking. We present a parametric walk model extending [9]. Using this 'wheel model' increases the robustness of walk of the legged robot. A means of automatically optimizing the gait pattern is presented, focusing on flexibility rather then speed alone. We then show how the gait pattern can be calibrated to achieve reproducible performance and odometry data of high quality. Lastly, we show how the walking engine can be extended to integrate specialized motions without sacrificing flexibility.

2 Method

2.1 The Wheel Model

Legged robots can access areas which wheeled robot are unable to cope with. Unfortunately, robot control for a legged robot is extremely complex due to the many degrees of freedom involved. For the four-legged Sony Aibo ERS-210, [9] introduced the *wheel model* which assumes that the robots paws are performing circular motions and robot control works much like that of a differential drive robot (fig. 1). Using this model, dynamic stability of the robot can be achieved easily by constraining the phase shift between the movement of individual legs. In our experiments, we used both Aibo ERS-210s and ERS-7s.

Each movement (consisting of a forward, a sideways, and a turning speed) results in a circular movement around a common center of rotation (see fig. 1 a). This movement is realized by making steps tangential to the circle with the speed a wheel would have at the same position.

Actual gait patterns are described by a set of parameters of the walking engine. Several approaches to parameterize these patterns where developed in the context of gait optimization allowing the trajectories to differ from the first used trapezoidal shape (e. g. *canter action* in [9] and *free form quad* in [2]).

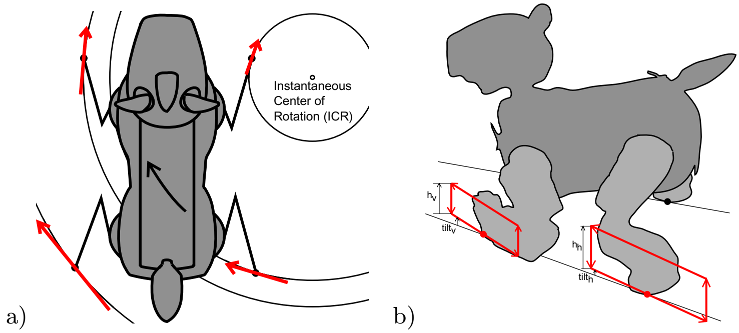


Fig. 1. a) Simultaneous walking and turning using the wheel model: The resting positions of the feet move on circles around a common rotation center. This is realized by steps (red) that are deduced from imagined wheel movements and tangential to the optimal circles. **b)** Parameters of the foot movements: Feet are moved in parallelograms (rhomboids), the direction and size of which is determined by the wheel model.

We chose the parameter set shown in table 1 which allowed us to fully describe the characteristics of different gait patterns while still being reasonably small for genetic optimization.

Table 1. All used parameters of a parameter set and their meaning

- x_f, y_f, z_f: position of a front foot relative to middle between the front shoulders
- x_h, y_h, z_h: position of a hind foot relative to middle between the hind shoulders
- h_f, h_h: maximum height of the feet above ground during a step
- $tilt_f, tilt_h$: tangent of the the arc between the theoretical foot trajectory and the ground; influences the intensity of touching the ground and can avoid sliding on it
- gp_v, gp_h: fraction of time a front or hind foot has ground contact (in theory)
- $l = (l_{fl},\ l_{fr},\ l_{hl},\ l_{hr})$: relative time of lifting each leg, (0, 0.5, 0.5, 0) describes the usually used trot and means lifting left front and right hind foot at the beginning of a full step (0) and lifting the other two feet half a full step later (0.5)
- T: duration of a full step in frames à 8 milliseconds.

2.2 Localization

One of the goals of our work was to allow fully automatic gait optimization. We wanted to be independent of external hardware (such as an external camera and computer to track the robot) and wanted everything to run on the robot. This requires the robot to be well localized in order for it to determine its current speed and performance. The standard beacons on a RoboCup field were used in previous work, but they have two disadvantages: 1) if only one beacon is visible, the robot has no way of determining its lateral position, and 2) tracking multiple beacons is error prone and the localization error is often bigger than the covered distance. We therefore devised a special bar code like pattern (see fig. 2) that allows the robot to precisely monitor its x-y-position and orientation from a single camera image from a wide range of positions. The black and white

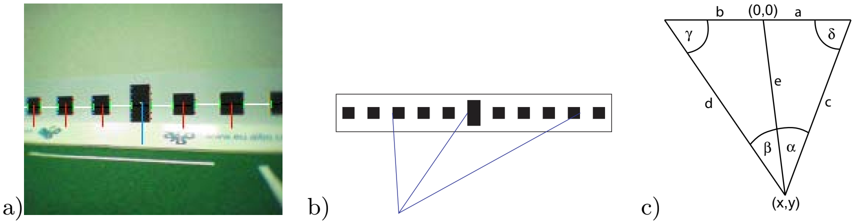


Fig. 2. The b/w pattern used for localization: **a)** A camera image of an ERS-210 with detected pattern parts highlighted, **b)** Schematic view of the pattern, **c)** Variables used to calculate the robot pose

pattern has the further advantage of being clearly detectable in various lighting conditions. Size and structure of the pattern are chosen such that equally good localization performance can be achieved from the majority of distances to it.

As shown in fig. 2 c, the current position can be calculated from the known distances a and b and the recognized angles α and β.

The angle δ can be calculated by applying the sine theorem twice. This yields the relative position (x, y) w.r.t. the center of the pattern:

$$\begin{aligned} x &= -c \sin(\delta) \\ y &= -a + c \cos(\delta) \end{aligned}$$

with $c = (a \sin(\pi - \alpha - \delta))/\sin(\alpha))$. For a distances from the pattern of 60 cm to 260 cm, the position error from single images during walking is $\Delta p = (16\,\text{mm}, 38\,mm)$. The noise caused by camera vibrations can be reduced by using simple PID smoothing. This decreases the average position error and allows meaningful speed calculations even for short durations.

2.3 Evolution Run

Most previous optimization approaches have in common that they are focused on improving a particular motion, such as walking forward at high speed. They fail, however, to take into account the overall performance of the robot in real world situations, where constant adjustments of the robot's direction and orientation are necessary.

To evaluate a parameter set, the robot's performance following a path is determined. Unlike other experiments, this is not a simple straight line, but a rather complex path that requires the robot to strafe and turn too to follow it (see fig. 3). Deviations from the path are penalized by the fitness function (see below). Such a course is a sequence of target positions and robot orientations. An evaluation run consists of two parts, a forward part with desired robot orientation changing over time as shown in fig. 3, and a backward part with constant robot orientation. Instead of executing a fixed sequence of steps,

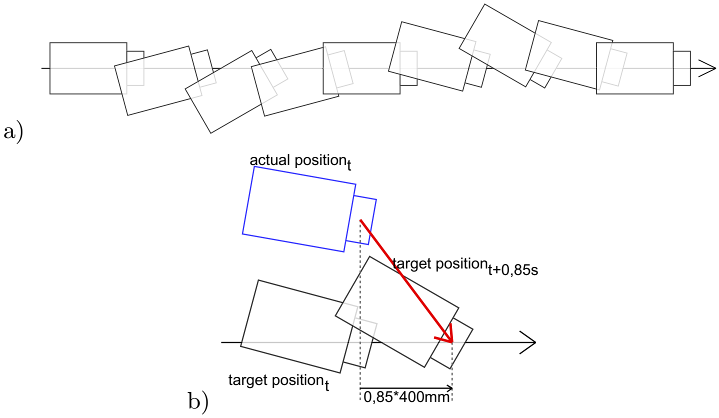


Fig. 3. a) The path for evaluation omnidirectional gait pattern defines the target position and orientation of a robot. **b)** The activated movement (red) results from the distance to the target position 0.85 seconds later.

the robot is forced to correct its previous mistakes to stay on the course. Such a task was chosen because it is easily reproducible and quite typical for actual applications where the robot constantly changes the direction while walking forward (e. g. chasing a ball). The control algorithm is very similar to the one used by us in RoboCup games to steer the robot towards the ball or some other target position.

The runs are performed fully autonomous. No manual replacement of the robot is necessary since it is generally able to localize using the pattern described above. The start and end point of the path are fixed position relative to the localization pattern.

The obvious performance criteria for a gait are speed and accuracy. The fitness function F of a parameter set P used in the experiments favors more stable gaits and can be interpreted as walk speed corrected by unwanted vibrations and position deviations:

$$F(P) = \dot{x} - \Delta y/6 - 33\Delta\varphi - \left(10^{-5}\ddot{z} - 5\right) - 40p_{\text{blind}}$$

where $\dot{x}$ is the average speed, Δy and $\Delta\varphi$ the deviation from the course, $\ddot{z}$ describes vertical vibrations and p_{blind} is the percentage of images without recognition of the localization pattern, e. g. because of vibrations or totally wrong gaze direction. As walking forward is more important, the fitness of the forward walking part contributes more than that of walking backward.

Evaluating a single parameter set takes about 30 seconds: 10s for walking forward with turning, 10s for walking backward and 10s for positioning for the next run. If the performance of a parameter set is below a threshold, the run is aborted and the robot returns to the starting point using the standard gait. Typically, a complete optimization run with 50 to 60 parameter sets takes about 30 minutes.

2.4 Evolution

Since it is very difficult to model the interdependencies between parameters and the resulting speed and quality of a gait pattern, genetic algorithms were used for optimization (see [14]). A population of parameter sets is exposed to evolution. Each parameter set corresponds to an individual and each parameter to a gene. All genes of an individual are stored on a single chromosome.

Using real robots for the evolution is time consuming, therefore choosing the following evolution parameters turned out to be a good compromise between fast advance and avoidance of unusable parameter sets.

A population consists of only ten individuals, starting with a known parameter set (a manually tweaked one used by our team in previous years) and nine mutations of it. In every generation, half of the population with the worst fitness is selected and replaced by mutations and recombination of the better half. 40% of the descendants are created by mutation and 60% by recombination. Mutation changes single genes with a probability of 30%, equally distributed up to ±6% of its original value. Recombination interpolates the value of each gene randomly between the parent values of that gene or even extrapolates into the direction of the better parent.

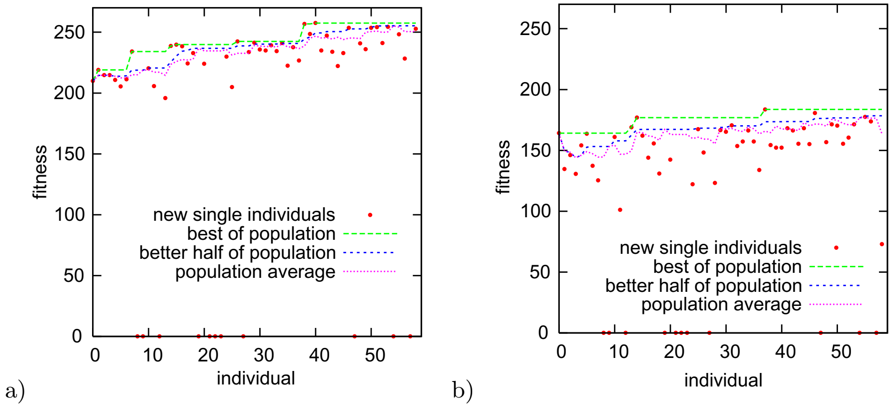


Fig. 4. ERS-7: Evolution of walking parameter sets with two separated populations for the forward and the backward part of the course, **a)** Development of the fitness in the population of forward walking parameters, **b)** Development of the fitness in the population of backward walking parameters

Using these values results in visible and measurable differences between single individuals without getting unusable parameters. This method is useful for local optimization in a sensible part of the search space without prior knowledge about correlation between parameters and with only a few abortive attempts (fig. 4).

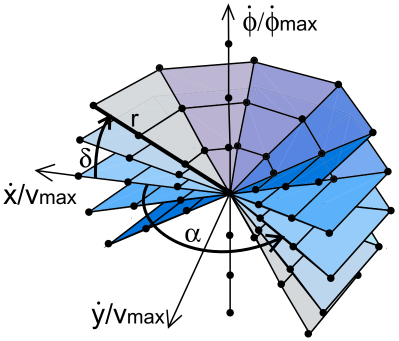
2.5 Odometry Calibration

Experience shows that the executed motion does not always match the one intended by the calculated step sizes. The actual speed depends non-linearly on the target speed.

A walk request commonly consists of forward, sideways, and turn speed. It turned out, however, that for our application this is not a very good way of describing motions. We therefore devised a different means of describing a walk request consisting of the walk direction α, the 'turn-walk-ratio' δ, and the 'overall speed' r as defined in fig. 5. To approximate the non-linear dependency of target and actual speed, the target speed for each combination of walk direction and turn-walk-ratio is divided up into three ranges [0, small[, [small, med[, and [med, max]. Within each of these ranges, the dependency is assumed linear. The values for the boundaries are determined in the calibration process. The calibration is done for all combinations of forward, sideways, and turn speed which results in 127 boundary points to be calibrated. Luckily, this can be done automatically:

In a first run, an autonomous behavior measures the influence of increasing the step size on the walk speed for each combination of walk direction and turn-walk-ratio for a certain parameter set. If increasing the step sizes does not increase the overall speed any more, the behavior starts to measure the next walk direction. This first calibration run determines the respective minimum and maximum speeds and enables to chose a medium speed minimizing the deviations when using linear interpolation in between.

In a second calibration pass, all chosen 127 boundaries are adjusted independently to match their requested forward, sideways, and turn speed by changing the target step size proportional to half of the detected speed difference. These adjustments minimize the gap between target and actual motion without risking to alternate only the sign of the difference. Iteratively running the calibration further decreases the error, as shown in fig. 5 b.



walk direction $\alpha = \arctan(\dot{x}, \dot{y})$

$\rightarrow \left[-\pi, -\frac{3\pi}{4}, -\frac{\pi}{2}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right]$

turn-walk-ratio $\delta = \frac{2}{\pi} \arctan\left(\frac{v}{v_{max}}, \frac{\dot{\varphi}}{\dot{\varphi}_{max}}\right)$

$\rightarrow \left[\underbrace{-1}_{\text{turn right}}, -\frac{3}{10}, -\frac{1}{10}, 0, \frac{1}{10}, \frac{3}{10}, \underbrace{1}_{\text{turn left}}\right]$

overall speed $r = \sqrt{\left(\frac{v}{v_{max}}\right)^2 + \left(\frac{\dot{\varphi}}{\dot{\varphi}_{max}}\right)^2}$

$\rightarrow$ [slow, medium, fast] .

Fig. 5. The position of the 127 parameter sets used (black dots: standing, 6× turning only, 8×5×3× with walking): The azimuth α denotes the walk direction, the declination δ denotes the normalized turn-walk-ratio and the radius r the normalized overall speed

Calibration is done for constant walk requests only. A constant walking motion always results in an arc. Knowing time, position and orientation of the robot at a starting and an ending point, the average walking and turning speed inbetween can be calculated.

This calibration process is repeated when the robot has to operate on a new surface.

2.6 Using Multiple Parameter Sets

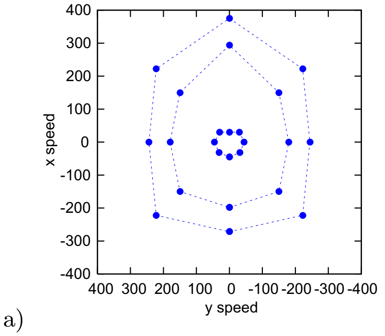
Once a good omnidirectional gait pattern was found, we explored ways of including specialized gaits to further increase performance. This brings about the problem of switching or interpolating between gait patterns. That problem can be solved by allowing different parameter sets for the 127 boundary points described in the previous section. For interpolation to work, neighboring parameter sets must not differ *too much*.

If e. g. a particularly good (and similar enough) parameter set was found for turning, it can be used to replace the parameter set(s) associated with the boundary points (walk direction = 0, turn-walk-ratio = ±1, overall speed = $v_{\max}$).

Whether a parameter set is suitable for interpolation is evaluated manually. For example, we extended the walking engine by including a parameter set for fast turning. This was derived manually from the parameter set for the omnidirectional walk by decreasing the distance between front and hind feet.

2.7 Performance

The walking engine consisting of the omnidirectional parameter set and specialized motions for ‘walking backwards in a straight line’ and ‘fast turning (only)’



b)

	$\vert\Delta\dot{x}\vert$ in mm/s	$\vert\Delta\dot{y}\vert$ in mm/s	$\vert\Delta\dot{\varphi}\vert$ in rad/s
uncalibrated	12,9	12,1	0,086
measured	7,7	11,4	0,070
calibration 1	6,4	7,6	0,038
calibration 2	5,8	8,0	0,030
calibration 3	4,4	8,7	0,021
2 weeks later	7,9	7,5	0,046
different robot	6,9	7,8	0,039

Fig. 6. a) The used minimum, medium, and maximum speed for an ERS-7 in all 8 walk directions without turning: By using several optimized and calibrated parameter sets, a much higher speed range can be covered than by using a single parameter set and the (necessary) speed limitation e. g. to an ellipse. **b)** The average difference between assumed and real speed of the 127 parameter sets can be decreased significantly with a few calibration steps.

proved to deliver highly reliable and reproducible performance. The speed ranks amongst the highest that have been achieved on the Aibo ERS-7 to date (fig. 6 a). The precision was evaluated qualitatively: the robot was able to stay on a rectangular path only using odometry for localization. In contrast, using uncalibrated motions, the robot would turn farther than desired at every corner resulting in a triangular trajectory.

3 Conclusion

Using the presented experimental setup, we were able to perform automated evolutionary optimization of gait patterns on a legged robot. Unlike most other approaches, the gait pattern found performs well in actual applications where target speed and direction of the robot continuously change. This was achieved by having the robot follow a path that closely resembles a real life situation for evaluation. The gaits found are calibrated to allow for remarkably accurate odometry which greatly improves localization. Lastly, the walking engine was extended to allow interpolation from one parameter set to another. Using this approach, we were able to use highly optimized/specialized motions in combination with the general, highly accurate gait pattern found in the evolution. The walking engine was successfully used in the RoboCup 2004 world championships Sony Four-Legged League. Its outstanding performance and precision was one of the key advantages of the GermanTeam over other teams and helped to win the championship.

Acknowledgments

The program code used was integrated into the development of the GermanTeam, a joint effort of the Humboldt University of Berlin, University of Bremen, University of Dortmund, and the Technical University of Darmstadt. Source code is available for download at `http://www.germanteam.org`.

References

1. A. Billard and A. J. Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. 2000.
2. J. Chen, E. Chung, R. Edwards, N. Wong, E. Mak, R. Sheh, M. S. Kim, A. Tang, N. Sutanto, B. Hengst, C. Sammut, and W. Uther. runswift 2003. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004.
3. X. Chen, K. Watanabe, K. Kiguchi, and K. Izumi. Optimal force distribution for the legs of a quadruped robot. *Machine Intelligence and Robotic Control*, 1(2):87–93, 1999.
4. U. Düffert, M. Jüngel, T. Laue, M. Lötzsch, M. Risler, and T. Röfer. GermanTeam 2002. In *RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.)*, number 2752 in Lecture Notes in Artificial Intelligence. Springer, 2003. More detailed in http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf.

5. J. Duysens, H. V. de Crommert, B. Smits-Engelsman, and F. V. der Helm. A walking robot called human: lessons to be learned from neural control of locomotion. *Journal of Biomechanics*, 2000.
6. M. Fujita, S. Zrehen, and H. Kitano. A quadruped robot for RoboCup legged robot challenge. In *Proceedings of the second RoboCup Workshop*. Springer, 1998.
7. M. Hardt and O. von Stryk. The role of motion dynamics in the design, control and stability of bipedal and quadrupedal robots.
8. B. Hengst, D. Ibbotson, S. B. Pham, J. Dalgliesh, M. Lawther, P. Preston, and C. Sammut. The UNSW RoboCup 2000 Sony Legged League team. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 70–72 (64–75). Springer, 2001.
9. B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut. Omnidirectional locomotion for quadruped robots. In *RoboCup 2001 Robot Soccer World Cup V, A. Birk, S. Coradeschi, S. Tadokoro (Eds.)*, number 2377 in Lecture Notes in Computer Science, pages 368–373. Springer, 2002.
10. J. Hoffmann and U. Düffert. Frequency Space Representation and Transitions of Quadruped Robot Gaits. In *Proceedings of the 27th conference on Australasian computer science*, volume 26, pages 275 – 278. Australian Computer Science Society, Inc., 2004.
11. G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with Aibo. *IEEE International Conference on Robotics and Automation*, pages 3040–3045, 2000.
12. V. Hugel and P. Blazevic. Towards efficient implementation of quadruped gaits with duty factor of 0.75. In *Proceedings of the IEEE International Conference On Robotics and Automation*, 1999.
13. H. Kimura, Y. Fukuoka, Y. Hada, and K. Takase. 3d adaptive dynamic walking of a quadruped robot by using neural system model. 2001.
14. J. R. Koza. *Genetic Programming.* MIT Press, Cambridge, MA, 1992.
15. S. Lenser, J. Bruce, and M. Veloso. CMPack: A Complete Software System for Autonomous Legged Soccer Robots. 2001.
16. M. A. Lewis. Gait adaptation in a quadruped robot. *Autonomous Robots*, 12(3):301–312, 2002.
17. S. Nolfi and D. Floreano. Learing and evolution. *Autonomous Robots*, 7(1):89–113, 1998.
18. S. Nolfi and D. Floreano. *Evolutionary Robotics.* MIT Press, 2000.
19. K. Sims. Evolving 3D morphology and behavior by competition. In *Proceedings in Artificial Life IV, R. Brooks and P. Maes (editors)*, pages 28–39. MIT Press, 1994.

Toni: A Soccer Playing Humanoid Robot

Sven Behnke, Jürgen Müller, and Michael Schreiber

Albert-Ludwigs-University of Freiburg, Computer Science Institute
Georges-Köhler-Allee 52, 79110 Freiburg, Germany
{behnke, jmuller, schreibe}@informatik.uni-freiburg.de

Abstract. This paper describes the humanoid robot Toni that has been designed to play soccer in the RoboCup Humanoid League. The paper details Toni's mechanical and electrical design, perception, self localization, behavior control, and infrastructure.

Toni is fully autonomous, has a low weight (2.2kg), and is much taller (74cm) than most servo-driven humanoid robots. It has a wide field of view camera, ample computing power, and wireless communication.

Toni possesses basic soccer skills. It walks dynamically in all directions (up to 20cm/s in forward direction) and turns on the spot. It perceives the ball and the goals and localizes itself on the field. Toni is able to approach the ball and to dribble it. It can kick the ball without falling.

We performed tests in our lab and penalty kick demonstrations at RoboCup German Open 2005. Toni's successors Jupp, Sepp, and Max performed well at the RoboCup 2005 Humanoid League competitions.

1 Introduction

The ultimate goal of the RoboCup initiative is stated as follows: By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, comply with the official rule of the FIFA, against the winner of the most recent World Cup [9]. As one step towards this long-term goal, the RoboCup Federation added in 2002 a league for humanoid robots to their annual soccer championships.

Three competitions took place in the RoboCup Humanoid League so far. In preparation for real soccer games, the robots had to demonstrate their capabilities by solving a number of subtasks.

In the Humanoid Walk, they had to walk towards a pole, to turn around it, and to come back to the start. Scoring was based on walking speed and stability. In the Penalty Kick competition, two robots faced each other. While one robot tried to score a goal, the other defended. In the Freestyle competition, the robots had five minutes to show a performance to a jury. Each year, there is also a new technical challenge. In 2004, it consisted of an obstacle walk, a passing task, and balancing across a sloped ramp.

The RoboCup Humanoid League competition rules [11] require robots to have a human-like body plan. They must consist of a trunk, two legs, two arms, and a head. The only allowed mode of locomotion is bipedal walking. The robots must

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 59–70, 2006.

be fully autonomous. No external power, computing power, or remote control is allowed.

In the 2002 Humanoid League competition, the Nagara robot was the overall winner. A Honda Asimo prototype of team HITS Firstep [6] won in 2003. Team Osaka won the 2004 competition with the robot VisiON [17]. This robot used an omnidirectional camera as head. As a goalie it could defend against a shot by jumping to the ground. Afterwards, VisiON got up without help. Another highlight of the competition was the passing demonstration between two Hoap-2 robots of team Senchans A [14].

Despite these impressive achievements, the overall performance of the soccer playing humanoids is still far from perfect. Basic soccer skills, such as robust dynamic walking and kicking without loosing balance are not possessed by all robots. Even the best robots sometimes show instability while walking, fail to kick the ball, or defend against shots not taken. Consequently, further research is needed. Within the Humanoid League, the performance of smaller, servo-driven robots in general exceeds the performance of larger robots. The only convincing larger robot so far was the Honda Asimo prototype, out of reach for almost all researchers.

In the following, we describe the humanoid robot Toni, which has been designed by our team NimbRo for the 2005 RoboCup Humanoid League soccer competitions. We use servo motors to drive its 18 joints for their relatively low cost and for their good weight-to-torque ratio. Our design focused on weight reduction to make Toni agile. We used standard components for computing power and camera because of their high degree of integration and their relatively low cost.

This paper is organized as follows. In the next section, we review some of the related work. Section 3 describes Toni's mechanical and Section 4 its electrical design in detail. In Section 5 proprioception, computer vision, and self localization are covered. Section 6 describes how Toni is controlled using a hierarchy of reactive behaviors, and Section 7 details its infrastructure components.

2 Related Work

Humanoid robots are not only a good choice for playing soccer. The anthropomorphic body shape is also helpful for acting in environments that have been designed for humans, in particular for the interaction with people. In addition to speech, a humanoid robot can try to use the same means for intuitive multimodal communication that people use: body language, gestures, mimics, and gaze. Consequently, a number of research groups, especially in Japan, are constructing humanoid robots. A list of projects is maintained by Willis [22].

Among the most advanced humanoid robots developed so far is the 58cm tall Sony Qrio [16]. It contains three CPUs and has 38 degrees of freedom (DOF). Qrio is able to walk and dance. Research on map building and navigation, as well as on human-robot interaction is carried out inside Sony. Currently, it is unclear if and when this robot will be available to a larger research community, but the costs of Qrio have been compared to the price of a luxury car.

Unlike Qrio, Hoap-2 (25 DOF, 50cm tall), developed by Fujitsu [4], has been sold to some labs for about USD 50,000. It is used by the RoboCup team Senchans A, but never won the competition.

A taller humanoid, Asimo, has been developed by Honda [5]. The recently announced research version has 34DOFs and a height of 130cm. It can walk at 69cm/s and jogs at 83cm/s. It is possible to rent Asimo for about USD 162,000 per year for presentations.

Approximately the same size of Asimo has a trumpet playing humanoid robot which has been announced recently by Toyota [20]. It is displayed at Expo 2005 in Aichi, Japan.

While these humanoid robots are impressive, they are not available to researchers outside the industry labs or are too expensive for academic research. There are few publications about the details of humanoid robots developed by industry. Academic projects, like H7 of Tokyo University [7, 12] and HRP-2P [8] of the Japanese AIST are better documented.

Many existing humanoid robots are not adapted to the needs of soccer competitions. In soccer, the robots must survive a fall. Some robots even jump or lie down and get up by themselves again. The soccer robots must also be fully autonomous. Many existing robots have only limited senses. Some of them rely on external computers for behavior control. It might also prove difficult to get access to the lower levels of their APIs.

In addition to the expensive larger humanoid robots, which are usually driven by DC-motors and harmonic drive gears, some smaller servo-driven humanoid robots have been developed recently [10, 21, 23]. Some of these robots performed well at RoboCup competitions.

The servo-driven robots have up to 22DOFs and a size of 30-40cm. As the competition rules require the robots now to be fully autonomous, it becomes difficult for these small robots to carry a camera and sufficient computing power. The small body size also limits the walking speed of these robots.

To overcome the described limitations, we designed a taller servo-driven humanoid robot that keeps the low weight of the smaller robots. We added a wide-angle camera, ample computing power, and wireless communication to it.

3 Mechanical Design

Fig. 1 shows two views of our humanoid robot Toni, ready to kick the ball. As can be seen, Toni has human-like proportions and a slim appearance. Its mechanical design focused on weight reduction. Toni is 74cm tall and has a total weight of only 2.2kg.

The robot is driven by 18 servo motors: 6 per leg and 3 in each arm. The six leg-servos allow for flexible leg movements. Two orthogonal servos form the 2DOF hip joint and the 2DOF ankle joint. One servo drives the knee joint.

A special feature of Toni is its active toes joint, located in the foot plate. It allows for over-extending the leg. This is needed for walking with a straight stance leg. Most humanoid robots shorten the stance leg by bending the knee

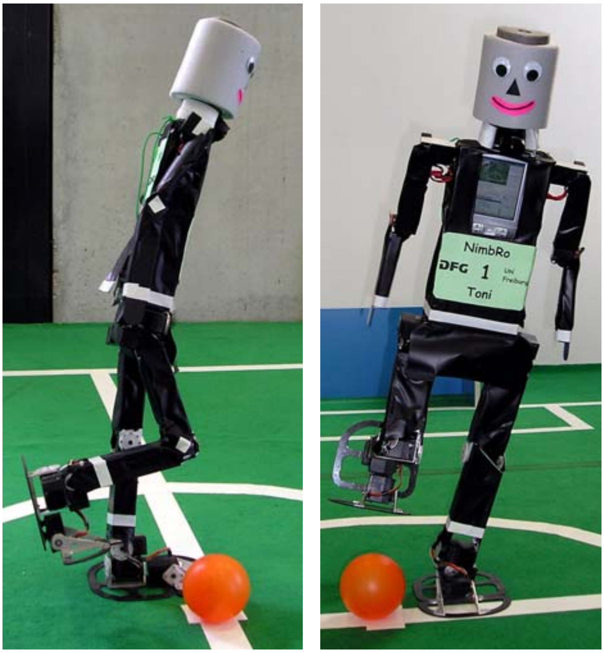


Fig. 1. Two views of the humanoid robot Toni, ready to kick

joint to avoid singularities. This requires high-torque knee actuators and leads to an unnatural gait pattern. When the stance leg is kept straight, torques are reduced and the gait is more natural. However, when walking with large steps, it is now necessary to over-extend the other leg before toes-off in order to shift the weight to the stance leg. The extra segment between the ankle joint and the toes joint provides this extra leg extension [1]. The toes joint is also used when kicking the ball. To our knowledge, only the H6/7 robots of Tokyo University possess active toes joints [13].

We selected the S9152 servos from Futaba to drive the hips, the knees, and the ankles. These digital servos are rated for a torque of 200Ncm. They have a weight of only 85g. The toes joints need less torque. They are powered by JR 8511 servos (185Ncm, 66g). We augmented all servos by adding a ball bearing on their back, opposite to the driven axis. This made a stiff joint construction possible.

Toni's arms do not need to be as strong as the legs. They are powered by SES640 servos (64Ncm, 28g). Two orthogonal servos constitute the shoulder joint and one servo drives the elbow joint.

The skeleton of the robot is mostly constructed from aluminum extrusions with rectangular tube cross section. In order to reduce weight, we removed all material not necessary for stability. Toni's feet and its arms are made from sheets of carbon composite material. Its head is there for completeness only. It is made of lightweight foam.

4 Electronics

Toni is fully autonomous. It is powered by high-current Lithium-polymer rechargeable batteries, which are located in its pelvis. Two Kokam 2000H cells last for about 30 minutes of operation. They can be discharged with 30A and have a weight of only 110g.

The servos are interfaced to three tiny ChipS12 microcontroller boards, shown in Fig. 2(a). One of these boards is located in each thigh and one board is hidden in the pelvis. These boards feature the Motorola MC9S12C32 chip. This 16-bit controller belongs to the popular HCS12 family. We clock it with 24MHz. It has 2kB RAM, 32kB flash, a RS232 serial interface, CAN bus, 8 timers, 5 PWM channels, and 8 A/D converters. We use the timer module to generate pulses of 1...2ms duration at a rate of 180Hz in hardware. These pulses encode the target positions for the servos. Up to eight servos can be controlled with one board.

In order to keep track of the actual servo movements, the potentiometer voltages are digitized by the A/D converters and processed by the microcontrollers. By analyzing the temporal fine structure of these signals, we estimate not only the current servo positions, but also the PWM duty cycles of their motors.

In addition to these joint sensors, Toni is equipped with an attitude sensor, shown in Fig. 2(b). It consists of a dual-axis accelerometer (Analog Devices ADXL203, ±1.5g) and two gyroscopes (ADXRS 150/300, ±150/300 deg/s). This attitude sensor is located in its pelvis. The four analog sensor signals are digitized with A/D converters of the HCS12 and are preprocessed by the microcontroller.

The microcontrollers communicate with each other via a CAN bus at 1MBaud and with a main computer via a RS232 serial line at 115KBaud.

Every 6ms, target positions for the servos are sent from the main computer to the HCS12 boards. The microcontrollers send the preprocessed sensor readings back. This allows to generate smooth trajectories for the servos and to keep track of the robot's state in the main computer.

We use a Pocket PC as main computer, which is located in Toni's chest (see Fig. 1). The model FSC Pocket Loox 720 has a weight of only 170g, including the battery. It features a 520MHz XScale processor PXA-272, 128MB RAM, 64MB flash memory, a touch-sensitive display with VGA resolution, Bluetooth, wireless LAN, a RS232 serial interface, and an integrated 1.3 MPixel camera.

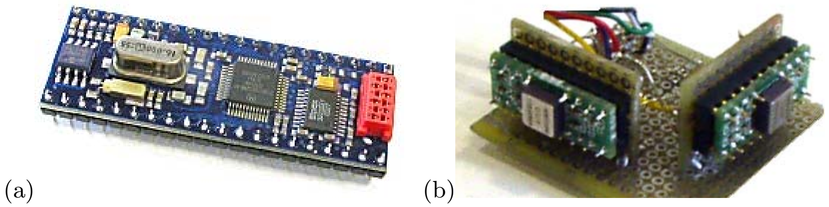


Fig. 2. Electronics: (a) ChipS12 microcontroller board featuring HCS12C32; (b) Attitude sensor consisting of a dual-axis accelerometer and two gyroscopes

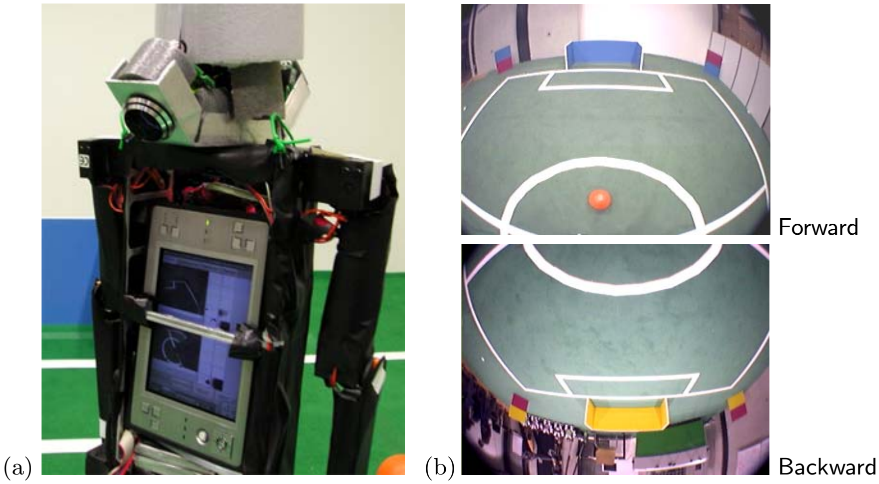


Fig. 3. Toni with a small PC and two ultra-wide-angle USB cameras: (a) Picture of the robot's back and neck; (b) Pictures captured simultaneously from the two cameras

This computer runs behavior control, computer vision, and wireless communication. It is equipped with a Lifeview FlyCAM CF 1.3M that has been fitted to an ultra-wide-angle lens. The lens is located approximately at the position of the larynx. The wide field of view of this camera (vertically about 112°) allows Toni to see at the same time its own feet and objects above the horizon. The horizontal field of view is approximately 150°.

We also tested a Sony Vaio U750P PC as main computer, which is small enough to fit into Toni's trunk, as shown in Fig. 3(a). The PC has a weight of 550g, including batteries, and features an ultra-low voltage 1.1GHz Pentium M 733 processor, 512MB RAM, 20GB harddrive, a touch sensitive display with SVGA resolution, and wireless LAN.

This PC is interfaced to two ultra-wide-angle USB cameras, also visible in Fig. 3(a). They consist of webcam electronics, a 1/3"CCD imager, and a door viewer lens. Toni can see almost all objects around it when these cameras are pointed towards the front and the back. Two simultaneously captured images are shown in Fig. 3(b). The cameras deliver up to 30fps and the PC is fast enough to process the images and to localize the robot at this rate. The total weight of the robot in this configuration is only 2.7kg.

5 Perception

In order to play soccer, Toni needs information about itself and about the situation on the field. In particular, it must know where the ball and the goals are, localize itself on the field, and perceive other players.

We fuse the accelerometer and gyro readings to obtain an estimate of Toni's attitude. This is done by integrating the rotation speeds measured by the

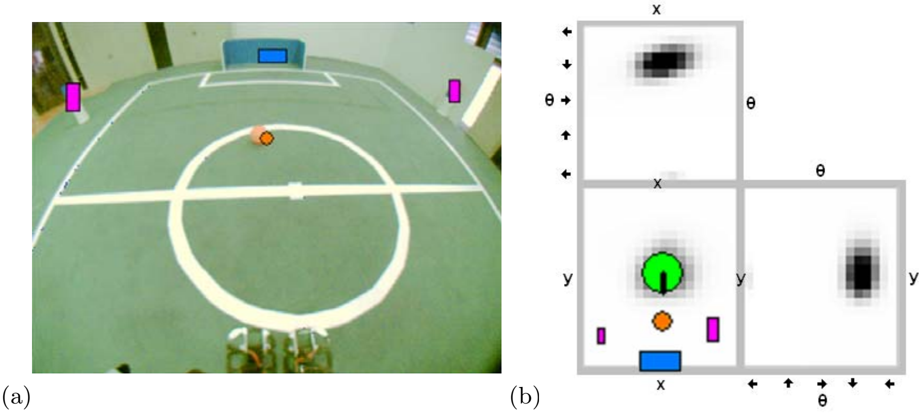


Fig. 4. (a) Image captured from Toni's perspective while it was walking. Detected objects: goal (blue horizontal rectangle), ball (orange circle), and pole (magenta vertical rectangle); (b) Three 2D projections of the 3D grid representing the probability distribution of robot poses (x, y, θ). The green circle is drawn at the estimated robot location (x, y). The black line represents its estimated orientation θ. The detected objects are drawn relative to the robot

gyroscopes. The long-term readings of the accelerometers are used to provide a starting point for the integration. Furthermore, the biases of the gyros are compensated using the accelerometer readings. In combination with the measured joint angles, the attitude allows to reconstruct the camera pose.

The only source of information about Toni's environment is its camera. The wide-angle CF color camera allows seeing the ball at the robots feet, the goal, and poles simultaneously (see Fig. 4(a)). In RoboCup soccer, these key objects are color-coded. The ball is orange, the field is green, goals are yellow and blue, and poles contain magenta and a goal color.

Our computer vision software converts the captured RGB images into the YUV color space to decrease the influence of different lighting conditions. The colors of pixels are classified with the pie-slice method [19]. In a multistage process insignificant colored pixels are discarded and the colored objects are detected. Their coordinates are estimated in an egocentric frame (distance to the robot and angle to its orientation), based on image position and object size. These relative coordinates suffice for many relative behaviors, like positioning behind the ball while facing the goal.

To implement global team behaviors, such as kick-off, we need the robot coordinates in an allocentric frame (position on the field and orientation). We estimate these using a probabilistic Markov localization method that integrates egocentric observations and motion commands over time. As proposed by Fox, Burgard, and Thrun [3] this method uses a three-dimensional grid (x, y, θ) to represent the probability distribution of robot poses. The grid is shown in Fig. 4(b).

6 Behavior Control

We control Toni using a framework that supports a hierarchy of reactive behaviors [2]. This framework allows for structured behavior engineering. Multiple layers that run on different time scales contain behaviors of different complexity. This framework forces the behavior engineers to define abstract sensors that are aggregated from faster, more basic sensors. Abstract actuators give higher-level behaviors the possibility to configure lower layers in order to eventually influence the state of the world.

The framework also supports an agent hierarchy. For Toni, we use three levels of this hierarchy: individual joint, body part, and entire robot. This structure restricts interactions between the system variables and thus reduces the complexity of behavior engineering.

The lowest level of this hierarchy, the control loop within the servo, has been implemented by the servo manufacturer. It runs at about 300Hz for the digital servos. We monitor target positions, actual positions, and motor duties.

At the next layer, we generate target positions for the individual joints of a body-part at a rate of 167Hz. We make sure that the joint angles vary smoothly. This layer implements an interface that describes the behavior of body parts. For example, the entire leg can be positioned using leg extension (the distance from the hip joint to the ankle joint), the leg angle (angle between the pelvis plate and the line from hip to ankle), and foot angle (angle between foot plate and pelvis plate).

Such a more abstract actuator space simplifies the implementation of dynamic walking on the next layer. A central pattern generator running in the trunk determines the step frequency. Both legs derive their own gait phase by shifting the trunk phase by $\pm\pi/2$. Based on this gait phase, each leg generates trajectories for its leg extension, leg angle, and foot angle.

The two key ingredients for generating dynamic walking are lateral shifting of the robots center of mass, and movement of the legs in walking direction. The swinging leg is shortened while it is moved quickly into the walking direction. At

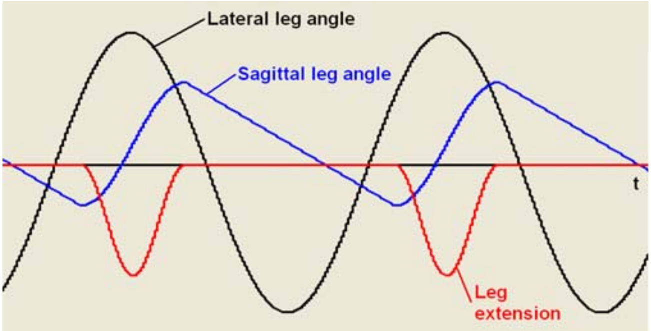


Fig. 5. Dynamic walking. Leg angles and leg extension of a leg while walking forward with small steps. The other leg moves in the same way, with a phase offset of π.

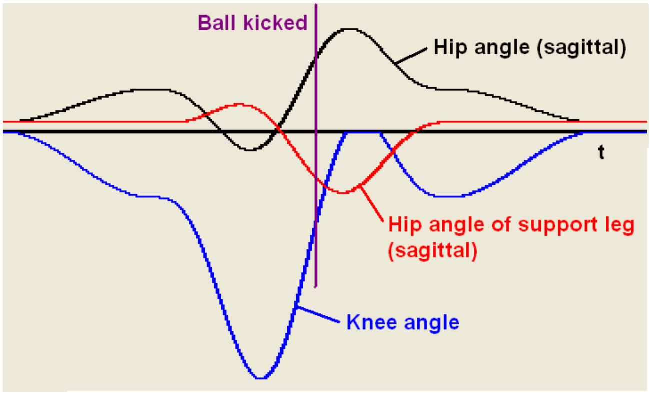


Fig. 6. Kicking. Joint angles of the knee of the kicking leg and the sagittal hip joints of both legs.

the same time, the supporting leg has maximal extension and is moved slowly against the walking direction. The resulting trajectories of leg angles and leg extension are shown in Fig. 5 for the case of walking in forward direction with small steps.

Toni's maximum walking speed is about 20cm/s. The robot is not only able to walk forward and backward, but it can also walk to the side. By blending these gait directions, we generate omnidirectional walking. Toni is also able to turn on the spot. We used this interface to implement higher-level behaviors, like approaching the ball, dribbling, and positioning for penalty kicks.

In addition to omnidirectional walking, we implemented a kicking behavior for Toni. Fig. 6 shows trajectories for the sagittal hip joints of both legs and the knee joint of the kicking leg. The kicking leg strikes out, accelerates smoothly until it hits the ball and decelerates again. Note that all three joints reach their maximum speed in forward direction when the ball is hit. The kicked ball rolls for approximately two meters on carpet. Afterwards, Toni goes back to a stable stand.

Toni's toes joints are used when walking with larger steps during the double-support phase. The unloading leg is over-extending to push the center of mass above the loading leg [1]. We also used the toes joints to balance Toni on the frontal part of the foot plate (toes).

7 Infrastructure

In addition to the robot itself, we implemented some infrastructure components to support behavior engineering.

• **Simulation:** In order to be able to design behaviors without access to the real hardware, we implemented a physics-based simulation for Toni. This simulation is based on the Open Dynamics Engine [15]. It is visualized in real time using OpenGL [18].

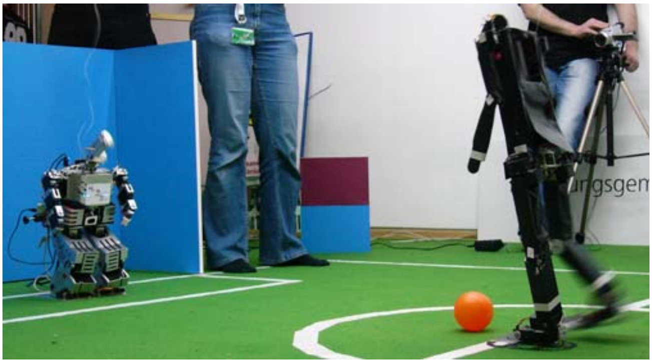


Fig. 7. Penalty Kick demos at RoboCup German Open 2005: Toni vs. Mr. DD

- **Communication:** Toni is equipped with a wireless network adapter. We use the wireless communication to transmit via UDP debug information to an external computer, where it is logged and visualized. This computer is also used to fuse local views to a team view when multiple robots are playing. We plan to use it to compute team behaviors, such as the assignment of roles to the individual players. We also plan to use the wireless network for transmitting the game state (kickoff, penalty ...) to the robots.
- **Test Behaviors and Remote Control:** On all levels of the behavior architecture, we implemented test behaviors that generate sequences for the abstract actuators in order to test the behavior of the lower layers. Similarly, the lower layers can also be tested by setting the actuators to the readings of a joystick. In this way, the user can determine an individual joint angle, a leg parameter, or the walking direction and speed of the robot.
- **Monitoring:** All variables of the system, such as joint angles, sensor readings, actuator values, and behavior activations, are logged. They can be visualized live or after an experiment. The analysis of not only the observable robot behavior, but also its internal state is extremely helpful when debugging behaviors.

8 Conclusions

This paper described the humanoid robot Toni, which was designed to play soccer. It detailed its mechanical and electrical design, perception, behavior control, and infrastructure.

Toni is fully autonomous, has a low weight of only 2.2kg, and is much taller (74cm) than most servo-driven robots. It has 18 DOFs, with toes joints as special feature. The robot is equipped with a wide field-of-view color camera, ample computing power, and wireless communication.

Toni possesses basic soccer skills. It can walk dynamically in all directions and is able to turn on the spot. It perceives the ball and the goals and localizes itself on the field. Toni is able to approach the ball and to dribble it. It is able to kick the ball without falling.

We presented Toni for the first time to the public during the 21st Chaos Communication Congress (Berlin, Dec. 2004), where it was playing with a soccer ball. At RoboCup German Open (Apr. 2005), we showed penalty kick demonstrations against Mr. DD of Darmstadt Dribblers. Toni was able to approach the ball, such that the ball was located in front of its kicking foot and the robot was facing the goal. Toni slowed down when coming close to the ball and triggered its kicking behavior when positioned well enough. The robot smoothly stopped walking, reached out and kicked the ball strongly towards the goal.

For the RoboCup 2005 Competition, which took place in July in Osaka, Japan, we constructed three more robots, based on Toni's technology. Jupp and Sepp have a size of 60cm and played in the KidSize class. Max has a size of 70cm and played in the MidSize class. All three robots have an additional yaw-joint in the thigh, a pitch-joint in the trunk, and stronger arms. They can rotate easier on the spot and are able to get up after a fall. These robots performed well. Our team NimbRo got 2nd and 3rd in the Technical Challenge, next to Team Osaka. Max won the MidSize Penalty Kick final 3:0 against Aria (Iran). Sepp and Jupp reached the final in the 2 vs. 2 soccer games. They lost 2:1 against Team Osaka. They also scored the second highest number of goals in the Penalty Kick competition, next only to Team Osaka. In the overall Best Humanoid ranking, Osaka won, NimbRo KidSize came in 2nd, and NimbRo MidSize was 3rd.

Videos of the competitions can be found at: http://www.NimbRo.net.

Acknowledgements. This work is supported by grant BE 2556/2-1 of Deutsche Forschungsgemeinschaft (DFG). The following students contributed to Toni's development: Thorsten Kramer, Tobias Langner, Jörg Stückler, Konstantin Welke, and Rui Zhou. The RoboCup team FU-Fighters provided the sources of a framework for implementing a hierarchy of reactive behaviors.

References

1. Sven Behnke. Human-like walking using toes joint and straight stance leg. In *Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines (AMAM2005), Ilmenau, Germany*, to appear Sept. 2005.
2. Sven Behnke and Raul Rojas. A hierarchy of reactive behaviors handles complexity. In M. Hannebauer, J. Wendler, and E. Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 125–136. Springer, 2001.
3. Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.
4. Fujitsu. HOAP-2. http://www.automation.fujitsu.com/en/products/products09.html.
5. Honda. ASIMO. http://world.honda.com/asimo.
6. Shuji Imura and N. Maeda. Description of HITS dream team 'Firstep' of Honda International Technical School (HITS). In *Team descriptions for RoboCup 2003 Humanoid League*, 2003.

7. Satoshi Kagami, Koichi Nishiwaki, James J. Kuffner Jr, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Online 3D vision, motion planning and bipedal locomotion control coupling system of humanoid robot: H7. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02), Lausanne, Switzerland*, pages 2557–2563, 2002.
8. Kenji Kaneko, Fumio Kanehiro, Shuuji Kajita, Kazuhiko Yokohama, Kazuhiko Akachi, Toshikazu Kawasaki, Shigehiko Ota, and Takakatsu Isozumi. Design of prototype humanoid robotics platform for HRP. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'02)*, 2002.
9. Hiroaki Kitano and Minoru Asada. The robocup humanoid challenge as the millennium challenge for advanced robotics. *Advanced Robotics*, 13(8):723–737, 2000.
10. Kondo Kagaku Co., Ltd. KHR-1. http://www.kondo-robot.com.
11. Norbert M. Mayer. Humanoid Kid Size League and Medium Size League rules and setup. http://er04.ams.eng.osaka-u.ac.jp/humanoid_webpage/humanoid.pdf.
12. Koichi Nishiwaki, Satoshi Kagami, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired ZMP. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02), Lausanne, Switzerland*, pages 2684–2690, video 615, 2002.
13. Koichi Nishiwaki, Satoshi Kagami, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Toe joints that enhance bipedal and fullbody motion of humanoid robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, pages 3105–3110, 2002.
14. Masaki Ogino, Masaaki Kikuchi, Junichiro Ooga, Masahiro Aono, and Minoru Asada. Optic flow based skill learning for a humanoid to trap, approach to, and pass a ball. In *RoboCup 2004: Robot Soccer World Cup VIII*, pages 323–334, 2005.
15. Russel Smith. Open Dynamics Engine. http://opende.sourceforge.net.
16. Sony. Dream Robot QRIO. http://www.sony.net/qrio.
17. Team Osaka. VisiON. http://www.sansokan.jp/robot/info/vision_en.html.
18. The Industry's Foundation for High Performance Graphics. OpenGL. http://www.opengl.org.
19. Peter J. Thomas, Russel J. Stonier, and Peter J. Wolfs. Robustness of colour detection for robot soccer. In *Proc. of 7th Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, volume 3, pages 1245–1249, 2002.
20. Toyota. Partner Robot. http://www.toyota.co.jp/en/special/robot.
21. Vstone Co., Ltd. http://www.vstone.co.jp.
22. Chris Willis. World's greatest android projects. http://www.androidworld.com.
23. Changjiu Zhou and Pik Kong Yue. Robo-Erectus: a low-cost autonomous humanoid soccer robot. *Advanced Robotics*, 18(7):717–720, 2004.

Using a Symmetric Rotor as a Tool for Balancing

N. Michael Mayer[1,2], Minoru Asada[1,2], and Rodrigo da Silva Guerra[1]

[1] Department of Adaptive Machine Systems,
[2] Handai Frontier Research Center (FRC)
Osaka University, Osaka, Japan
{norbert, asada, guerra}@er.ams.eng.osaka-u.ac.jp

Abstract. In the Humanoid Leagues balancing during walking and running is still the biggest challenge for most of the teams. We present here some work in which a dynamic walker is stabilised by using a fast heavy rotor, a gyro. The dynamics of a symmetric, fast rotating gyro is different from that of a non-rotating solid body, e.g. in the case of small disturbances it tends to keep the axes the same. Results show that the rotor enhances the stability of the walking in the simulations. In a model for an actuated robot the rotor is used as a reaction wheel, i.e. the pitch of the robot is stabilised by accelerating and decelerating it. We see this method –though it is not biologically inspired – as an intermediate step for learning balancing in biped robots. The control algorithm responsible for balancing the pitch is discussed in detail. The simulations show that, by using this kind of stabilisation, movements like stand up, walk and jump are easily possible by using open loop control for the legs, however high torques for the rotor are necessary. Finally, a robot design that consists just of a trunk is presented.

1 Introduction

Balancing men-like walking is still one of the biggest challenges of robot control. The use of trajectory-based control and zero moment points for statically stable walking are among the classical approaches to the problem [1]. Nevertheless, these methods tend to perform slower movements and still consume more energy (far beyond humans of the same weight and size, cf. e.g. the Honda Asimo specifications [2]). Recently, several new approaches have appeared that aimed to overcome this situation. The so-called passive dynamic walking robots (PDW robots) are certainly among the most pronounced examples in this category [4].

The present work proposes the use a heavy fast rotating gyro attached to the robot's body (e.g. a PDW) as a way to stabilise roll and yaw. The principle of the application of the gyro was already outlined in the literature [9]. The dynamics of the gyro can be described by the Euler equations.

Results from an investigation using a gyro to stabilise an – apart from the gyro – non-actuated passive dynamic walker are presented in the first part of the results of this paper. Additionally, this work also explores how accelerations and decelerations in the rotation of the gyro can be controlled in a way such that the pitch is also balanced – gyro serves as a reaction wheel (also called inertia

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 71–80, 2006.

actuator). A non-legged rolling robot actuated only by the gyro/reaction wheel effect was simulated. Finally, further investigations of the application of these same principles in an actuated robot are also presented. In a final discussion we concentrate on how such a device can be realized.

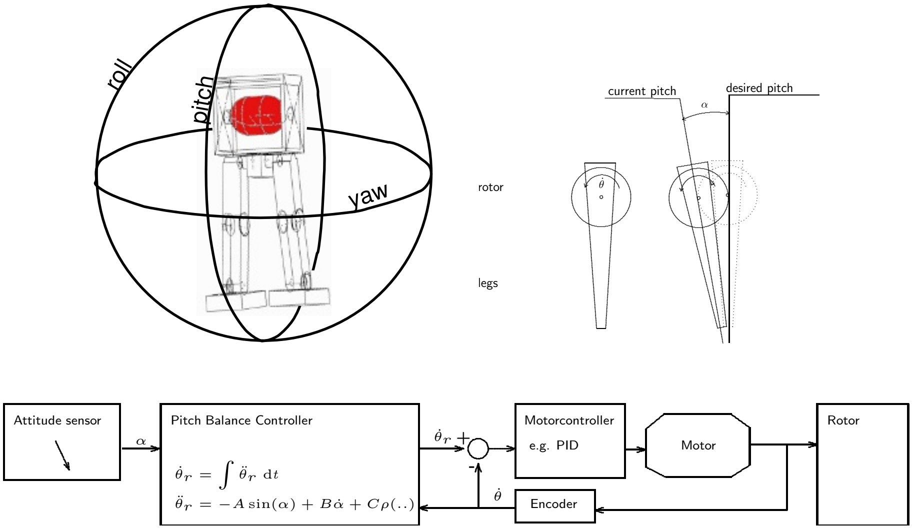


Fig. 1. Above left: Scheme for the proposed actuator in a biped robot. The rotor is marked bold. The rotor's axis is parallel to the hip. Above right: Schematic view from the left side of the robot. The pitch angle is α and the speed of the rotor $\dot{\theta}$. The positive values mean both the velocity and the pitch angle have the same direction. Below: Feedback loop for pitch balancing: The controller uses information about the attitude of the robot and the encoder values from the rotor as input, giving α as the output (e.g. gyroscope and gravity sensor combination). The parameters A, B, C have to be adapted to the properties of the robot.

1.1 Passive Dynamic Walking

The idea behind PDW is to exploit the natural dynamics of pendulum-like legs so as to achieve fast and economic walking in bipedal robots. Explicit methods of motion analyses like Poincaré Return Maps are often used in order to find the stable attractors of the physical motion dynamics. By doing this PDW research seeks feasible designs which, for being controlled, require least energy consume or are optimal with respect to some other eligible criteria.

Usually two-dimensional walkers [3, 7] are used as an intermediate towards fully-dimensional PDW systems. For instance, PDW walkers such as those studied by McGeer and colleagues show a moderately stable gait at downhill slopes without using any control whatsoever. With respect to speed and smoothness of movements these walkers can compete with state-of-the-art humanoid robots, provided the slope is sufficiently steep - however, speed is determined by the slope and payload, being unchangeable for a given design.

In 2D PDW, only the dynamics throughout the cross-sectional vertical plane is considered, leaving roll and yaw stable by definition. This is done either by disregarding depth information in simulations or by designing iso-static shaped legs for directions outside the plane of study in real robots. This is often seen as a valid first approach for concerning only about pitch dynamics on the way towards fully-dimensional implementations. Still, it is a big technological challenge to further the balance control over all three directions – pitch, roll and yaw – simultaneously.

1.2 Gyro/reaction Wheel Actuator

Gyros are symmetric rotors that spin fast giving rise to very specific physical properties of rigidity, precession and nutation. Gyros became well known for being applied both as sensors (e.g. gyroscopes) and as actuators (e.g. camera stabilisers). In the present time gyros are being used in many technical devices such as satellites, artillery, navigation units, cameras, planes and so forth. As for robotics, gyros are certainly most well known as pan/tilt sensors (gyroscopes). Nevertheless, there are a few works in the robotics field in which gyro are explored as means for stabilising and (less common) acting as a reaction wheel. A worth-mentioning robotic gyro-actuator approach accounting for both stabilisation and inertial effects is the Gyrover robot [8]. Gyrover is a robot in s wheel-shaped body which rotates on its own axis, driven by an asymmetric internal rotor (stabilising effect in external body and reactive inertial effect due to internal mass). Approaches for the use of gyro-actuators in biped robots have also been done in previous studies [10, 11]. In these studies the axis of the rotor was set parallel to the direction of motion of the robot, whereas in the present study the axis of the gyro is set parallel to the hip.

By positioning the gyro parallel to the hip, additionally to roll and yaw stabilisation, the gyro reaction wheel effect can be controlled such as to balance the pitch.

Summarising, this gyro/inertia actuator can influence the robot's dynamics in either of two ways:

1. by stabilising roll and yaw through rotation – the higher the speed, the slower the robot reacts to perturbations. In this case care should be taken on what refers to precession and nutation unusual and unexpected effects;
2. by acting as a reaction wheel, exploiting inertial effect through accelerations and decelerations. This is only useful if controlled in closed-loop.

These features were simulated in different robotic configurations and results from these investigations are presented in this paper.

1.3 Simulation Engine

The Open Dynamics Engine (ODE) open source mechanics and dynamics simulator was used as a framework for the experiments herein mentioned [12]. The value of the gravitation constant was set to 9.81 units, so as to couple metric interpretation to all physical measures. Therefore, one time unit in the simulation can be interpreted as one second and one distance unit can be interpreted as one meter.

2 Gyro Stabilised Passive Dynamic Walker

The slope was set to 3.0 degrees. The walker was designed simple. The hips and knees were hinge joints and the round shaped feet were fixed at the end of the each respective lower leg. The gyro's rotor axis was implemented as an unbounded hinge joint parallel to the hip axis, perpendicular to the direction of motion. The masses of the parts were the following: upper leg 0.07 kg, lower leg 0.012 kg, middle part 0.011 kg, rotor (heavy gyro) 0.565 kg. The speed of the gyro was set to 175 rad/s. The hinge joints of the knees were bounded to stop at an angle of 0.23 rad, which allows a stable stance during walking. The joints were hinge joints without friction. The starting conditions were optimised manually.

The walker was not actuated except for the heavy gyro which was capable of spinning in a constant constant speed. In turning on/off the gyro-actuator the resulting effects due to gyro properties could be readily detected. Simulations demonstrated the PDW robot could walk several more steps with the rotating gyro than with the gyro disabled.

For screen-shots of the simulated passive dynamic biped please cf. Fig. 3. The walker was able to walk up to five steps in the simulation with the rotating gyro. In contrast, for the case of the non-rotating gyro, the walker wasn't able to walk more than one or two steps.

3 Pitch Balance Controller

As for the pitch it is better to have it as close as possible to the – possibly unstable – balance point. For doing so the authors propose a feedback control system, which necessarily supposes the existence of a sensor for pitch angle error (i.e. actual pitch against desired one). The actual pitch angle could be detected by a gyroscope. In the following this pitch error shall be referred to as α.

The control equation was implemented as

$$\ddot{\theta}_r = -A\sin(\alpha) + B\dot{\alpha} + C\rho(\dot{\theta}) \tag{1}$$

Where $\ddot{\theta}_r$ is the motor speed control signal sent to the motor controller, e.g. a PID controller. Dynamics of this motor controller were not taken into consideration. The constants A, B and C depend on the size, weight and current shape of the robot and have to be optimised similarly as it happens in a PD controller.

The principle of the actuator is the one of a reaction wheel. As in PD control there are parameters that can to be calculated analytically if the values of the

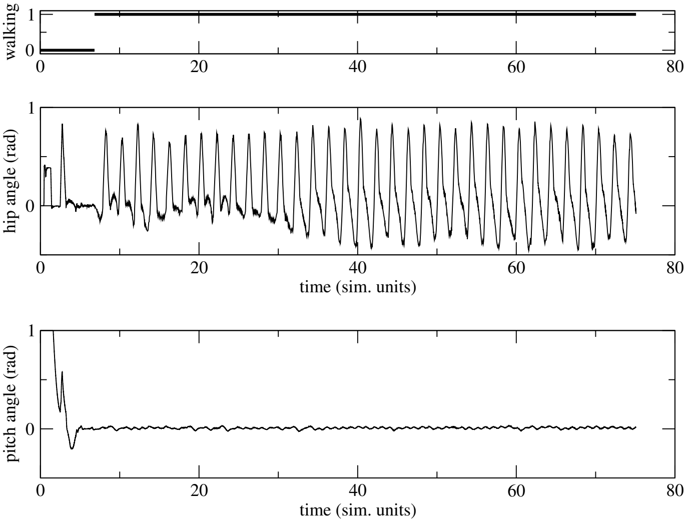


Fig. 2. Motion patterns of walking in the actuated biped robot. On top: the state of the waling behavior. Middle: Motion pattern of the left hip. The motion pattern is not entirely regular. Below: The state of the pitch value (α). A small oscillation is to see.

inertia tensors of the controlled robot and the rotor are known, which is comparable to the P value in the PD controller paradigm. The parameter B prevents the overshoot and is thus analogous to the D value of PD controller paradigm.

The parameter C and the the function $\rho(..)$ can be designed so as to keep the speed of the rotor within an operable range. The specific design of $\rho(..)$ depends in particular on the type of the balance point. For example if the balance point is unstable the following function $\rho(..)$ can be applied in the controller equation

$$\rho(\dot{\theta}) = H\left(\dot{\theta} - \dot{\theta}_{\text{opt}}\right) \tag{2}$$

where $H(x)$ is a piecewise linear function, which is whether equal to x if within the limits of $|x| < H_{lim}$ or either of H_{lim} or $-H_{lim}$ for bigger or lower values of x, respectively. This design makes the robot move slightly ahead its balance point if the rotor speed is low and behind it if the rotor speed is too high. It causes a continuous ingression – or degression, respectively – of the rotor speed in order to balance the robot. In case of a stable balance point just the inverse can be used

$$\rho(\dot{\theta}) = -H\left(\dot{\theta} - \dot{\theta}_{\text{opt}}\right) \tag{3}$$

in order to control the rotor speed. The parameters C, H_{lim} should be chosen to be small enough not to interfere with the balancing, yet strong enough to keep the rotor speed in its limits and to let it converge against $\dot{\beta}_{opt}$.

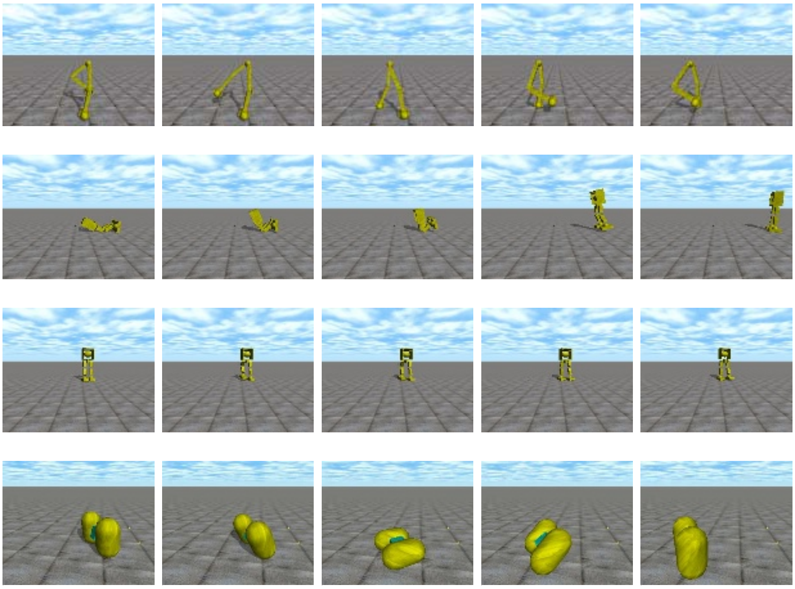


Fig. 3. Simulation results for three variant types of robots: First row shows a non-actuated passive dynamic biped walker (PDW) that is stabilised with a gyro that rotates with constant speed (open loop control). Second row shows an actuated biped robot with 3 degrees of freedom per leg and an actuated gyro (close loop control). The third row shows the same robot walking using an open loop control for the legs. Fourth row shows an robot the only consists of a trunk. This robot can move just by using the acceleration and deceleration of an internal reaction wheel.

4 Results for an Actuated Walker with Gyro Reaction Wheel

The pitch balance controller was tested on a biped robot with 3 degrees of freedom in each leg. The simulation resulting in standing up, walking and in jumping was done with the following parameters:

Body width	1.0 u	rotor radius	0.25 u
Body height	1.3 u	foot length	0.80 u
Body depth	0.6 u	foot height	0.20 u
upper leg length	0.5 u	foot width	0.50 u
lower leg length	0.6 u	Dist. betw. legs	0.35 u

The values are given in units of the simulation program which might be thought as metric units since gravitation was set to 9.81 units.

In the simulations the walker control was able to perform the following three functions: Stand up, walk and jump. For walking the step length was variant and had a maximum of about 0.4 units per step. The robot was able to walk for a long period without falling down. However, the speed of the rotor tended for go out of its boundaries, so that the robot should stop while by while and recover the optimal speed of the rotor. The reason for this is that phases of stable and unstable balance occur during the walking, and thus the optimal control (i.e. leaning forward and backward) interchanges eventually driving rotor speed out of its boundaries. The graph in Fig. 2 shows that the walking (indicated by the regular pattern of hip movements) is anticipated by an oscillation of the pitch angle (α). This happened because the constant A was set such as to make the pitch controller have a moderated suboptimal control. Experiments showed that less strict pitch control resulted in better walking patterns than the more strict ones. So it turned out to be useful that the walking is anticipated elastically by the hip movement. In addition, the swinging leg is moving downward before it hits the ground. During jumping the robots attitude could be controlled while it was completely in the air.

The motor specifications in the simulations tend to be out of the limits of present technology. However, they can be adapted by using a more advanced control and some mechanical tricks like brakes (please cf. the discussion section of this paper).

5 Robot Without Legs

Briefly a non-legged robot was simulated that used only the gyro for locomotion. The robot consists of two relevant parts with regard to its dynamics: A symmetric rotor and the body. The body is moved by its own inertia while the rotor is accelerated and decelerated. The control of this robot can be performed manually. Preliminary experiments show that the The design seems suitable for very light robots. The robot designed here is similar to the Gyrover [8]. The main difference between our robot and the Gyrover is that the first has its locomotion supported by an asymmetry in the robots body, whereas the the locomotion in the Gyrover is supported by an asymmetry of the shape of the rotor.

6 Summary of the Results and Conclusion

We investigated the applicability of a combined reaction wheel/gyro actuator to a biped walker by using numerical simulations. As simulation environment the open dynamics engine (ODE) realistic mechanic simulator [12]. We presented three examples, two of legged and one of a non-legged robot to demonstrate the virtues of this kind of approach. In total we demonstrated three approaches:

1. A biped passive dynamic walker with a non-actuated gyro. In this approach the balancing by the rotor was restricted to yaw and roll. Though stable walking was not accomplished the simulations showed that a rotating gyro resulted in an improved stability in comparison to the non-rotating case.
2. A biped robot with three actuated degrees of freedom in each leg. The gyro stabilised the pitch by accelerating and decelerating. The walker was able to

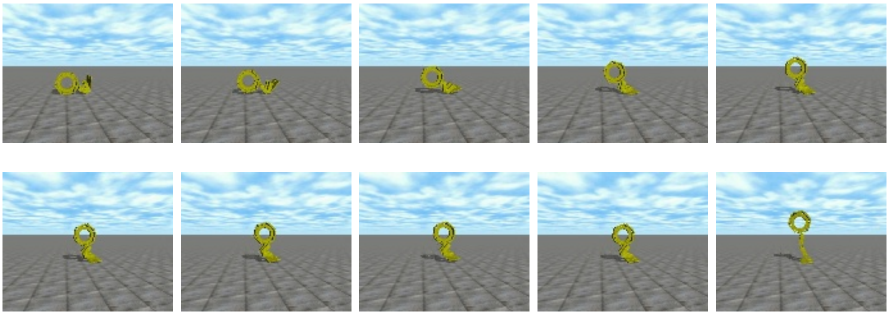


Fig. 4. Biped with realistic specifications of a motor/brake combination standing up

walk for virtually infinite times, though the walking has to be interrupt time by time to adjust the rotor speed.

3. A non-legged robot with only one degree of freedom that was the actuated gyro. This robot was able to move by just accelerating and decelerating the speed of the gyro. Thus, in this case the gyro is used not only for balancing but also for locomotion of the robot.

7 Discussion

Intention of this work was to design an applicable control for balancing the robot. We achieved this goal in the simulation framework. Still there are issues that are open to future work and considerations.

The first and biggest issue is how to build a real world robot of this type. The design of a biped robot that uses the gyro effect – similar to the first example – is rather simple to realize. The rotor can be designed with sufficient precision with professional tools. Some preliminary experiments were done already. Fig. 5 shows an example for such a robot.

One more difficult task is to design a reaction wheel actuator that produces sufficient torque. A robot of the size of Sony's Qrio needs roughly a torque of 5-10 Nm in order to stand up.

It seems easy to produce any torque with a motor and a gear-head. The problem is that this torque has to prevail over a sufficient time. The higher the gear transmission ratio is the faster the rotor reaches it maximum rotation speed. In addition, the effect of the gyro vanishes if the rotation of the rotor is not sufficiently fast. Another way to produce torque is to use mechanical or electrical brakes. Thus, a control can accelerate the rotor slowly in a first phase and then by using the brake it can rapidly change the attitude of the robot. On example of standing up has been simulated for a biped robot.

We did one additional simulation with ODE that showed that such a design is possible if the brake/rotor system can produce sufficient torque. Fig. 4 shows a motion pattern for standing up for a real world robot. The simulated brake can produce 2 Nm; the rotor speed is about 4000 RPM. By using an appropriate leg control the robot was able to stand up.

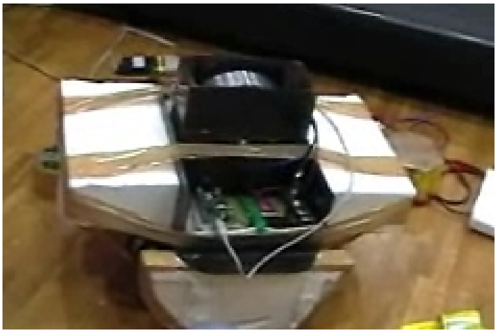


Fig. 5. A experimental setup for a gyro/reaction wheel (at Freiburg University)

For the project proposed to NEDO the brake method has been suggested and presently such a motor brake combination is being designed.

On additional important question is what the possible applications could be appropriate for such kind of actuator. First, the rotor can be thought as an additional balancing device that supports the balancing in an intermediate step of the design of a biped robot. For example during the early stages of a reinforcement or evolutionary learning process of walking and running, rough terrain tasks and others more. In this kind of intermediate steps the speed of the rotor can be reduced more and more and thus, the yaw and roll stabilisation be learned in convenient steps. With regard to the non-legged robot we see our experiments as preliminary tests for a possible design of a simple locomotion for a robot. One possible area for application is the case that the robot is in a medium where the usage of flexible materials, or gaskets is not possible. One example for this could be underwater robotics in great depth. In this case the cover of the robot can be hard and the actuator is inside the robot, not directly interacting with the surrounding environment.

Acknowledgment

This work was supported by the NEDO program for the Aichi Expo 2005. We thank Mrs. Helena Selbach for revising the English of this work. N.M.M. thanks Sven Behnke and Michael Schreiber, Freiburg University, for their support in the summer of 2004, Ferenc Farkas for his competent advice and Ziton Hsu for her patience and help.

References

1. T. Sugihara, H. Inoue, Real time Humanoid Motion Generation through ZMP Manipulation Based on Pendulum Control, IEEE Intl. Conference on Robotics and Automation, (ICRA 2002) p 1404-1409
2. Calculated from data taken from http://world.honda.com/ASIMO (2004)

3. T. McGeer. Passive Dynamic Walking. International Journal of Robotics Research 9:2 62-82 (1993)
4. SH. Collins, M. Wisse, a. Ruina, A Three-Dimensional Passive-Dynamic Walking Robot With Two Knees and Legs, The International Journal of Robotics Research Vol. 20, No. 7, July 2001, pp. 607-615
5. M. Garcia (1999) PhD Thesis. Cornell University (1999)
6. NM. Mayer, JM. Herrmann, AA. F.-Nassiraei, M. Browne and T. Christaller, An Assymetric 2-D Passive Dynamic Walker, Proceedings of the AROB (Oita), 2004
7. Stabilizing dynamic walking with physical tricks N. Mayer, AA. F.-Nassiraei, F. Farkas, Z. Hsu and T. Christaller, Intl. Conf. on Climbing and Walking Robots (CLAWAR), 2004
8. E. Ferreira, S. Tsai, C. Paredis, and H. Brown Control of the Gyrover: a single-wheel gyroscopically stabilized robot Advanced Robotics, Vol. 14, No. 6, June, 2000, pp. 459 - 475.
9. JA. Christian et al., Development of a Variable Inertia Reaction Wheel System for Spacecraft Attitude Control WR, AIAA Guidance, Navigation, and Control Conference and Exhibit 16 - 19 August 2004, Providence, Rhode Island
10. H. Hirakoso, (Japanese title) (Vol.A) pp 552-555 (1996) (in Japanese)
11. AD. Kuo, Stabilizing the Lateral Motion in Passive Dynamic Walking, Intl. J. Rob. Res., 18:9 p 917-930 (1999)
12. R. Smith et al., The Open dynamics engine, open source library for simulating rigid body dynamics, http://ode.org/

Discovering Relevant Sensor Data by Q-Analysis

Pejman Iravani

The Open University, Milton Keynes MK77AA, UK
p.iravani@open.ac.uk
http:mcs.open.ac.uk/pi56

Abstract. This paper proposes a novel method for supervised classification based on the methodology of Q-analysis. The classification is based on finding 'relevant' structures in the features describing the data, and using them to define each of the classes. The features not included in the structural definition of a class are considered as 'irrelevant'. The paper uses three different data-sets to experimentally validate the method.

1 Introduction

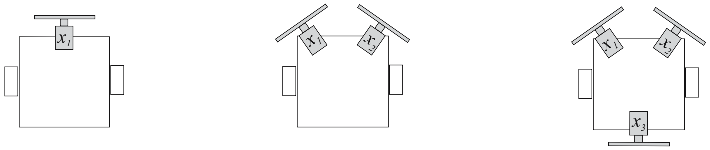
Designing autonomous robots capable of operating in different environments requires, among other things, the necessary sensor information to adapt to each specific situation.

In principle, the designer can *a priori* provide the robot with the set of necessary sensing capabilities. Although this is the most frequent approach, it lacks the flexibility desired for autonomous systems. Moreover, unforeseen situations and possible sensor damage limit the success of the approach.

Another approach would be to design robots with extensive, and even redundant, sensor capabilities. This approach achieves robustness through redundancy (eg. spare sensors) and flexibility through the usage of extensive sensing capabilities. Although theoretically sound, this approach becomes impractical mainly because of the dimensionality implications of adding 'one more sensor'. This impracticality is exemplified in the following.

Figure 1(a) illustrates a mobile robot with a unique bumper sensor (x_1) such that in this case, the robot's sensory space is only composed of two states, i.e. when x_1 is *pressed* ($x_1 = 1$) or *not-pressed* ($x_1 = 0$). Figure 1(b) and 1(c) illustrate the same robot with added sensors. The sensory space formed by two sensors has four possible states ($\{0,0\}$,$\{0,1\}$,$\{1,0\}$,$\{1,1\}$), three sensors result in eight possible states, and so on. As it can be seen, the size of this sensory space grows exponentially as a function of the number of sensors, in this particular case, in the order 2^n for n sensors. The exponential growth of a hyperspace (e.g. sensor space) as a function of its dimensions (e.g. sensors) is known as the *curse of dimensionality* [1], and has direct effect on the computational complexity of a system using such spaces. For instance, a neural network with many inputs or dimensions will need a large quantity of resources (i.e., network size, training data, training time) to represent the resulting hyperspace.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 81–92, 2006.



(a) One bumper sensor (b) Two bumper sensors (c) Three bumper sensors

Fig. 1. Adding sensor capabilities

To alleviate the problems of dimensionality, a novel method based on the Q-analysis methodology is proposed, which discovers 'relevant sensor information, thus reducing the total number of dimensions by eliminating 'irrelevant' sensors. The process of discovering relevant sensors or features is known in the literature as *feature selection*. The main task of feature selection is: given a set of n features describing some entities, selecting a sub-set of m relevant features, where $m < n$ such that the sub-set provides the same or similar information about these entities [2]. Usually the relevance of a feature is measured in the context of classification, where relevant features are those which provide useful information for discriminating entities of different classes [2].

The method developed in this paper is demonstrated on three experiments, an experiment based on the CorrAL synthetic data-set, another based on Fisher's iris data-set and finally, one based on data from the RoboCup Simulation League. In particular, the experiments are focused on discovering the relevant sensors or features for classifying particular entities into predefined classes, i.e., feature relevance in relation to *supervised classification*. In this paper the terms feature and sensor are used synonymously.

The following section introduces some of the theoretical aspects of the Q-analysis methodology that are relevant to this paper.

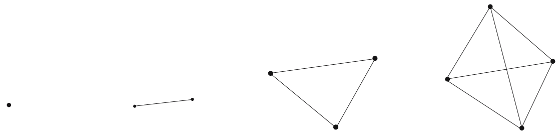
2 The Q-Analysis Methodology

Q-analysis is a multidimensional generalisation of network theory introduced by Atkin [3]. Q-analysis is able to model general *n-ary* relations between multidimensional data elements. This analysis is specially suited for discovering *relational structure* in the data.

2.1 Representation of Multidimensional Sensor Data

Let us assume that a robot has *p-active* binary sensors. For example, p pressed bumper sensors or *p-active* camera pixels. Then in Q-analysis, a relation between the set of these active sensors $\{x_1, x_2, ..., x_p\}$, can be considered to determine a new object called a *simplex*, denoted by: $\sigma = \langle x_1, x_2, \ldots, x_p \rangle$.

Simplices can be represented by *polyhedra* in multidimensional spaces. Let an individual, x_i, be called a *vertex*, denoted by $\langle x_i \rangle$. Then a simplex with one vertex is a point (Figure 2(a)), a simplex with two vertices is a line (Figure 2(b)), a simplex with three vertices is triangle (Figure 2(c)), a simplex with



(a) Point (b) Line (c) Triangle (d) Tetrahedron

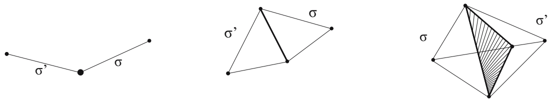
Fig. 2. Simplices

four vertices is a tetrahedron (Figure 2(d)), and so on. In general, a simplex with $p + 1$ vertices is a p-dimensional object and is referred to as a *p-simplex*. Following the previous example, a robot with only one active or pressed bumper is represented by a point, a robot with two active bumpers is represented by a line, three activated bumpers are represented by a triangle, and so on.

Typical robotic sensors provide their responses in continuous ranges (eg. a luminosity sensor which provides a response from 0 to 255 proportional to the luminosity it receives). The assumption that sensors are binary and represented by simplices can be made without loss of generality, as the method presented here can also deal with continuous sensors. The next section explains how Q-analysis defines structural relationships in the robot's sensory information using its simplex representation.

2.2 Multidimensional Data Relationships in Q-Analysis

Let the intersection of two simplices be defined as their shared *faces*. For example, Figure 3 illustrates some shared faces between simplices. Figure 3(a) illustrates two 1-simplices sharing a face which is a point. Figure 3(b) illustrates two 2-simplices sharing a line. Figure 3(c) illustrates two 3-simplices sharing a triangular face. An important notion in Q-analysis is that of *q-nearness*. Two simplices σ and σ' are said to be q-near if their shared face has at least q dimensions. Thus, the simplices in Figures 3(a), 3(b) and 3(c) are 0-near, 1-near and 2-near, respectively.



(a) 0-connected (b) 1-connected (c) 2-connected

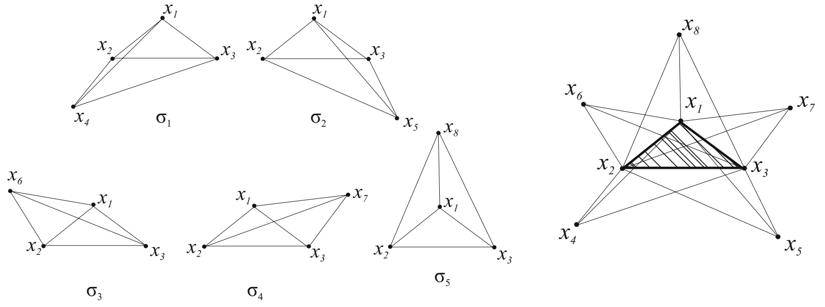
Fig. 3. q-connected simplices

A fundamental hypothesis is that q-nearness is a measure of structural similarity between simplices, i.e., simplices that share faces are considered similar. Thus q-nearness can be used as the basis for association or classification of multidimensional data. In our example, simplices represent active sensors, so that q-nearness is a measure of the structural similarity of a robot's active sensory inputs. Hence, two situations which activate a set of similar sensors will result in q-near simplices.

The notion of q-nearness alone does not solve the problem of feature relevance, as two simplices could be q-near by sharing irrelevant, redundant or even noisy features (sensors). The next section introduces the idea of a *hub*, which generalises the notion of q-nearness between a pair of simplices into a set of simplices.

2.3 Hubs and Stars

Given a set of simplices, $\sigma_1, \sigma_2, \ldots, \sigma_n$, their *hub* is their largest shared face [4]. Thus, $\textbf{hub}(\sigma_1, \sigma_2, \ldots, \sigma_n) = \cap_{i=1}^{n} \sigma_i$. Figure 4(a) illustrates five simplices, $\sigma_1, \sigma_2, \ldots, \sigma_5$, representing five different active sensor configurations. As can be seen, although the five configurations are different, they all share, $\langle x_1, x_2, x_3 \rangle$, which is their hub. Figure 4(b) illustrates this hub (grey triangle). Any simplex sharing a given hub is known as being part of the hub's *star*.



(a) Simplices $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ and σ_5 (b) hub($\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5$)

Fig. 4. Simplices and their hub

As seen previously, q-nearness between simplices provides some measure of their structural similarity; similarly hubs provide the structural similarity of a set of simplices.

Consider that the hub shared by the five previous simplices is in some sense 'relevant' in relation to a class. Then these simplices could be representing five elements of the same class. For example, a class *bird* could have the following relevant hub: $\langle alive, wings, non\text{-}mammal, vertebrate \rangle$, such that any animal sharing these essential characteristics would be classified as a *bird*, irrespective of the value of other possible features, such as colour, size or weight.

Relevant hubs are known in here as *classifier hubs*. Simplices sharing the same classifier hub are considered to be of the same class, even if they don't share *all of their faces*. This last remark is important, as the faces which are not contained in the classifier hub are considered 'irrelevant' for that particular class. The problem of identifying relevant features, is then the problem of finding classifier hubs from a set of simplices.

3 Simple Classification Using Classifier Hubs

This section presents a simple example of a classification task using classifier hubs. For this example, the CorrAL data-set has been used [2] which contains one target class and uses six binary features to describe it: ($A0$, $A1$, $B0$, $B1$, I, C). The target class is $(A0 \wedge A1) \vee (B0 \wedge B1)$. Thus ($A0$, $A1$, $B0$, $B1$) are relevant features, I is an irrelevant random feature, and C is correlated to the target class 75% of the time. In total, the data-set contains 40 instances, 20 of which are of the target class.

3.1 Star-Hub Analysis

The star-hub analysis takes each one of the data instances represented as simplices, and searches for their shared hubs. Broadly speaking, this is done by intersecting the simplices and finding their shared vertices. Each hub has an associated value that indicates the number of simplices of each class that share the hub. Table 1 illustrates some of the hubs from the CorrAL data-set ordered by their probability of occurrence.

Table 1. Some hubs from the CorrAL data-set

hub	#target	#¬target	total	specificity	broadness
$\langle A1\rangle$	16	8	24	16/24	24/40
$\langle C\rangle$	15	5	20	15/20	20/40
$\langle A0\rangle$	12	6	18	12/18	18/40
$\langle A0, A1\rangle$	10	0	10	10/10	10/40
$\langle B0, B1\rangle$	10	0	10	10/10	10/40
$\langle A1, B0, C\rangle$	8	1	9	8/9	9/40
$\langle A0, A1, I, C\rangle$	3	0	3	3/3	3/40

For example, the hub $\langle A1\rangle$ is the hub with highest probability of occurrence. $\langle A1\rangle$ contains 24 simplices, 16 of which belong to the target class and 8 which don't. The table also illustrates two statistical measures, *specificity* and *broadness*. The *specificity* of a hub is the maximum class conditional probability given a hub, i.e., the maximum probability of any simplex being of class C when it shares hub H, $P(C|H)$. For example, the probability of a simplex being of the target class given the hub $\langle A1\rangle$ is 16/24. The *broadness* of a hub is its probability of occurrence. There are three important characteristics to observe from Table 1. (i) In general, hubs of higher q-dimension are shared by fewer simplices than hubs with lower dimension. This is because high dimension hubs pose more requirements (number of vertices) to be satisfied, and fewer simplices satisfy them. (ii) Some hubs contain only simplices related to a unique class (specificity = 1). For example, hub $\langle A0, A1\rangle$ is shared by 10 simplices, and all of them are of the target class. In principle, such hubs are good for classification, as a simplex sharing this hub has a high probability of belonging to the target class. (iii) Hubs with few vertices and low specificity, eg. the hub $\langle A1 \rangle$ has a specificity of $16/24 \approx 0.66$, when taken in combination with other vertices becomes more specific, eg. $A1$ taken in combination with $A0$, $\langle A0, A1 \rangle$, has a specificity of 1. This means that a vertex could be irrelevant for classification (having low specificity) when taken individually, but become relevant when is combined with other vertices.

3.2 Heuristic Selection of Classifier Hubs

The total number of hubs found in a data-set is potentially very large, eg. the small CorrAL data-set contains approximately 60 different hubs. Moreover, not all these hubs are interesting for classification, as many of them represent the connection of different classes through irrelevant or noisy features. For example, hub $\langle A1 \rangle$ is shared by the 16 target and 8 no-target classes, thus providing no clear-cut distinction between the classes.

Ideally, a small set of classifier hubs must be selected from the total set of hubs. To this end a heuristic method to select classifier is presented. The method operates as follows: (i) hubs are ordered by their broadness, that is starting from the hubs with highest probability of occurrence. (ii) Starting from the broadest hub, a hub is selected as a classifier if its specificity is higher than a threshold, and if it is shared by a minimum number of simplices. As the CorrAL is noise-free, the specificity threshold is set to 1; this means that only classifier hubs with a 100% class-conditional probability can be selected. In robotic applications, where data are noisy, the threshold will have to be lower than 1 (accepting classifier hubs that contain elements of different classes) and manually selected by the designer.

Following this heuristic, only two classifier hubs are selected to represent the target concept; these are $\langle A0, A1 \rangle$ and $\langle B0, B1 \rangle$. These two classifiers represent the correct target function (A0 $\wedge$ A1) $\vee$ (B0 $\wedge$ B1) as any simplex will be considered of the target class if it shares any of the two classifiers, $\langle A0, A1 \rangle$ or $\langle B0, B1 \rangle$. It is important to observe that these classifiers do not contain any of the irrelevant or correlated features (I,C) initially present in the data. These have been filtered as 'irrelevant' by the heuristic method. To extend these results, two more experiments are described in the following sections.

4 Experiments with the Iris Data-Set

In this experiment, the popular Fisher's *iris* data has been used. This data-set contains four measurements, the sepal and petal sizes (sepal width, sepal length, petal width and petal length) of three different types of plant, *setosa*, *versicolor* and *virginica*. The complete data-set contains 50 instances of each plant. The task consists in classifying each plant on the basis of their sepal and petal sizes. Figure 5 illustrates the iris data-set, where the vertical axis represents the sepal and petal sizes in centimetres, and the horizontal axis represents the plant instances ordered as follows. From 1 to 50 setosa, from 51 to 100 versicolor, and 101 to 150 virginica.

Before classifying, the complete iris data-set is divided into training and test data-sets. Each of these contains half of the complete data, i.e., 75 instances of

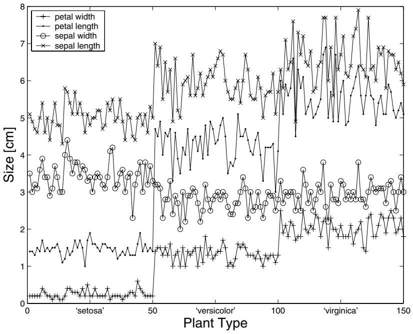


Fig. 5. Iris data

plants, 25 of each class. The aim of this experiment is three-fold: to (i) find the classifier hubs for the training data-set, (ii) measure the classifier's accuracy in classifying the unseen plants of the test data-set, and (iii) study the relevance of the classifier's features.

4.1 Simplex Representation of Continuous Features

To represent the continuous features (see Figure 5) using binary values, each feature has been first normalised to a 0 to 1 range and then segmented into 10 equal intervals. This segmentation results in each continuous feature having 10 possible binary features. Let us refer to each of these binary features as: sw_i (sepal width), sl_i (sepal length), pw_i (petal width) and pl_i (petal length); each with $i = 1, 2, ...10$. Of the total of 40 binary features, only four will be present for any given plant. In other words, each plant will be represented by a 3-simplex.

4.2 Star-Hub Analysis and Classifier Hubs

The star-hub analysis is used on the iris training data to discover its hubs. A total of 526 different hubs are found. The heuristic method presented in Section 3.2 is now applied to this set of hubs to find classifier hubs. In this case, although the data contains random fluctuations and noise (see Figure 5) the specificity threshold was set to 1, thus only the hubs related to a unique class are considered as possible classifiers. Table 2 illustrates the resulting classifier hubs.

4.3 Results in Classification Using Classifier Hubs

The test data-set is now used to measure the classification accuracy of the classifier hubs of Table 2. A plant is of a certain class if it shares a classifier hub related to that class. For example, if a plant shares hub $\langle pl_1 \rangle$, it will be classified as class setosa. Using the previous classifiers on the iris test data provides a 100% accuracy on classifying instances of setosa, 80% on the versicolor and 80% on the virginica. The remaining 20% for both versicolor and virginica are unclassified instances, as their simplices do not share any of the existing classifier hubs. These results indicate that a small number of classifier hubs, 12 in total, can be used to classify with acceptable accuracy.

Table 2. Classifier hubs from the iris data

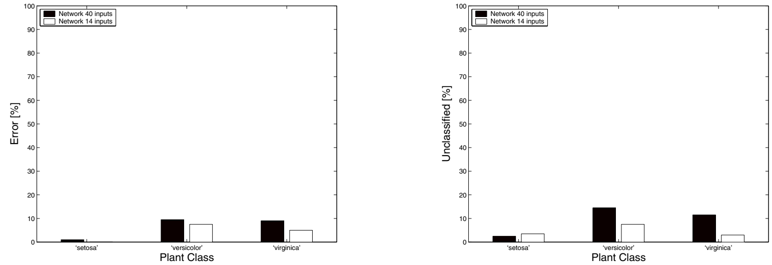
setosa			vesicolor			virginica		
classifier hub	spec	broad	classifier hub	spec	broad	classifier hub	spec	broad
$\langle pl_1 \rangle$	20/20	20/75	$\langle pl_5 \rangle$	8/8	8/75	$\langle pw_8 \rangle$	7/7	7/75
$\langle pw_1 \rangle$	19/19	19/75	$\langle pw_5 \rangle$	4/4	4/75	$\langle pw_9 \rangle$	6/6	6/75
$\langle pw_2 \rangle$	6/6	6/75	$\langle pl_4 \rangle$	4/4	4/75	$\langle pl_9 \rangle$	6/6	6/75
			$\langle sl_5, pl_6 \rangle$	6/6	6/75	$\langle pl_{10} \rangle$	6/6	6/75
						$\langle sl_3, pl_8 \rangle$	5/5	5/75

4.4 Study of Feature Relevance

In the experiment presented in Section 3 it was easy to validate whether the features composing the classifier hubs were relevant or not, as the data-set itself defined which features were relevant, irrelevant and correlated. In the iris data-set there is no straightforward definition of which features are relevant and which are not. Thus, in order to study the features' relevance the following experiment is conducted.

Two multilayer neural networks are used to classify the iris data, one using the complete set of features and the other using only the features that compose the classifier hubs of Table 2. That is, the first network takes as input the 40 features described in Section 4.1. The second network uses only 14 features, i.e. $(pl_1, pw_1, pw_2, pl_5, pw_5, pl_4, sl_5, pl_6, pw_8, pw_9, pl_9, pl_{10}, sl_3, pl_8)$. Both networks have 3 hidden neurons, 3 outputs, and are trained using backpropagation with 10 training instances of each class of plant.

The trained networks are tested five times against the remaining plant instances, and the average of their classification error is measured. The results of the classification are summarised in Figure 6. In black is the result for the 40-input network, in white is the result of the 14-input network. Figure 6(a) illustrates the misclassification error and Figure 6(b) illustrates the unclassified error. These show that the 14-input network results in a slightly better classification accuracy. Although this accuracy is not significant to decide that the 14-input network outperforms the 40-input network, it can be said that both networks



(a) Classification error

(b) Unclassified error

Fig. 6. Neural network classification error

have an almost similar accuracy. Thus, removing what the heuristic considered as irrelevant features does not decrease the network's prediction accuracy. This implies that those features were irrelevant in the first place.

5 Experiments with the RoboCup Data-Set

One of the important behaviours in robot soccer is that of ball-passing. If the player with the ball decides to pass, it must also decide which is the 'appropriate' team-mate to pass to. In order to make this decision, the passer evaluates many features. For example, it could measure the positions, distances, speeds, direction of movements of each player, including features of low relevance such as the players' names. Now, the question is to find which of these features are most relevant for a successful pass.

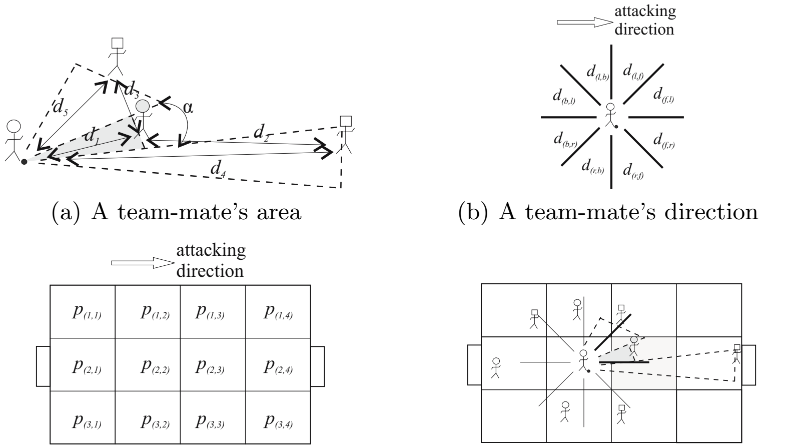
To investigate this question, the following experiment applies the previous method to a data-set extracted from the RoboCup 2003 Simulation League final game. The data-set is composed of the successful passes in the game. For each pass, the data-set indicates some arbitrary features that the passer sensed with respect to the receiver as well as with respect to the remaining team-mates [5], in other words, the state of the receiver and the non-receiver team-mates as "seen" from the passer's perspective. The next section describes these features.

5.1 Simplex Representation of the Pass Data-Set

The state of each team-mate player (receiver or not) is described by fifty binary features:

$$\{d_1, \ldots, d_5, \alpha, RN, \bar{RN}, LN, \bar{LN}, OPP, \bar{OPP}, d_{(f,l)}, \ldots, d_{(b,l)}, p_{(1,2)}, \ldots, p_{(3,4)}\}$$

Some are illustrated in Figure 7. Figure 7(a) illustrates a set of features related to the players's 'controlled area'. This area is defined by the distances $(d_1, \ldots, d_5)$ and the angle (α) between the four players seen in the figure, namely, the passer, the team-mate in focus (in grey) and its two neighbouring players, which may or may not be of the same team. These features have four possible states indicating their values ('very small', 'small', 'big' or 'very big'). Figure 7(b) illustrates eight directions $(d_{(f,l)}, \ldots, d_{(b,l)})$ in which the team-mate player could be located, these are defined from the passer's perspective and the attacking direction. Figure 7(c) illustrates the playing field divided into twelve positions $(p_{(1,2)}, \ldots, p_{(3,4)})$, in which the team-mate player could be located. These positions are also relative to the attacking direction. Figure 7(d) illustrates how the previous features are measured with respect to the team-mate player (in grey). The non-illustrated features, i.e., RN, $\bar{RN}$, LN, $\bar{LN}$, OPP and $\bar{OPP}$, have the following significance. The response of 'right neighbour' $r(RN)$ and 'left neighbour' $r(LN)$ is 1 if the neighbouring players (see Figure 7(a)) of the player in focus are also team-mates. The response of 'opponent closer' $r(\bar{OPP})$ is 1 if a neighbouring players is an opponent, and is closer to the ball. The features $\bar{RN}$, $\bar{LN}$ and $\bar{OPP}$ are the opposite of the previous.



(a) A team-mate's area (b) A team-mate's direction

(c) A team-mate's position (d) Team-mate state representation

Fig. 7. State representation

Of these fifty features only eleven are active for any given pass, because some of the features are exclusive. For example, if d_1 is 'very small' it can not be any of the other remaining values ('small', 'big', 'very big'). Given this definition of the state of a team-mate player, each pass determines a 10-simplex with respect to each player, that is, one simplex representing the receiver's state, labelled as *receiver*; and nine simplices representing the remaining team-mates' state, labelled as *non-receiver*. The pass data-set contains 1062 'non-receiver' and 118 'receiver' instances.

The next section applies the star-hub analysis to investigate if there exists any structural difference between the simplices representing 'receivers' and those representing 'non-receivers'.

5.2 Star-Hub Analysis

The star-hub analysis applied to the pass data-set produces over 53000 hubs. From these, a selection is presented in Table 3. As can be seen in the table, there are some hubs that occur more in relation to 'receiver' players than in relation to 'non-receiver' ones. This indicates that some structural differences exist between the state of 'receiver' and the 'non-receiver' players, as also shown in [5].

The two last hubs in the table show a similar probability of occurrence in both classes. Thus, in principle, if these features do not appear as relevant in other hubs, they could be considered as irrelevant for these classes.

A complete analysis, including the heuristic method for finding classifier hubs, is not applicable in this experiment. The reasons are: (i) a relatively small data-set, containing 118 'receiver' instances and 1062 'non-receiver' instances, results in a huge number of hubs, approximately 58000. In such a large hub space, most hubs are only shared by two simplices. Thus they do not provide reliable

Table 3. A selection of hubs from the pass data

hub	receiver	non-receiver
$\langle \bar{RN}, \bar{OPP} \rangle$	43%	24%
$\langle d_1(s), \bar{LN} \rangle$	36%	15%
$\langle \bar{RN}, \bar{LN}, \bar{OPP} \rangle$	24%	10%
$\langle d_1(s), \bar{RN}, \bar{LN} \rangle$	20%	9%
$\langle d_1(s), d_5(vb), \alpha(vs), \bar{OPP} \rangle$	15%	4%
$\langle d_1(vs), \bar{RN}, LN, \bar{OPP} \rangle$	8%	2%
$\langle d_2(s), d_3(s) \rangle$	13%	10%
$\langle d_3(s), d_4(b) \rangle$	9%	7%

statistics of 'specificity' and 'broadness'. (ii) Some of the players labelled as 'non-receivers' share similar features to those labelled as 'receivers'. This is because it is possible for many team-mates to be in a good passing states, but only one of them can become a 'receiver'.

6 Conclusions

This paper has introduced feature selection methods as a possible solution to the problems related to dimensionality. Feature selection methods were defined in the context of classification.

The Q-analysis methodology was described, and a method for supervised classification based on it was presented. The method consisted of representing training instances as simplices and applying the star-hub analysis to discover the hubs in the training data. Then, a heuristic method was used to discover classifier hubs. Other well-known supervised classification methods exist, such as inductive trees, which use different criteria for selecting the relevance of the features. Thus, it will be necessary to extend the experiments in this paper to evaluate and contrast different supervised classification methods.

The main idea behind classification using classifier hubs, is that if a set of simplices share 'relevant' vertices or features (i.e. the classifier hub), then these simplices can be considered to belong to the same class. By definition, the vertices or features not contained in the set of classifier hubs are 'irrelevant' for the classification. Thus, discovering (ir)relevant features is seen as the task of finding classifier hubs.

Two experiments, one based on the CorrAL data-set and one based on the iris data-set were used to validate experimentally the classification method. The results from those experiments indicate that it is possible to classify using classifier hubs; and that eliminating the features not contained in the set of classifiers did not deteriorate the classification accuracy. Thus, those features were irrelevant, and were correctly identified as such by the heuristic method.

A third experiment based on data from the RoboCup Simulation League was also proposed. Although the results from this last experiment were not conclusive, the Q-analysis method was capable of finding structural differences between the classes in that data. These are promising results, but require further heuristics to alleviate the combinatorial explosion of having to compute the complete set of hubs.

References

1. Bellman, R. E.:Dynamic Programming. (1957)
2. Dash, M. and Liu, H.: Feature Selection for Classification. Intelligent Data Analysis **1** (1997) 131–156
3. Atkin, R.H.: Combinatorial connectivities in social systems. Basel: Birkhäuser Verlag. (1977)
4. Johnson, J.H.: Stars, Maximal Rectangles, Lattices: a new perspective on Q-analysis. International Journal of Man-Machine Studies **24** (1986) 293–299
5. Iravani, P., Johnson, J.H., Rapanotti, L.: Robotics and the Q-analysis of behaviour: International Symposium on Artificial Life and Robotics (2005)

Keepaway Soccer: From Machine Learning Testbed to Benchmark

Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188
{pstone, kuhlmann, mtaylor, yxliu}@cs.utexas.edu

Abstract. Keepaway soccer has been previously put forth as a *testbed* for machine learning. Although multiple researchers have used it successfully for machine learning experiments, doing so has required a good deal of domain expertise. This paper introduces a set of programs, tools, and resources designed to make the domain easily usable for experimentation without any prior knowledge of RoboCup or the Soccer Server. In addition, we report on new experiments in the Keepaway domain, along with performance results designed to be directly comparable with future experimental results. Combined, the new infrastructure and our concrete demonstration of its use in comparative experiments elevate the domain to a machine learning *benchmark*, suitable for use by researchers across the field.

1 Introduction

Keepaway soccer in the Soccer Server used at RoboCup has been previously put forth as a testbed for machine learning [15]. Since then it has been used for research on temporal difference reinforcement learning with function approximation [16], evolutionary learning [12], relational reinforcement learning [20], and behavior transfer [18].

These successes notwithstanding, the domain has remained inaccessible to many potential users due to the considerable implementation effort required. In particular, though the *domain* itself is publically available, it has been necessary for each researcher to build up *agents* capable of following the rules of Keepaway and executing the required low-level behaviors such as intercepting a moving ball, passing the ball, and moving to open space. Due to the complexity of the simulator, building up such agents is no simple task. As a result, researchers who have no prior experience in RoboCup, or who are unwilling to invest considerable start-up effort, have not been able to use the testbed.

Even if willing to invest this effort, the resulting experiments are not directly comparable with previously published results. Because each experiment is done with different underlying agents, the performance results may vary more from inter-agent differences than from the learning algorithms under study.

These two barriers — inaccessibility to the non-domain-experts and incomparability of results across studies — have prevented the Keepaway *testbed* from

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 93–105, 2006.

serving as a *benchmark* for the machine learning community. This paper reports on new resources that elevate the Keepaway testbed to a benchmark problem for machine learning. In particular, it describes a repository that:

1. Provides standard, open source implementations for all aspects of the problem except the learning algorithm itself;
2. Provides a step-by-step tutorial for non-domain-experts to get up to speed easily; and
3. Provides the graphical tools necessary to evaluate progress.

In addition, we report concrete numerical results for several easily replicable approaches using the material from the repository. These numerical results are designed to be directly comparable with future experimental studies.

Finally, we illustrate the use of this benchmark by presenting an empirical study of different function approximators used within a temporal difference learning approach to the problem. Previous results indicated that using a form of linear tile-coding (a CMAC [1]) to approximate the value function led to results that were better than hand-coded approaches. However, it was not known to what extent the CMAC itself was responsible for these results. In this paper we directly compare the CMAC against other function approximators, including a variation of CMAC based on radial basis functions, as well as neural networks.

2 Background

This section introduces the Keepaway task, surveys previous examples of learning in this domain, and specifies a standardized learning scenario to be used as a machine learning benchmark.

2.1 The Keepaway Task

Keepaway is a subproblem of RoboCup simulated soccer in which one team, the *keepers*, tries to maintain possession of the ball within a limited region, while the opposing team, the *takers*, attempts to gain possession [15]. Whenever the takers take possession or the ball leaves the region, the *episode* ends and the players are reset for another episode (with the keepers being given possession of the ball again). Parameters of the task include the size of the region, the number of keepers, and the number of takers. Figure 1 shows a screen shot of an episode with 3 keepers and 2 takers (called 3 vs. 2, or 3v2 for short) playing in a $20m \times 20m$ region[1].

2.2 The Soccer Server

Since late 2002, the Keepaway task has been part of the official release of the open source Soccer Server used at RoboCup[2]. Agents in the simulator [11] receive visual perceptions every 150 *msec* indicating the relative distance and angle

[1] Flash files illustrating the task are available at `http://www.cs.utexas.edu/~AustinVilla/sim/keepaway/`

[2] Starting with version 9.1.0.

to visible objects in the world, such as the ball and other agents. They may execute a primitive, parameterized action such as `turn`(*angle*), `dash`(*power*), or `kick`(*power*,*angle*) every 100 *msec*. Thus the agents must sense and act asynchronously. Random noise is injected into all sensations and actions. Individual agents must be controlled by separate processes, with no inter-agent communication permitted other than via the simulator itself, which enforces communication bandwidth and range constraints. Full details of the simulator are presented in the server manual [6].

When started in a special mode, the simulator enforces the rules of the Keepaway task, as described above, instead of the rules of full soccer. In particular, the simulator places the players at their initial positions at the start of each episode and ends an episode when the ball leaves the play region or is taken away. In this mode, the simulator also informs the players when an episode has ended and produces a log file with the duration of each episode.

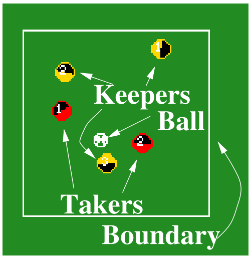


Fig. 1. A screen shot from the middle of a 3 vs. 2 Keepaway episode in a 20m x 20m region

2.3 Previous Studies

Although the Keepaway task has been available in the server for some time, it required knowledge of player development to be useful as a machine learning testbed. Nonetheless, there are a few examples of machine learning research involving RoboCup Keepaway.

In addition to our own previous work [16, 18], work by DiPietro et al. [12] applied evolutionary algorithms to train 3 keepers against 2 takers in the Soccer Server. Other work by Walker et al. [20] used relational reinforcement learning to learn the value function for a keeper coordinating with 2 "smart" teammates against 2 takers.

There has been additional previous work in Keepaway using simulators other than the Soccer Server. Whiteson et al. [21] used neuroevolution to train keepers in the SoccerBots domain [3], an extension of the more abstract TeamBots simulator [2]. Also, Hsu and Gustafson [9] evolved keepers for 3 vs. 1 Keepaway in the TeamBots simulator[3].

Clearly it is not possible to to directly compare performance of work using simulators with different primitive actions and different game dynamics. But even work within the same simulator cannot typically be compared directly because the approaches differ in their set of high-level behaviors, implementation of basic skills, fixed opponent policies, and sometimes even performance metrics. It is exactly this problem that our benchmark repository seeks to address.

[3] Gustafson has made some code available that contributes to this more abstract simulator having some of our defined properties of a benchmark: `http://www.cs.nott.ac.uk/~smg/kas/keepaway-v0.01.html`

2.4 Standardized Task

When Keepaway was introduced as a testbed [15], a standard task was defined. The main contribution of the current work is the infrastructure required to easily implement that task (Section 3). This section reviews the standardized task as previously formulated.

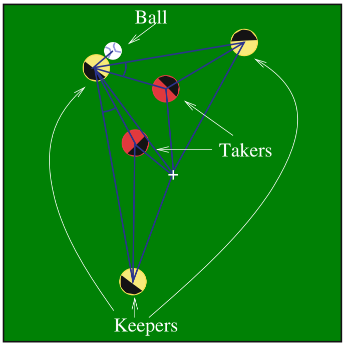


Fig. 2. This diagram depicts the 13 state variables used for learning with 3 keepers and 2 takers. There are 11 distances to players, the center of the field, and the ball, as well as 2 angles along passing lanes.

The Keepaway problem maps fairly directly onto the discrete-time, episodic, reinforcement-learning framework. As a way of incorporating domain knowledge, the learners choose not from the simulator's primitive actions but from a set of higher-level macro-actions implemented as part of the player. These macro-actions can last more than one time step, and the keepers have opportunities to make decisions only when an on-going macro-action terminates. To handle such situations, it is convenient to treat the problem as a semi-Markov decision process, or SMDP [13, 5]. The macro-actions used (and fully provided as a part of our repository) can be found in [16]. The agent can make decisions at discrete SMDP time steps, at which macro-actions are initiated and terminated.

For the purpose of defining a standardized task, we focus on training the keepers, but training the takers can be done similarly in the Keepaway domain, as suggested in [16]. The keepers learn in a constrained policy space. They have the freedom to decide which action to take only when in possession of the ball. A keeper in possession may either hold the ball or pass to one of its teammates. Keepers not in possession of the ball are required to execute the **Receive** macro-action in which the player who can get there the soonest goes to the ball and the remaining players try to get open for a pass.

For training the keepers, the behavior of the takers is "hard-wired" and relatively simple. The two takers that can get there the soonest go to the ball, while the remaining takers, if present, try to block open passing lanes. Similarly, for training the takers, the keepers may be hard-wired. These hard-wired behaviors are provided in the repository.

Also as a way of incoporating domain knowledge, the learners do not make decisions based on raw positional information but based on higher-level features that form the states for the Keepaway learning task. The keepers' states comprise distances and angles of the keepers $K_1, \ldots, K_n$, the takers $T_1, \ldots, T_m$, and the center of the playing region. Keepers and takers are ordered by increasing distance from the ball and when learning takes place, K_1 is always the keeper in possession of the ball. The 13 state variables for 3v2 Keepaway are (see Figure 2):

- Distances from players to center of region,
- Distances from other players to K_1,
- Distances from teammates to their closest opponent, and
- For each K_i ($i = 2, \ldots, n$), the minimal angle with the vertex at K_1 between K_i and an opponent.

We can easily vary the size of the Keepaway region, the number of keepers, and the number of takers to change the Keepaway task. Our framework provides a standard interface to the learner in terms of macro-actions, states, and rewards.

We choose episode duration as the performance measure for this task. The keepers attempt to maximize it while the the takers try to minimize it. To this end, it is natural to give the learners a constant positive reward for each time step an episode persists. Complete details on the task and the learning scenario can be found elsewhere [16].

3 Benchmark Repository

We have implemented a standardized Keepaway player framework in C++ and released its code base for public use in an online repository[4]. The code base provides an open source implementation for all aspects of the Keepaway problem except the learning algorithm itself, which is intended to be the object of study by each individual researcher.

3.1 Standardized Keepaway Player

We have created an implementation of a standardized player built upon the player framework developed by the UvA Trilearn team [8]. This framework handles communication and synchronization with the server, world model update, localization, and low- and mid-level skills.

On top of this framework, we added additional skills necessary for Keepaway and implemented the fixed policy carried out by the keepers when they do not have the ball. Also, we have implented hand-coded takers that follow the policy described previously.

By default, a player in the Soccer Server is only able to see objects in a 90-degree cone in front of them. A difficult problem in developing agents that are meant to coordinate in dynamic environments with limited vision is trying to maintain a correct distributed world model. This requires agents to decide where to look, what to communicate, whom to listen to, and how to incorporate second-hand information. Although we have previously demonstrated that it is possible to get learning to work in this scenario [10], the players do not perform at as high of a level. For this reason, the players operate with 360-degree

[4] `http://www.cs.utexas.edu/~AustinVilla/sim/keepaway/`

(but still noisy) vision by default. A rudimentary implementation of communication and information-gathering actions is included, but is not recommended. We hope that in future development of the players, the level of play in the 90-degree case can be increased such that the 360-degree assumption is no longer necessary.

Learning Agent Interface. The Keepaway player implementation is constructed in such a way that the details of the Keepaway domain are completely abstracted from the high-level action selection. This was done to allow new learning algorithms to be integrated into the players with minimal effort.

From the learning algorithm's perspective, the Keepaway problem is presented as a generic SMDP. The state is represented as a fixed-length vector of continuous values. A macro-action is represented as an integer ranging from 0 to $numActions - 1$. The reward is a single continuous value received after each macro-action terminates.

A new learning agent must implement the **SMDPagent** interface, which consists of the following three functions:

- **int startEpisode(double state[])**
  This function is called the first time this player has an action opportunity in an episode. In other words, it is called when the player first has possession of the ball. If the player never obtains the ball, this function will never be called. The agent is supplied with the current state and is expected to return a macro-action to be executed.
- **int step(double reward, double state[])**
  This function is called at every action opportunity for this player after the first one. The reward accumulated during the execution of the previous macro-action is given along with the new state. A macro-action is again expected to be returned.
- **void endEpisode(double reward)**
  This function is called when the player receives notice from the server that the episode has ended. The agent is supplied with the reward accumulated from the last macro-action up until the end of the episode. Note that this function is called after every episode, even when this player never touches the ball.

Although the players do not come with any learning code, a few fixed policies are supplied:

- **Random** - choose actions uniformly at random
- **Always Hold** - always choose the hold action
- **Hand-coded** - a simple handcoded policy that holds the ball when no takers are nearby and passes to the most open teammate otherwise. More details can be found in [16].

The known performance of these policies as reported later in this paper (see Section 4) serve as a sanity check for new installations of the system.

3.2 Tools

The player source code package comes with a set of scripts that are helpful in running experiments and analyzing the resulting data. There are scripts to start and stop the simulation. Another script can be executed during the simulation to launch the Soccer Monitor, the standard visualization tool for the simulator, in a special mode that displays the Keepaway region on field. Additional scripts use the known structure of the `.kwy` log files to compress and decompress them much more compactly than using a standard compression utility.

One of the most useful tools is a script that converts the episode duration data in a `.kwy` file into a representation that is appropriate for generating a learning curve using a tool such as *gnuplot*. This script uses a "sliding window" to find average episode durations for each fixed-sized sequence of episodes. Along with the size of the window (number of episodes), the script takes in an additional parameter that specifies the coefficient of a low-pass filter used for smoothing the curve. A gnuplot style file is also supplied to produce graphs similar to the ones included in this paper (e.g. Figure 4).

Finally, there is a tool to generate histograms of episode durations. The intended purpose of this tool is to allow someone to visualize the distribution of episode durations when evaluating a fixed policy. Again, a gnuplot style file is included for generating figures appropriate for publication.

3.3 Online Tutorials

In addition to source code itself, the web site repository contains a step-by-step tutorial of how to use the code. The goal of the tutorial is to allow for someone who is not an expert in the RoboCup simulated soccer domain to get up to speed easily.

The tutorial is divided into two sections. The first section walks through the necessary steps for downloading and installing the simulator and players, starting a simulation using one of the supplied hand-coded policies, and generating a learning curve. Two graduate students from our lab that had never worked in the Keepaway domain before were able to successfully complete this section in a matter of minutes.

The second part of the tutorial discusses how to incorporate a new learning algorithm into the provided player source code. We include skeleton code for download from within this section of the web site that can serve as the starting point for a new learning agent implementation. Thus, the main effort required on the part of a new user is exactly the porting of one's own learning approach to the place-holders within the provided source code.

4 An Empirical Study

Though many learning approaches are possible in this domain, we now consider a particular learning approach to learning Keepaway for the keepers. The keepers learn their task using episodic SMDP Sarsa(λ) [17, 16], a well-understood temporal difference algorithm and naturally fit into the **SMDPagent** interface.

The state variables are continuous and therefore suggest value function approximation. We consider episodic SMDP Sarsa(λ) but with different function approximators. A function approximator in a Sarsa(λ) learner maps states to a vector of state-action values, one entry for each action, and the Sarsa(λ) learner uses the state-action values to perform on-policy learning.

Within this framework, a question that arises is the efficacy of different function approximators. By using an implementation of the SMDPagent interface we are able to easily substitute different function approximators, a key component of a value-based RL learning. In this section we compare three such function approximators, CMACs, a novel extension to CMACs using radial basis functions which we denote RBF networks, and neural networks, in the Sarsa approach to the Keepaway task. The training performs a version of Sarsa using function approximators [14], with adaptations for SMDPs.

In order to quantify the learning rates of different learning algorithms and function approximators, we analyze the `.kwy` files produced by the Soccer Server in Keepaway mode. To produce a learning curve we "window" the data so that every point on a learning graph is the average of 1000 Keepaway episodes. The noise in the sensors and actuators is large enough that the variance between different episodes is large, even within a single trial using a static policy. By averaging episodes over time we are able to reduce much of the noise in our graph and still show representative curves.

Table 1. This table details the average possession time and standard deviation in seconds for three simple policies included in our distributed code on three different Keepaway tasks. 3 vs. 2 is run on a 20m x 20m field, 3 vs. 4 is run on a 25m x 25m field, and 5 vs. 4 is run on a 30m x 30m field. These numbers may be used as benchmarks to be compared against other learned policies.

Static Keepaway Policies

Policy	3 vs. 2	4 vs. 3	5 vs. 4
Always Hold	**3.4**±1.5	**4.1**±1.8	**4.8**±2.2
Random	**7.5**±3.7	**8.3**±4.4	**9.5**±5.1
Hand-coded	**8.3**±4.7	**9.2**±5.2	**10.8**±6.7

The possession times for the three static policies provided can be found in Table 1. Note that the standard deviation is a measure of the difference of windowed averages across trials, not a measure of the variation in episode lengths within a single trial. We will see that both function approximators examined allow players to learn an average possession time better than these three static policies. While past research [16] has shown that CMAC function approximation allows learning in the Keepaway domain, this is the first research showing comparable, or perhaps better, learning results, in this case using the new RBF function approximator.

4.1 CMAC Function Approximation

CMACs are a form of linear tile-coding function approximation that have been successfully used in many reinforcement learning systems [1]. CMACs allow us to take arbitrary groups of continuous state variables and lay infinite, axis-parallel tilings over them (see Figure 3). Using this method we are able to discretize the

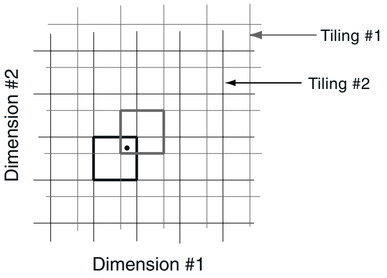


Fig. 3. The tile-coding feature sets are formed from multiple overlapping tilings. The state variables are used to determine the activated tile in each of the different tilings. Every activated tile then contributes a weighted value to the total output of the CMAC for the given state. Increasing the number of tilings allows the tile-coding to generalize better while decreasing the tile size allows more accurate representations of smaller details. Note that we primarily use one-dimensional tilings but that the principles apply in the n-dimensional case.

continuous state space by using tilings while maintaining the capability to generalize via multiple overlapping tilings. The number of tiles and width of the tilings are hardcoded and this dictates which state values will activate which tiles. The function approximation is learned by changing how much each tile contributes to the output of the function approximator. By default, all the CMAC's weights are initialized to zero. This approach to function approximation in the RoboCup soccer domain is detailed by Stone et al. [16].

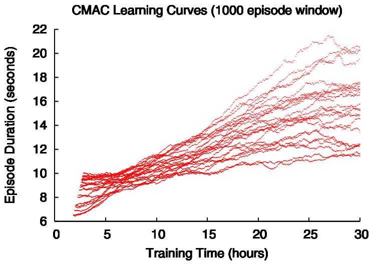


Fig. 4. This figure presents the learning curves for 24 independent 3 vs. 2 trials using a CMAC function approximator

CMACs have been used previously in the Keepaway domain [16]. We include results here as a point of comparison. In Figure 4 we see that the keepers learn to increase the time they are able to control the ball through training. The average learned possession time over 24 trials after 30 simulator hours of training is 15.69 seconds with a standard deviation of 2.81 seconds. Again, the average possession time for each trial is averaged over 1000 episodes to reduce the impact of noise.

4.2 RBF Function Approximation

A radial basis function (RBF) is a generalization of the tile coding idea to a continuous function [17]. In the one-dimensional case, an RBF approximator is a linear function approximator $\hat{f}(x) = \sum_i w_i f_i(x)$, where the basis functions have the form $f_i(x) = \phi(|x - c_i|)$, x is the current state, and c_i is the center for feature i. A CMAC is a special type of RBF approximator with c_i's equally spaced and $\phi(x)$ a step function. Here we use Gaussian radial basis functions, where $\phi(x) = \exp(-\frac{x^2}{2\sigma^2})$, and the same c_i's as a CMAC. The learning for RBF networks are identical to that for CMACs except for the calculation of state-action values where the RBFs are used. As is the case for CMACs, the state-action values are computed as a sum of one-dimensional RBFs, one for each feature. By tuning σ, the experimenter can control the width of the Gaussian function and therefore the amount of generalization over the state space. In our implementation, a value of $\sigma = 1.0$ creates a Gaussian which roughly spans 9 CMAC tiles, a value of $\sigma = 0.5$ spans 5 tiles, and $\sigma = 0.25$ activates roughly 3 tiles. We found the value of $\sigma = 0.25$ to perform the best, but more tuning would possibly produced better results than reported here.

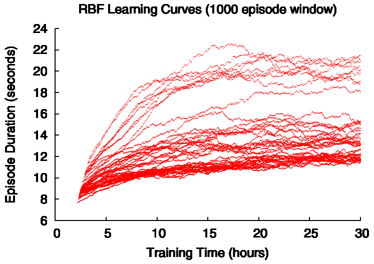


Fig. 5. This figure presents the learning curves for 40 independent 3 vs. 2 trials using an RBF function approximator

In Figure 5 we see keepers can successfully learn to keep the ball with an RBF function approximator for successively longer periods of time after training. The average learned possession time over 40 trials after 30 simulator hours of training is 14.23 seconds with a standard deviation of 3.14 seconds.

By comparing Figures 4 and 5, we note that the RBF trials appear to be learning faster: they have better performance between the 5-hour and 25-hour marks (though not significantly different average performance). The best-case performance of RBF at the 30-hour mark is comparable to that of CMAC, but more RBF trials land at the worst-case end.

4.3 Neural Network Function Approximation

Feedforward neural networks are a very popular type of function approximator and have had some notable past successes in reinforcement learning [7, 19]. We use three seperate 13-20-1 networks, one for each action. Inputs to the neural network are set to the value of state variables and each network's output corresponds to an action. Nodes in the hidden layer have a sigmoid transfer function and output nodes are linear. We use standard backpropagation to modify the weights in the neural networks to back up Sarsa(λ) values.

In Figure 6 we see keepers also learn to keep the ball with a neural network function approximator for longer periods of time after training. The average learned possession time over 20 trials after 30 simulator hours of training is 10.13 seconds with a standard deviation of 0.29 seconds.

By comparing Figure 6 to Figures 4 and 5, we note that neural networks do not learn as fast as CMACs or RBFs, but have a lower variance. However, we did not try to optimize the neural networks' parameters and ran fewer experiments

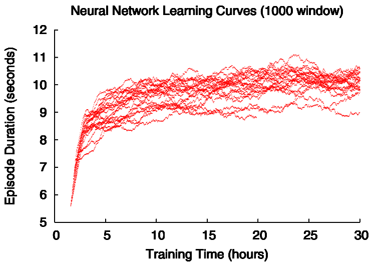


Fig. 6. This figure presents the learning curves for 20 independent 3 vs. 2 trials using a neural network function approximator

relative to the CMACs and RBFs. Neural networks can theoretically have better performance: they can encode arbitrarily complex interactions among state variables while CMACs and RBFs in our implementation only combine effects of different state variables in a simple way (summation).

5 Future Work

In the Section 4 we give an example study comparing the efficacy of two different function approximators using the Keepaway benchmark. Other function approximators and learning algorithms can be easily inserted into the testbed and directly compared to these two function approximators, allowing experimenters to quickly test the relative benefits of different techniques.

The infrastructure is of course not limited to function approximation studies. Different reinforcement learning algorithms, both value-based and policy search, may be compared. Evolutionary methods (as in [12]) may be evaluated and compared directly to temporal difference methods; different temporal difference methods may be directly compared (such as Sarsa vs. Q-learning as in [16]). Multi-agent learning questions may be addressed, such as how learning rates are affected when opponents or teammates learn simultaneously with an agent, learn sequentially, or do not learn at all. Different state representations can be easily expressed, enabling the investigation of state abstraction. Alternate actions can be implemented, allowing the investigation of hierarchical RL questions such as using options to speed up learning. We anticipate exploring some of these areas in the future using this infrastructure.

Our Keepaway benchmark infrastructure provides an easy way of consistently comparing the relative performance of different methods when investigating any of the previously mentioned research questions. By using the same benchmark platform, researchers will be able to make quantitative comparisons between different learning methods. To that end we will make every effort to not make changes to the infrastructure that affect performance, thus making direct comparisons across versions of our implementation legitimate. However, we intend to make improvements to the communication and distributed sensing code for use with limited vision. Because of this and other possible changes, when reporting results, we encourage researchers to always cite the implementation version number as well as any non-default settings that were used. Doing so will ensure the validity of direct comparisons as well as enable the repeatability of all experiments.

6 Conclusion

This paper makes two main contributions. First and foremost, it introduces a source code repository including a set of tools and tutorials designed to enable all machine learning researchers to use the Keepaway soccer domain as a benchmark task. Second, it introduces a function approximation method not previously tested in this domain and empirically evaluates it on this benchmark task. The RBF network performs at least as well as, and perhaps a little better than, the previously-used CMAC method.

While there is an established repository for supervised machine learning benchmarks [4], there is currently no comparable repository of sequential decision-making tasks. For this reason, releasing our new benchmark provides a valuable service. We plan to continue maintaining the repository at `http://www.cs.utexas.edu/~AustinVilla/sim/keepaway/` for the foreseeable future. It is our hope that it will be a benefit to both the RoboCup and the machine learning communities. For people already involved in RoboCup, it standarizes a benchmark for machine learning research within the domain, and could serve as a domain of interest for the Special Interest Group (SIG) on multiagent learning. [5] For the machine learning community it makes experimentation within the RoboCup domain accessible to everyone.

Acknowledgments

This research was supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

1. J. S. Albus. *Brains, Behavior, and Robotics.* Byte Books, Peterborough, NH, 1981.
2. Tucker Balch. Teambots, 2000. `http://www.teambots.org`.
3. Tucker Balch. Teambots domain: Soccerbots, 2000. `http://www-2.cs.cmu.edu/~trb/TeamBots/Domains/SoccerBots`.
4. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
5. S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In T. Leen G. Tesauro, D. Touretzky, editor, *Advances in Neural Information Processing Systems 7*, pages 393–400, San Mateo, CA, 1995. Morgan Kaufmann.
6. Mao Chen, Ehsan Foroughi, Fredrik Heintz, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Patrick Riley, Timo Steffens, Yi Wang, and Xiang Yin. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later, 2003. Available at `http://sourceforge.net/projects/sserver/`.

[5] `http://sserver.sourceforge.net/SIG-learn/`

7. Robert H. Crites and Andrew G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1996. MIT Press.
8. Remco de Boer and Jelle R. Kok. The incremental development of a synthetic multi-agent system: The uva trilearn 2001 robotic soccer simulation team. Master's thesis, University of Amsterdam, The Netherlands, February 2002.
9. W. H. Hsu and S. M. Gustafson. Genetic programming and multi-agent layered learning by reinforcements. In *Genetic and Evolutionary Computation Conference*, New York,NY, July 2002.
10. Gregory Kuhlmann and Peter Stone. Progress in learning 3 vs. 2 keepaway. In *Proceedings of the RoboCup-2003 Symposium*, Padova, Italy, July 2003.
11. Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
12. Anthony Di Pietro, Lyndon While, and Luigi Barone. Learning in RoboCup keepaway using evolutionary algorithms. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1065–1072, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
13. M. L. Puterman. *Markov Decision Processes*. Wiley, NY, 1994.
14. G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994.
15. Peter Stone and Richard S. Sutton. Keepaway soccer: a machine learning testbed. In Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*, pages 214–223. Springer Verlag, Berlin, 2002.
16. Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 2005. To appear.
17. Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
18. Matthew E. Taylor and Peter Stone. Behavior transfer for value-function-based reinforcement learning. In *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, July 2005. To appear.
19. Gerald Tesauro. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.
20. T. Walker, J. Shavlik, and R. Maclin. Relational reinforcement learning via sampling the space of first-order conjunctive features. In *Proceedings of the ICML Workshop on Relational Reinforcement Learning*, Banff, Canada, July 2004.
21. Shimon Whiteson, Nate Kohl, Risto Miikkulainen, and Peter Stone. Evolving keepaway soccer players through task decomposition. *Machine Learning*, 59(1):5–30, May 2005.

Learning to Approach a Moving Ball with a Simulated Two-Wheeled Robot

Felix Flentge

Department of Computer Science
Johannes Gutenberg-University Mainz
D-55099 Mainz, Germany

Abstract. We show how a two-wheeled robot can learn to approach a moving ball using *Reinforcement Learning*. The robot is controlled by setting the velocities of its two wheels. It has to reach the ball under certain conditions to be able to kick it towards a given target. In order to kick, the ball has to be in front of the robot. The robot also has to reach the ball at a certain angle in relation to the target, because the ball is always kicked in the direction from the center of the robot to the ball. The robot learns which velocity differences should be applied to the wheels: one of the wheels is set to the maximum velocity, the other one according to this difference. We apply a *REINFORCE* algorithm [1] in combination with some kind of extended *Growing Neural Gas* (GNG) [2] to learn these continuous actions. The resulting algorithm, called *ReinforceGNG*, is tested in a simulated environment with and without noise.

1 Introduction

In this paper we propose using a reinforcement-based learning method to learn the actions of a simulated two-wheeled robot to approach a moving ball. The ball has to be reached under certain conditions in order to kick it towards a given target. First, the ball has to be in front of the robot. Second, since the ball is always kicked in the direction from the center of the robot to the ball, the angle between the vector from the robot to the ball and the vector from the robot to the target has to be sufficiently small (as sketched in Fig. 4).

While there are algorithms to find optimal trajectories for two-wheeled robots to reach a given static target [3], the problem gets significantly harder if the target is moving and has to be reached under certain conditions. Although the basic principles of these algorithms of either turning around or moving forward may be applicable in this case, they are far from optimal if there is noise in perception or in the execution of actions. Under these circumstances there have to be small corrections every step. There is no straightforward way to design a good control strategy. We would have to simulate the ball's movement and possible movements of the robot to determine the point where we could meet the ball under the right conditions.

As far as we know our work is the first attempt to learn continuous actions for a two-wheeled robot in a problem as complex as sketched above. There have been some limited attempts with learning algorithms on similar problems. For example

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 106–117, 2006.

in [4] a robot arm learns to grasp a rolling ball, but only with a fixed initial situation and discrete actions. In [5] a simulated autonomous underwater vehicle with controls very similar to a two-wheeled robot is used in a two-dimensional environment. While the vehicle learns the continuous actions to reach the given static targets, the resulting trajectories do not look particulary time-optimal.

For learning we use a *REINFORCE* algorithm [1] together with an extended *Growing Neural Gas* (GNG) [2] in an actor-critic architecture [6]. The GNG is a self-organizing map which is able to adapt to the local dimension and the local density of an unknown possibly high-dimensional input distribution. This is particulary useful in the context of action learning because the input distribution is generally not known in advance and changes during training. Also, the data often stems from some lower-dimensional structures in the input space due to implicit dependencies between the data. The GNG is able to detect and to adapt to these structures. Using a REINFORCE algorithm to learn continuous actions should lead to smooth trajectories and should faciliate learning because similar situations should require similar actions.

In the next section we give a short description of the GNG and of the necessary extensions for function approximation. Since the data in our experiments have a specific order, i.e. they stem from trajectories, there have to be some further modifications to the original GNG. After describing these, we give a short introduction to REINFORCE algorithms and show how to use the extended GNG as a function approximator for these kind of algorithms. In section 4 we explain the simulation environment and show some results.

2 Growing Neural Gas for Function Approximation

2.1 Growing Neural Gas

The *Growing Neural Gas* (GNG) [2] is a self-organizing map without a fixed global network dimensionality, i.e. its topological structure adapts to the local dimensionality of the input data. It consists of a set of neurons $\{c_1, c_2, \ldots, c_N\}$ where each neuron c_i is associated with a position $w_i \in \mathbb{R}^d$ in input space. Neurons may be connected by undirected edges. If there is an edge between the neurons c_i and c_j, we denote it by $e_{i,j} = e_{j,i}$. The neighbors of a neuron c_i (i.e. the neurons having a common edge with c_i) will also be indexed $c_{i,j}, j = 0, \ldots, N_i$, where N_i is the number of neighbors of c_i and $c_{i,0} = c_i$ by convention. We use this notation for other variables as well, e.g. $w_{i,j}$ is the position of the j-th neighbor of c_i.

Whenever some input data point $x \in \mathbb{R}^d$ is presented to the network during training, the idea is to move the best matching (i.e. the closest) neuron c_b by some fraction α_b and its neighbors $c_{b,i}$ by some smaller fraction α_n towards x. If we have different x we will also write $c_{b(x)}$ to make clear which x is meant. Edges are always created between the closest neuron and the second-closest one, which leads to the induced Delaunay triangulation [7]. As the neurons move around, some aging scheme has to be used to delete edges which are no longer part of the induced Delaunay triangulation. Therefore, each edge $e_{i,j}$ has a counter named $age_{i,j}$, and edges with an age $age_{i,j}$ exceeding a certain age_{max} are deleted. New neurons are inserted from time to time in regions with a high quantization error until a maximum number N_{max} of neurons is reached. To estimate the local quantization error, every neuron is equipped with an error variable err_i which is updated with the current quantization error every time the respective neuron is the best matching neuron. Figure 1 shows the GNG algorithm.

```
Repeat (until some stopping criterion fulfilled):

 1. Observe $x \in \mathbb{R}^d$
 2. Find neurons $c_b$ and $c_{b'}$ with smallest euclidean distances to $x$:
        $\|x - w_b\| \leq \|x - w_{b'}\| \leq \|x - w_i\| \, \forall i \neq b, b', \; b \neq b'$
 3. Increase the age $age_{b,i}$ of all edges $e_{b,i}$ emanating from $c_b$ by 1
 4. If edge $e_{b,b'}$ does not exist, create it
 5. Reset $age_{b,b'}$ to 0
 6. Update error variables:
        $\Delta err_b = \|w_b - x\|^2$
        $\Delta err_i = -\delta_{err} \cdot err_i, i = 1, \ldots, N$ with constant $0 < \delta_{err} < 1$
 7. Update position vectors:
        $\Delta w_b = \alpha_b \, (x - w_b)$
        $\Delta w_{b,i} = \alpha_n \, (x - w_{b,i})$ for all neighbors $c_{b,i}, i = 1, \ldots, N_b$ of $c_b$
 8. Remove edges with $age_{i,j} > age_{max}$ and neurons without neighbors
 9. If $N \leq N_{max}$, add a new neuron $c_s$ every $N_{steps}$ steps as follows:
      – Find the neuron $c_q$ with the largest error $err_q$
      – Choose its neighbor $c_{q,r}$ with the largest error $err_{q,r}$, $r = 1, \ldots, N_q$
      – Insert a new neuron $c_s$ between $c_q$ and $c_{q,r}$
      – Reduce the errors $err_q$ and $err_{q,r}$ and set the error $err_s$ of the new neuron:
            $\Delta err_q = -\delta_{new} \cdot err_q$
            $\Delta err_{q,r} = -\delta_{new} \cdot err_{q,r}$
            $err_s = 0.5 \cdot (err_q + err_{q,r})$
```

Fig. 1. The GNG algorithm

2.2 Growing Neural Gas for Function Approximation

The easiest way to approximate a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ from training examples (x, y) with a self-organizing map is to associate each neuron c_i with a value $v_i \in \mathbb{R}$ and to have a local constant approximation by assigning the same value v_b to all input vectors x with $\|x - w_b\| < \|x - w_i\|$ for all $i \neq b$.

In order to improve the approximation quality of such a network we do not only use the best matching neuron's value v_b but also the values $v_{b,i}$ of its neighbors $c_{b,i}$. We then calculate normalized weights $m_{b,i}(x)$ for these values using Gaussian functions as it is done in *Radial Basis Function Networks* (RBF). The radii of the Gaussians used for calculating these weights are set to the average lengths of all edges emanating from $c_{b,i}$ following the suggestion in [8].

$$m_{b,i}(x) = \frac{\exp\left(-\frac{\|x-w_{b,i}\|^2}{l_{b,i}^2}\right)}{\sum_{j=0}^{N_b}\exp\left(-\frac{\|x-w_{b,j}\|^2}{l_{b,j}^2}\right)} \quad i = 0, \ldots, N_b \tag{1}$$

$$l_{b,j} = \text{average length of all edges emanating from neuron } c_{b,j}$$

The final approximation $\widetilde{F}(x)$ is then calculated as a weighted sum:

$$\widetilde{F}(x) = \sum_{i=0}^{N_b} m_{b,i}(x) v_{b,i} \tag{2}$$

Please note that we only use the values of the best matching neuron and its neighbors for calculating the final approximation $\widetilde{F}(x)$ instead of using all values as it is done in the common RBF approach or in [8]. There are two reasons for this: first, it is faster since we do not have to calculate weights for all neurons, and second, using all values did not work well in the learning experiments described below. We think that is because the concept of "distance" is quite problematic if different input dimensions have different meanings (e.g. distances, angles and angular velocities). This problem is eased using (2) because neurons which are close to each other but are not direct neighbors have no influence on the approximation[1].

Whenever we observe some training data (x, y), the values of the best matching neuron and its neighbors $v_{b,i}$ are trained by gradient descent with learning rate α_v:

$$\Delta v_{b,i} = \alpha_v m_{b,i}(x)\left(y - \widetilde{F}(x)\right) \quad i = 0, \ldots, N_b. \tag{3}$$

2.3 Learning from Trajectories

An important difference to pure function approximation tasks is that in action learning the data stem from trajectories through the state space running from an initial state x_0 to some terminal state x_T. If we use the standard GNG algorithm, these data would lead to strange results because the neurons would concentrate at the ends of the trajectories. Therefore, we keep the neurons' positions fixed during an episode (i.e. the time span between $t = 0$ and $t = T$) and accumulate the changes in auxiliary variables $\widehat{w}_i$ attached to each neuron. We keep track of how often we changed these auxiliary variables by a counter z_i for each neuron. At the end of an episode we use these variables to change the positions of the neurons according to the averages of all changes:

$$\Delta w_i = \frac{\widehat{w}_i}{z_i} \text{ for all neurons } i = 1, \ldots, N \text{ with } z_i > 0 \tag{4}$$

Furthermore, we postpone the insertion of new neurons and edges as well as the deletion of edges until the end of an episode (steps 4, 5, 8, and 9 of the

[1] Of course, in contrast to RBF networks the resulting approximation is not continuous anymore, but has some points of discontinuity where the best matching neuron changes.

algorithm in Fig. 1). Therefore, we keep track of all pairs of neurons closest and second-closest to the input vectors x_t occurring during an epsiode in a list P. Figure 2 provides an overview of the complete algorithm for training. For look-up one has to perform only steps 1 and 2 as well as the steps 9 and 10 of the first part.

```
During an episode:

 1. Get input pair $(x, y)$
 2. Find the two nearest neurons $c_b$ and $c_{b'}$ (Fig. 1 step 2)
 3. Increase the age $age_{b,i}$ of all edges emanating from $c_b$ by 1
 4. If the pair $(c_b, c_{b'})$ is not in list $P$, add it
 5. Update quantization error variables (Fig. 1 step 6)
 6. Update auxiliary position variables:
        $\Delta\widehat{w}_b = \alpha_b\,(x - w_b)$
        $\Delta\widehat{w}_{b,i} = \alpha_n\,(x - w_{b,i})$ for all neighbors $c_{b,i}, i = 1, \ldots, N_b$ of $c_b$
 7. Increase counter variables $z_b$ and $z_{b,i}$ of $c_b$ and of all of its neighbors $c_{b,i}$ by 1
 8. If $N + N_{insert} < N_{max}$, increase insert counter $N_{insert}$ every $N_{steps}$ steps by 1
 9. Calculate the weights $m_{b,i}(x)$ according to (1)
10. Calculate the final approximation $\widetilde{F}(x)$ according to (2)
11. Train the values $v_{b,i}$ according to (3)

At the end of an episode:

 1. Update the weights $w_i$ according to (4)
 2. If there are any pairs $(c_i, c_j)$ in $P$ without edges $e_{i,j}$, create them
 3. Set the ages $a_{i,j}$ of edges $e_{i,j}$ with a pair $(c_i, c_j)$ in $P$ to 0 and delete $P$
 4. Remove edges with $age_{i,j} > age_{max}$ and neurons without neighbors
 5. Insert $N_{insert}$ new neurons (analogous to Fig. 1 step 9) and set the values of
    the new neurons according to the current approximations at their positions:
        $v_s = \widetilde{F}(w_s)$
 6. Set all $\widehat{w}_i$, counters $z_i$ and $N_{insert}$ to 0
```

Fig. 2. The extended GNG algorithm for data from trajectories

3 ReinforceGNG

In reinforcement learning problems there is an agent which perceives the states x_t of an environment and chooses some action a_t at each time step t. The agent also receives some reward $r_t \in \mathbb{R}$ which rates the situation x_t it is in and the action a_{t-1} it has chosen the last time step. The goal of the agent is to maximize the sum of rewards it receives over time. Transitions between states and rewards may be stochastic and the rewards may also be delayed.

3.1 REINFORCE Algorithms

The reinforcement learning component of ReinforceGNG is based on the class of REINFORCE algorithms introduced by Williams in 1992 [1]. The general setting for REINFORCE algorithms is a network of (stochastic) units u_i which propagate an input x through the net to produce an output a. This output leads to a direct scalar reinforcement signal R that is used by the units in the net to adjust their weights w_i. A stochastic unit u_i draws its output a_i from a probability distribution depending on its input x_i and its weight vector w_i. In the continuous case the probability distribution is given by a density function $g_i(\xi, x_i, w_i)$.

REINFORCE algorithms try to maximize the expectation value of the immediate reinforcement values R over time. Under certain stationary and independence conditions on the environment's choice of inputs and reinforcement signals, this expectation value only depends on the units' weight vectors w_i. Therefore, we will assume a weight matrix W consisting of all weight vectors w_i and will write $E\left\{R|W\right\}$ for this expectation value. Williams shows that with an update of the form

$$\Delta w_i = \alpha_i (R - b_i) e_i \tag{5}$$

the average update vector lies in a direction where this performance measure $E\left\{R|W\right\}$ is increasing. In this update equation $\alpha_i > 0$ is some learning rate (depending at most on the time t and w_i), b_i is the reinforcement baseline (just some value conditionally independent of the output a_i given W and x_i) and

$$e_i = \frac{\partial \ln g_i(a_i, x_i, w_i)}{\partial w_i} \tag{6}$$

is the characteristic eligibility of w_i.

In the following we assume that we have only one unit u and omit all the indices i. An interesting possibility is to draw the output a from a normal distribution $N(\mu(x), \sigma(x))$ with input dependent means $\mu(x)$ and standard deviations $\sigma(x)$ as weights. The standard deviations $\sigma(x)$ can be used to control the behavior of the algorithm: large $\sigma(x)$ lead to an explorative behavior while small $\sigma(x)$ lead to the exploition of the behavior learned so far. Using the density of the normal distribution we get the following characteristic eligibilities:

$$\frac{\partial \ln g\left(a, \mu(x), \sigma(x)\right)}{\partial \mu(x)} = \frac{a - \mu(x)}{\sigma(x)^2} \tag{7}$$

$$\frac{\partial \ln g\left(a, \mu(x), \sigma(x)\right)}{\partial \sigma(x)} = \frac{\left(a - \mu(x)\right)^2 - \sigma(x)^2}{\sigma(x)^3} \tag{8}$$

Williams suggests setting the learning rates to $\alpha\sigma(x)^2$ and to choose the reinforcement baseline $b(x)$ according to a reinforcement comparison scheme, i.e. to an estimate of the upcoming reinforcement based on the experience so far. This leads to the update equations:

$$\Delta\mu(x) = \alpha(R - b(x))(a - \mu(x)) \tag{9}$$

$$\Delta\sigma(x) = \alpha(R - b(x))\frac{\left(a - \mu(x)\right)^2 - \sigma(x)^2}{\sigma(x)} \tag{10}$$

These equations give some insight in how the algorithm works: if the agent gets an reinforcement signal R better than the reinforcement baseline $b(x)$, $\mu(x)$ is moved in the direction of the output a, if R is worse than $b(x)$, $\mu(x)$ is moved in the opposite direction. In (10) the standard deviation $\sigma(x)$ is changed to make the occurance of a more likely if R is better than the reinforcement baseline $b(x)$, and vice versa. E.g. if $R > b(x)$ and the distance $|a - \mu(x)|$ is greater than the standard deviation $\sigma(x)$, $\sigma(x)$ is increased.

3.2 ReinforceGNG

Since the task we want to learn involves delayed rewards, but REINFORCE algorithms learn only from direct reinforcement signals[2], we first have to think about how we can get a direct reinforcement value from delayed rewards. The standard way to do this is to use some kind of actor-critic architecture [6]. The actor chooses an action a_t for a state x_t and receives a direct reinforcement value R_{t+1} from the critic one time step later. This reinforcement value is used to improve the policy (i.e. the mapping from situations to actions) π used by the actor. The critic learns an approximation $\widetilde{V}(x)$ of the value function $V^\pi(x)$ for the current policy π of the actor. The value $V^\pi(x_t)$ of a state x_t for a certain policy π is given by the expectation value of the discounted sum of rewards gained when starting in this state and following the policy π until the last time step T.

$$V^\pi(x) = E_\pi \left\{ \sum_{k=0}^{T} \gamma^k r_{t+k+1} \;\middle|\; x_t = x \right\} \tag{11}$$

The future rewards r_{t+k+1} are discounted by γ to decrease the influence of later rewards in favor of more immediate ones. Usually, the value function cannot be calculated exactly, because the transition probabilities between states and the expectation values of the rewards are not known. Reinforcement learning methods often try to approximate this function. One way to learn an approximate value function, known as *Temporal Difference* (TD) learning, is to base the estimation of the value of a state x_t on the sum of the immediate rewards and the discounted approximated value of the next state $r_{t+1} + \gamma\widetilde{V}(x_{t+1})$. This sum can also be used as a direct reinforcement signal $R_{t+1}$for the actor rating the last action a_t. Since r_{t+1} and $\widetilde{V}(x_{t+1})$ are only known in the following timestep, the updates for time t are performed at time $t+1$.

The extended GNG is used to learn the approximate value function $\widetilde{V}(x)$, the means $\widetilde{\mu}(x)$, and the variances $\widetilde{\sigma}(x)$ used for determing the actions. The neurons' values v_i are used for approximating the value function $\widetilde{V}(x)$. They are trained as in (3) with the data pairs $\left(x_t, r_{t+1} + \gamma\widetilde{V}(x_{t+1})\right)$. Two additional variables are added to each neuron: μ_i and σ_i. These are used to calculate the approximations $\widetilde{\mu}(x)$ and $\widetilde{\sigma}(x)$ as weighted sums analogous to (2). They are trained according

[2] There is one exception for episodic tasks with only a single reward that is delivered at the end of an episode.

to (9) and (10) taking the weights $m_{b(x_t),i}(x)$ into account. With $\widetilde{V}(x_t)$ as the reinforcement baseline we get[3]:

$$\Delta\mu_{b(x_t),i} = \alpha_\mu m_{b(x_t),i}(x_t)\left(r_{t+1} + \gamma\widetilde{V}(x_{t+1}) - \widetilde{V}(x_t)\right)(a_t - \widetilde{\mu}(x_t)) \tag{12}$$

$$\Delta\sigma_{b(x_t),i} = \alpha_\sigma m_{b(x_t),i}(x_t)\left(r_{t+1} + \gamma\widetilde{V}(x_{t+1}) - \widetilde{V}(x_t)\right)\frac{(a_t - \widetilde{\mu}(x_t))^2 - \widetilde{\sigma}(x_t)^2}{\widetilde{\sigma}(x_t)} \tag{13}$$

If there are minimum and maximum values for actions a_{min} and a_{max}, it has to be assured that the resulting μ_i are within these bounds by setting them to their respective limits. The same holds for the σ_i with minimum σ_{min} and maximum σ_{max} to ensure proper exploration behavior. Figure 3 gives an overview of the complete ReinforceGNG algorithm. For look-up only the approximate mean $\mu(x_{t+1})$ has to be calculated and is then used as action a_{t+1}.

```
1. Receive situation $x_{t+1}$ and reward $r_{t+1}$
2. Calculate $\widetilde{V}(x_{t+1})$, $\widetilde{\mu}(x_{t+1})$ and $\widetilde{\sigma}(x_{t+1})$ as weighted sums according to (2)
3. Train the extended GNG with $\left(x_t, r_{t+1} + \gamma\widetilde{V}(x_{t+1})\right)$ according to the extended
   GNG algorithm as described in Fig. 2
4. Train the $\mu_{b(x_t),i}$ of last time step's best matching neuron $c_{b(x_t)}$ and its neigh-
   bors $c_{b(x_t),i}$ according to (12)
5. Train the $\sigma_{b(x_t),i}$ of last time step's best matching neuron $c_{b(x_t)}$ and its neigh-
   bors $c_{b(x_t),i}$ according to (13)
6. Determine action $a_{t+1} \sim N(\widetilde{\mu}(x_{t+1}), \widetilde{\sigma}(x_{t+1}))$
   if $a_{t+1} < a_{min} \Rightarrow a_{t+1} = a_{min}$; if $a_{t+1} > a_{max} \Rightarrow a_{t+1} = a_{max}$
```

Fig. 3. The complete ReinforceGNG algorithm

4 Experiments

It was intended to use ReinforceGNG to learn an "intelligent" approach ball behavior for the 3D Simulation League. Unfortunately, simulated two-wheeled robots with an explicit kick-effector were not yet available at the time of our experiments. So we had to create our own simulation environment, but we tried to build the simulation meaningful with respect to the Simulation League.

4.1 Simulation Setup

We decided two use circular robots with a diameter of 0.44 m, which was the size of the robots in the 3D Simulation League in RoboCup 2004 and is also

[3] This can be regarded as using a reinforcement comparison scheme since the value of a situation is an estimate of the upcoming rewards based on the experience so far. The resulting expression $r_{t+1} + \gamma\widetilde{V}(x_{t+1}) - \widetilde{V}(x_t)$ is known as TD error.

about the size of a midsize robot. The maximum velocity of the robot is 2 $\frac{\mathrm{m}}{\mathrm{s}}$. We do not consider acceleration, braking or friction for the robot's movements and do not simulate collisions. The robot is able to change the velocities of its wheels every 100 milliseconds. The diameter of the ball is 0.22 m. The velocity of the ball decreases exponentially with $v_{t+1}^B = 0.995 v_t^B$ every simulation step of 10 milliseconds, i.e. every second the ball loses about 40 percent of its velocity.

The goal is to position the robot relative to the ball so it is able to kick the ball towards a given target. Figure 4 shows the situation. To kick the ball, the distance between the robot and the ball d_t^B has to be 0.07 m or smaller and the angle between the orientation of the robot and the ball α_t^B has to be no bigger than 22.5° in either direction. The ball is kicked in the direction of the vector from the center of the robot to the center of the ball. Therefore, to reach the target the angle α_t^T between this vector and the vector to the target should not be bigger than 12.25°. If these conditions are met, the trial is considered successful. A trial is canceled after 25 seconds of simulation time if the robot does not reach the ball under the right conditions.

We concentrate on the situation where the ball is on the ground and not too fast. We think that otherwise it would be almost impossible to approach the ball at the required angle due to noise in the perception of the ball and in the execution of the actions. Also, we assume that the ball is somewhere near the robot, because this is very easy to achieve with some hand-coded behavior. Therefore, at the beginning of a trial, the ball is set to an arbitrary point up to 5 m away from the robot with a velocity of up to 4 $\frac{\mathrm{m}}{\mathrm{s}}$. The target is set to an arbitrary point 5–15 m away from the ball.

The action we learn is a continuous value a between -1 and 1. It encodes the difference in velocity which should be applied to the wheels, i.e. if the action is

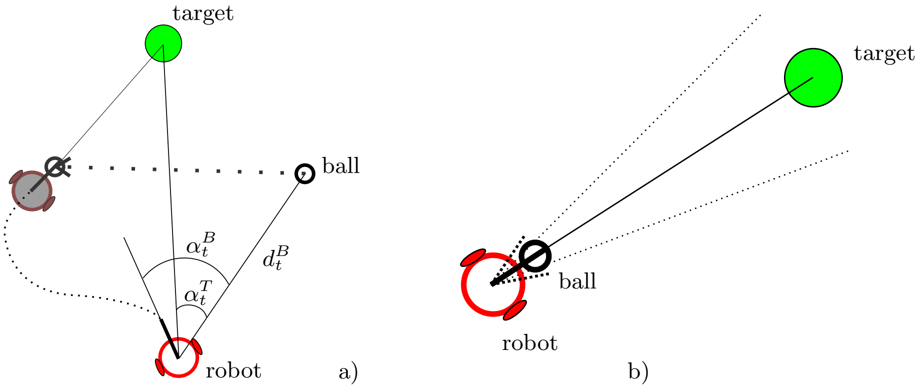


Fig. 4. Sketch of different situations with the relevant distance and angles. a) Some random initial situation with a possible solution trajectory. b) The terminal situation to reach in order to kick the ball towards the target. The dashed lines show the limits for the angle between the robot's orientation and the ball and for the angle between the kick direction and the target direction.

greater then zero, we apply a velocity of 1 to the left wheel and $1 - 2|a|$ to the right wheel and vice versa. So, the robot always tries to move as fast as possible.

The situation is encoded in a five dimensional input vector with the following dimensions (see Fig. 4):

- the distance d_t^B from the robot to the ball
- the angle α_t^B between the current orientation of the robot and the direction towards the ball
- the angle α_t^Tbetween the direction from the robot towards the target and the direction from the robot towards the ball
- the difference $\Delta\alpha_t^B = \alpha_t^B - \alpha_{t-1}^B$ in α_t^B between the last and the current time step
- the difference $\Delta\alpha_t^T = \alpha_t^T - \alpha_{t-1}^T$ in α_t^T between the last and the current time step

These dimension are normalized to lie roughly between 0 and 1. Every time step the robot does not reach the required conditions, it gets a penalty of $-0.01 d_t^B$. If the trial is successful, the robot gets a reward of 100. The small penalty on the distance makes successful trials more likely in the beginning because the robot tries to minimize the distance to the ball and will eventually meet the ball under the right conditions.

To choose the learning parameters we did some preliminary tests with different settings but they were not really optimized systematically. The maximum number of neurons is set to 1000, we started with $3^5 = 243$ neurons evenly distributed in the hypercube $[0,1]^5$. New neurons are inserted every 100 steps. The learning rates for the movement of the neurons α_n and α_b are set to 0.01 and to 0.025. The maximum age of edges is 100. The neurons' errors are reduced with rate $\delta_{err} = 0.001$ every time step and with $\delta_{new} = 0.01$ if a new neuron is inserted. In the beginning all μ_i and v_i are initialized to 0, σ_i to 0.25. The minimum and maximum values of σ_i are set to $\sigma_{min} = 0.1$ and $\sigma_{max} = 0.25$. The learning rates α_μ and α_σ are set to 0.1 and the discount factor γ to 0.9.

4.2 Simulation Results

To assess how well the simulated robot has learned the task we perform a fixed set of predefined tests. In each test the ball is set 2 m away in front of the robot. We put targets at 10 m distance from the ball and at 8 different angles relative to the robot's orientation $(-180°, -135°, -90°, \ldots, 90°, 135°)$. The ball either moves at 4 $\frac{\text{m}}{\text{s}}$ in one of the four directions $-180°, -90°, 0°, 90°$, or does not move at all. So all in all we have 40 different tests.

We performed several experiments and the robot always learned to approach the rolling ball within the required angles for all test situations. In Fig. 5 one can see some of the learned trajectories, which look quite smooth. The targets are almost always hit with high precision.

We also did experiments with noise in perception and in the execution of actions. For perceptional noise, we added normal distributed noise of $N(0, 0.0965)$ to the distance information and normal distributed noise of $N(0, 0.1225)$ to angular information as it is done in the 3D Simulation League. For noise in the

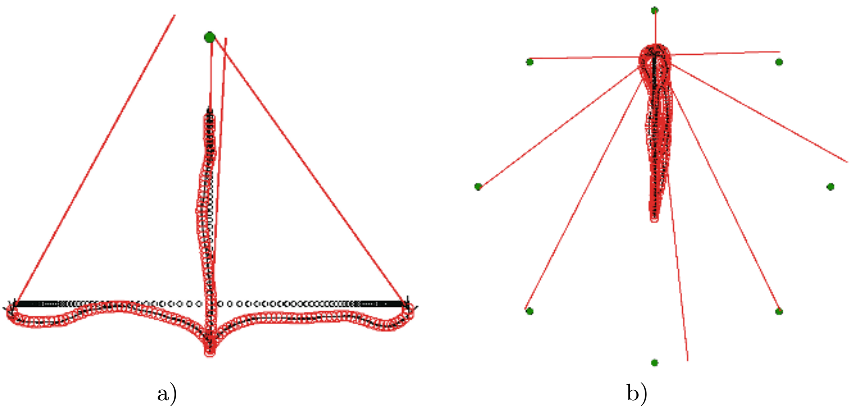


Fig. 5. Trajectories of the learned approach ball behavior. The robot is the bigger empty circle, the ball is the smaller empty circle and the target is the filled circle. The objects were drawn every 100 milliseconds. The straight lines show the direction the ball is kicked in at the end of each episode. a) The four different directions for the moving ball with one target. The ball moving towards the agent can hardly be seen because it is reached significantly faster than all other balls. b) Trajectories of the robot following an upward moving ball with shots to all eight different targets.

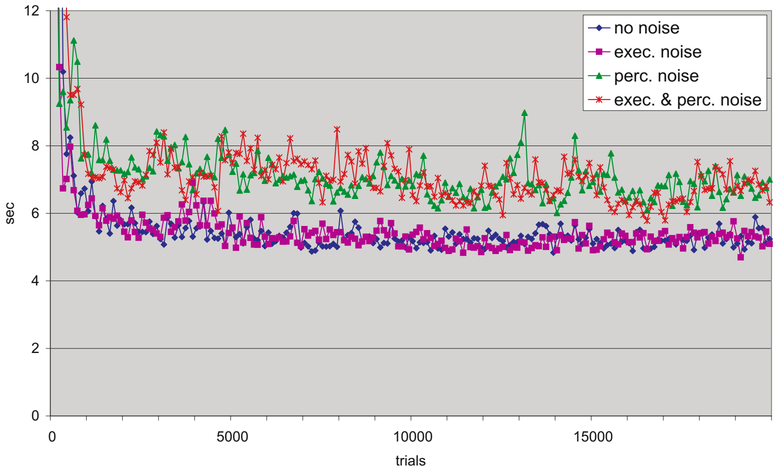


Fig. 6. Average simulation time needed per test from single runs without noise, with noise in the execution of the action, with noise in the perception, and both

execution of actions, we added normal distributed noise of $N(0, 0.05)$ to the velocities of the wheels every simulation step. ReinforceGNG was again able to learn to approach the rolling ball in all test situations. Noise in the execution of actions had practically no influence on the robot's performance. Figure 6 shows the development of the average simulation time needed per test. All 40 tests were performed every 100 trials. The values shown are from single runs. All in all we did 10 individual runs for all kinds of noise and the results always look very similar to those in the figure. We see that the method learns relatively quickly but sometimes the performance gets worse during learning. Therefore, we keep track of the best average time so far and if we get a better value, we save the current network for later use.

5 Conclusion

We showed how a REINFORCE algorithm in combination with an extended GNG with some modifications regarding the learning from trajectories can successfully learn to approach a moving ball under certain conditions with respect to a given target. For the future we plan to use this method in our agent for the 3D Simulation League. We also plan to do experiments with moving targets and train networks for different requirements on the precision of the kick. Unprecise but fast kicks should be used in situations were opponents are near and precise kicks can be used if the agent has enough time. We also plan to incorporate the proposed learning method into a midsize robot.

References

1. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8** (1992) 229–256
2. Fritzke, B.: A growing neural gas network learns topologies. In Tesauro, G., Touretzky, D., Leen, T., eds.: Advances in Neural Information Processing Systems 7, Cambridge MA (1995)
3. Balkcom, D., Mason, M.: Time optimal trajectories for bounded velocity differential drive vehicles. International Journal of Robotics Research **21** (2002) 199–217
4. Riedmiller, M., Janusz, B.: Using neural reinforcement controllers in robotics. In: Proc. 8th Australian Conference on Artificial Intelligence, Canberra (1995)
5. Gaskett, C., Wettergreen, D., Zelinsky, A.: Q-learning in continuous state and action spaces. In: Australian Joint Conference on Artificial Intelligence. (1999) 417–428
6. Sutton, R., Barto, A.: Reinforcement Learning. An Introduction. MIT Press, Cambridge, Massachusetts (2000)
7. Martinetz, T.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: Proc. of ICANN'93, Springer (1993) 427–434
8. Fritzke, B.: Fast learning with incremental RBF networks. Neural Processing Letters **1** (1994) 2–5

Sequential Pattern Mining for Situation and Behavior Prediction in Simulated Robotic Soccer

Andreas D. Lattner, Andrea Miene, Ubbo Visser, and Otthein Herzog

Center for Computing Technologies – TZI, Universitaet Bremen,
PO Box 330 440, D-28334 Bremen, Germany
{adl, andrea, visser, herzog}@tzi.de

Abstract. Agents in dynamic environments have to deal with world representations that change over time. In order to allow agents to act autonomously and to make their decisions on a solid basis an interpretation of the current scene is necessary. If intentions of other agents or events that are likely to happen in the future can be recognized the agent's performance can be improved as it can adapt the behavior to the situation. In this work we present an approach which applies unsupervised symbolic learning off-line to a qualitative abstraction in order to create frequent patterns in dynamic scenes. These patterns can be later applied during runtime in order to predict future situations and behaviors. The pattern mining approach was applied to two games of the 2D RoboCup simulation league.

1 Introduction

Agents in dynamic environments have to deal with world representations that change over time. In order to allow agents to act autonomously and to make their decisions on a solid basis an interpretation of the current scene is necessary. Scene interpretation can be done by checking if certain patterns match the current belief of the world. If intentions of other agents or events that are likely to happen in the future can be recognized the agent's performance can be improved as it can adapt the behavior to the situation. One step into this direction is the recognition of formations [24].

We focus on qualitative representations as they allow a concise representation of the relevant information. Such a representation provides means to use background knowledge, to plan future actions, to recognize plans of other agents, and is comprehensible for humans the same time. In our approach we map quantitative data to qualitative representations. We use time series which are divided into different segments satisfying certain monotonicity or threshold conditions [18, 19]. One example is that if the distance between two objects is observed it can be divided into increasing and decreasing distance representing approaching and departing relations (cf. [19]).

Additionally to the requirement to handle situations which change over time, relations between arbitrary objects can exist in their belief of the world. In this work we present an approach which applies unsupervised symbolic learning to a

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 118–129, 2006.

qualitative abstraction in order to create frequent patterns in dynamic scenes. Here, we apply an extended version of the sequential pattern mining algorithm presented in [13] to data from simulated robotic soccer games of the 2D RoboCup simulation league. We propose the application of the learned rules to predict future situations and behavior in order to support behavior decision.

2 Related Work

Association rule mining addresses the problem of discovering association rules in data. One famous example is the mining of rules in basket data [1]. Different algorithms have been developed for the mining of association rules in item sets (e.g., [2]). Mannila et al. extended association rule mining by taking event sequences into account [15]. They describe algorithms which find all relevant episodes which occur frequently in the event sequence. H"oppner presents an approach for learning rules about temporal relationships between labeled time intervals [8]. The labeled time intervals consist of propositions. Relationships are described by Allen's interval logic [3]. Other researchers in the area of spatial association rule mining allow for more complex representations with variables but do not take temporal interval relations into account (e.g., [11, 14, 16]).

The learning approach presented here combines ideas from different directions. Similar to H"oppner's work [8] the learned patterns describe temporal interrelationships with interval logic. Contrary to H"oppner's approach our representation allows for describing predicates between different objects similar to approaches like [14]. The generation of frequent patterns comprises a top-down approach starting from the most general pattern and specializing it. At each level of the pattern mining just the frequent patterns of the previous step are taken into account knowing that only combinations of frequent patterns can result in frequent patterns again which is a typical approach in association rule mining (e.g., [15]).

RoboCup is used as an application domain for learning approaches in different papers (e.g., [10, 12, 17, 21, 22, 23, 25]). Many of the recent approaches apply reinforcement learning to robotic soccer.

Riley and Veloso [20] use a set of pre-defined movement models and compare these with the actual movement of the players in set play situation. In new set play situations the coach then uses the gathered information to predict the opponent agent's behavior and to generate a plan for his own players. The approach uses probabilistic models and can be used both off-line and on-line.

Frank and colleagues [5] presented a real-time approach which is based on statistical methods. The approach gathers information such as the percentage of ball-holding of a certain player or which player passes the ball to which team mate. The result is a thorough statistical analysis which can then be used to derive information about a game being played. This can help for new future developments of a team.

A hybrid approach to learn the coordinated sequential behavior of teams was presented by Kaminka and colleagues [10]. The idea is to take time-series of continuous multi-variate observations and then parse and transform them into a single-variable categorial time-series. The authors use a set of behavior recognizers that focus only on recognizing simple and basic behaviors or the agents (e.g., pass, dribble). The data are then represented in a trie (a tree-like data structure) to support two statistical methods: (a) frequency counting and (b) statistical dependency detection. Experiments showed that the latter method is more suitable to discover sequential behavior.

Huang and colleagues [9] recently published an approach for plan recognition and retrieval for multi-agent systems. The approach is based on observations of agents' coordinative behaviors. The basis are players' element behaviors sequences (e.g., pass, dribble, shoot) which are sorted in a temporal order. The field is decomposed into cells where each cell denotes one agent's behavior at a time slice. Interesting and frequent behavior sequences are considered as the team's plans on the assumption that the team's plan is embedded in those sequences. The discovery of significance of sequence patterns are based on statistical evidences. The promising results are plans based on observation.

Most similar to our work are the approaches of Kaminka et al. [10] and Huang et al. [9] as they also create a sequence of certain events or behaviors and search for frequent sequences. The main difference to our approach is the representational power of the learned patterns. Our representation allows for using variables in the learned rules and for identifiying arbitrary temporal relations between predicates (e.g., like Allen's interval logic).

3 Qualitative Motion Description

Our approach to qualitative motion description maps quantitative data to qualitative representations. Time series are divided into intervals which satisfy certain monotonicity or threshold conditions [19, 18]. Time series include absolute motion of single objects and relative motion of pairs of objects, which is described by changes in their pairwise spatial relations over time.

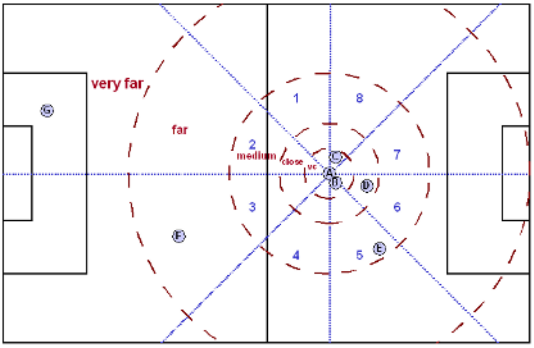


Fig. 1. Distance and direction classes

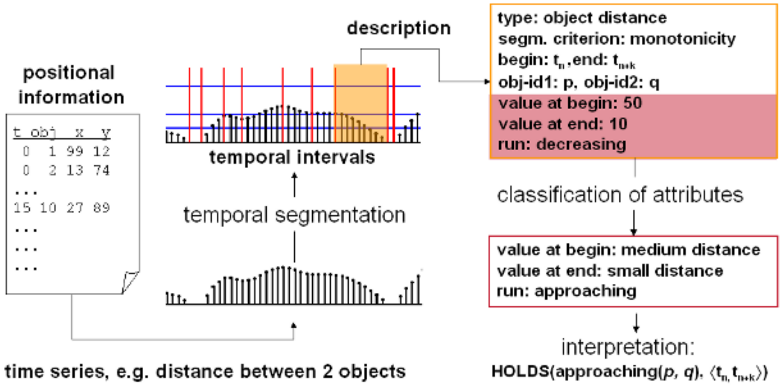


Fig. 2. Generation of motion description

On a quantitative level the objects' absolute and relative motion is described by four types of time series: the motion direction and speed of each object, and the spatial direction and distance for each pair of objects. In a first abstraction step each time series is segmented into time intervals of homogeneous motion values. We use two different segmentation methods: a threshold-based segmentation method, which represents the values of an interval by their average value and a monotonicity-based segmentation method, which groups together increasing, decreasing and constant values sequences. This preserves the dynamic of motion and allows for the description of dynamic aspects as, e.g., acceleration or approaching/departing. In a second step the attribute values describing the intervals are mapped onto qualitative classes for direction, speed or distance, respectively (see Fig. 1). We distinguish eight direction classes and five speed respective distance classes, which are organized in distance systems [7]. The radius of each distance class is double the size of the radius of the previous one. For the soccer domain we use the heading of each player as reference axis for the representation of the spatial relations to the surrounding objects which leads to an egocentric point of view.

Fig. 2 shows an example of the entire process of motion description for a time series of object distances, segmented by the monotonicity-based segmentation criterion.

The interval shown in the example in Fig. 2 is interpreted as an approaching of the objects p and q, which is expressed by the term HOLDS(approaching(p, q), $\langle t_n, t_{n+k}\rangle$). The predicate HOLDS expresses the coherence between a certain situation (here approaching) and the time interval $\langle t_n, t_{n+k}\rangle$ in which it is taking place or is valid [4].

As input for the learning algorithm described in the following section we use a subset of the motion description including intervals concerning the relations *meets, approaches* and *departs* between a player and the ball. For the relation *approaches* we restrict to intervals in which the two objects approach at least until they are at a medium distance. For the relation *departs* we restrict to intervals in which the two objects are in medium distance or closer when starting

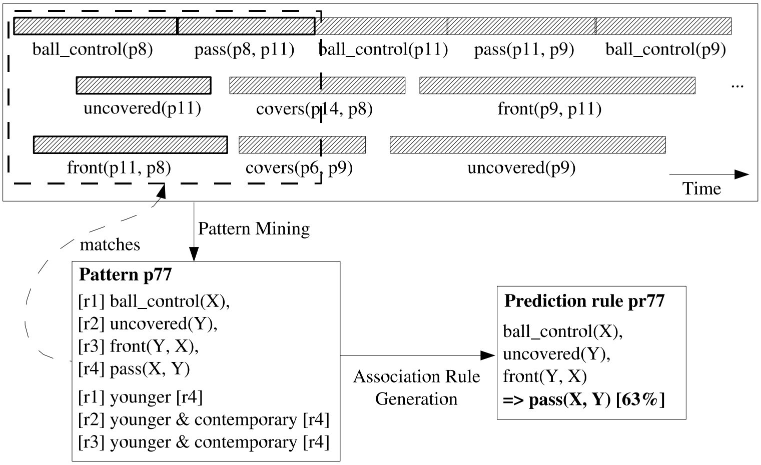


Fig. 3. Pattern and prediction rule generation

departing. Additionally further higher-level relations like *pass* and *ball_control* are extracted from the scenes.

4 Sequential Pattern Mining

Here, a dynamic scene is represented symbolically by a set of objects and predicates between these objects as created by the qualitative abstraction described in the previous section. The predicates are only valid for certain time intervals and the scene can thus be considered as a sequence of (spatial or conceptual) predicates. These predicates are in specific temporal relations regarding the time dimension. An example for such a sequence can be seen at the top of Fig. 3.

Each predicate r is an instance of a predicate definition rd. We use the letter r for predicates/relations; the letter p is used for patterns. $\mathcal{R}_{schema} = \{rd_1, rd_2, \ldots\}$ is the set of all predicate definitions $rd_i := \langle l_i, a_i \rangle$ with label l_i and arity a_i, i.e., each rd_i defines a predicate between a_i objects. Predicates can be hierarchically structured. If a predicate definition rd_1 specializes another predicate definition rd_2 all instances of rd_1 are also instances of the super predicate rd_2.

If we have to handle more than one dynamic scene, let $\mathcal{S} = \{s_1, s_2, \ldots\}$ be the set of the different sequences s_i. A single sequence s_i is defined as $s_i = (\mathcal{R}_i, \mathcal{TR}_i, \mathcal{C}_i)$ where $\mathcal{R}_i$ is the set of predicates, $\mathcal{TR}_i$ is the set of temporal relations and $\mathcal{C}_i$ is the set of constants representing different objects in the scene. Every constant is an instance of a class (default is the top concept "object") and classes form an inheritance hierarchy. Each predicate is defined as $r(c_1, \ldots, c_n)$ with r being an instance of $rd_i \in \mathcal{R}_{schema}$, having arity $n = a_i$, and $c_{i,1}, \ldots, c_{i,n} \in \mathcal{C}_i$ are representing the objects where the predicate holds. The set of temporal relations $\mathcal{TR}_i = \{tr_1, tr_2, \ldots\}$ defines relations between pairs of elements in $\mathcal{R}_i$. Each temporal relation is defined as

$tr_i(r_a, op, r_b)$ with $r_a, r_b \in \mathcal{R}_i$. op is the set of valid temporal relations. If Allen's temporal relations between intervals [3] are used, this set is defined as $op \in \{<, =, >, d, di, o, oi, m, mi, s, si, f, fi\}$. It is also possible to use other temporal relations, e.g., those defined by Freksa [6].

4.1 Pattern Representation and Pattern Matching

Patterns are abstract descriptions of sequence parts with specific properties. A pattern defines what predicates must occur and how their temporal interrelationship has to be. Let $\mathcal{P} = \{p_1, p_2, \ldots\}$ be the set of all patterns p_i. A pattern is (similar to sequences) defined as $p_i = (\mathcal{R}_i, \mathcal{TR}_i, \mathcal{V}_i)$.

$\mathcal{R}_i$ is the set of predicates $r_{ij}(v_{ij,1}, \ldots, v_{ij,n})$ with $v_{ij,1}, \ldots, v_{ij,n} \in \mathcal{V}_i$. $\mathcal{V}_i$ is the set of all variables used in the pattern. A class is assigned to each variable. $\mathcal{TR}_i$ defines the set of the temporal relations which have already been defined above.

A pattern p matches in a (part of a) sequence sp if there exists a mapping of a subset of the constants in sp to all variables in p such that all predicates defined in the pattern exist between the mapped objects and all time constraints of p are satisfied by the time intervals in the sequence without violating the class restrictions. In order to restrict the exploration region a window size can be defined. Only matches within a certain neighborhood (specified by the window size) are valid.

During the pattern matching algorithm a sliding window is used, and at each position of the window all matches for the different patterns are collected. A match consists of the position in the sequence and an assignment of objects to the variables of the pattern. Fig. 3 illustrates a sample pattern and one of the matches in the given sequence. In this example temporal relations as defined by Freksa [6] are used.

4.2 Pattern Generation

Different patterns can be put into generalization-specialization relations. A pattern p_1 subsumes another pattern p_2 if it covers all sequence parts which are covered by p_2: $p_1 \sqsubseteq p_2 := \forall sp, matches(p_2, sp) : matches(p_1, sp)$.

If p_1 additionally covers at least one sequence part which is not covered by p_2 it is more general: $p_1 \sqsubset p_2 := p_1 \sqsubseteq p_2 \wedge \exists sp_x : matches(p_1, sp_x), \neg matches(p2, sp_x)$. This is the case if $p_1 \sqsubseteq p_2 \wedge p_1 \not\sqsupseteq p_2$.

In order to specialize a pattern it is possible to add a new predicate r to $\mathcal{R}_i$, add a new temporal relation tr to $\mathcal{TR}_i$, specialize the class of a variable, unify two variables, or specialize a predicate, i.e., replacing it with another more special predicate. Accordingly it is possible to generalize a pattern by removing a predicate r from $\mathcal{R}_i$, removing a temporal relation tr from $\mathcal{TR}_i$, inserting a new variable, or generalizing a predicate r, i.e., replacing it with another more general predicate.

At the specialization of a pattern by adding a predicate a new instance of any of the predicate definitions can be added to the pattern with variables which are not used in the pattern so far. Specializing the class of a variable means that the

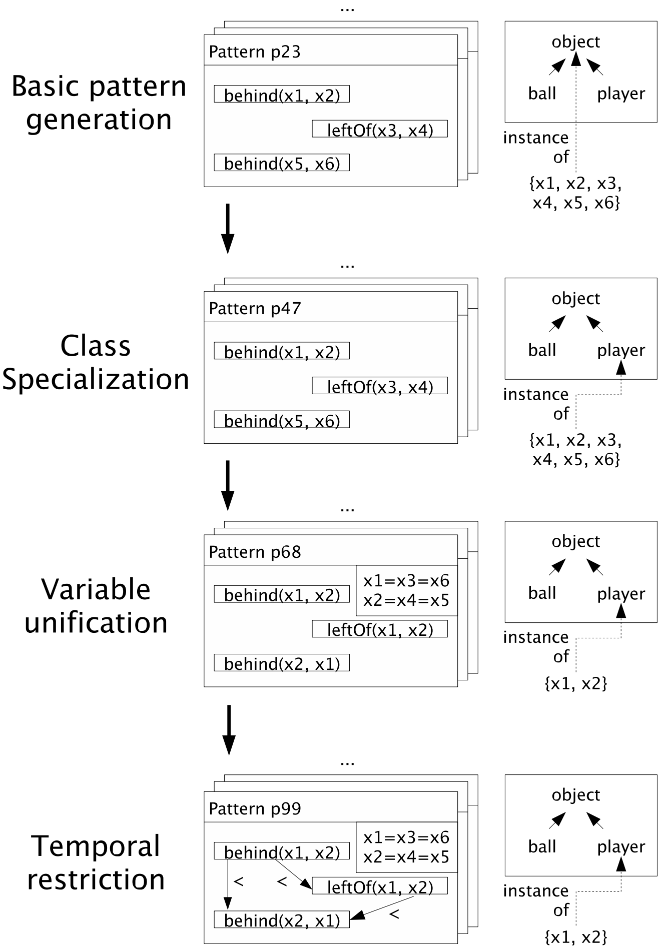


Fig. 4. Pattern generation

current class assigned to a variable is replaced by one of its subclasses. If a pattern is specialized by adding a new temporal relation for any pair of predicates in the pattern (which has not been constrained so far) a new temporal restriction can be added. A specialization through variable unification can be done by unifying an arbitrary pair of variables, i.e., the different predicates can be “connected” via identical variables after this step. Another specialization would be to replace a predicate by a more special predicate. These different specialization steps can be used at the top-down generation of rules (see Fig. 4).

In order to evaluate the patterns (and to just keep the *k* best patterns in the steps of the pattern learning algorithm) an evaluation function was defined. Right now our evaluation function takes six different values into account: relative pattern size, relative pattern frequency, coherence, temporal restrictiveness, class restrictiveness, and predicate preference. Further criteria can be added if necessary. The overall evaluation function of a pattern computes a weighted sum of the different single criteria.

The coherence gives information how "connected" the predicates in the pattern are by putting the number of used variables in relation to the maximum possible number of variables for a pattern. The temporal restrictiveness is the number of restricted predicate pairs in the pattern to the maximum number of time restrictions. The class restrictiveness is the relation of the number of variables which cannot be specialized anymore to the total number of variables. Further information about the evaluation criteria can be found in [13].

Compared to the work presented in [13] the representational power was extended by introducing classes and allowing temporal parallelism of time points and limiting the patterns to keep by discarding those patterns which create more special patterns without loss of frequency. With these extensions it is possible to restrict variables in patterns to specific classes, e.g., indicating that an object must be a player or a ball. Allowing identical time points for start and end time points of different predicates extends the number of learnable temporal relations (e.g., *meets* and *starts*; cf. [3]). Discarding general patterns which are not more frequent than their specialized patterns reduces complexity in further steps of the algorithm because the overall number of kept pattern decreases.

4.3 Situation and Behavior Prediction

Fig. 5 illustrates the idea how to use sequential pattern mining for situation and behavior prediction. We assume that the quantitative data perceived from the sensors is mapped to a qualitative representation. This should be a concise representation with all relevant information for behavior decision where similar

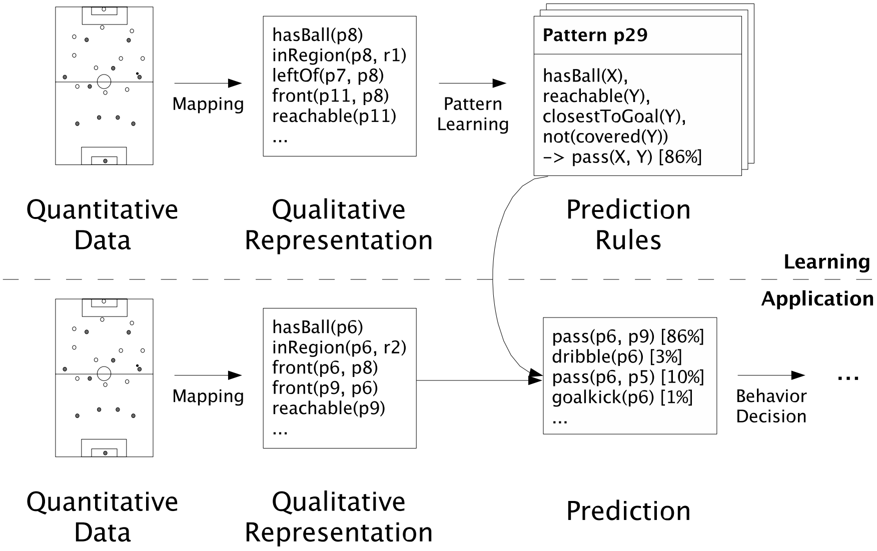


Fig. 5. Pattern learning for situation and behavior prediction

(quantitative) information is mapped to one qualitative class. This can include various information, e.g., about distances, regions, ball possession, score, time etc. This information is used as input for the learning algorithm. The start and end time points where different predicates are valid are known and can be used to set up a (temporal) sequence of predicates describing a dynamic scene. In the current setting learning is performed off-line. The result of learning is a set of prediction rules which give information what (future) actions or situations might occur with some probability if certain preconditions are satisfied. During a game these rules can be applied in order to predict what future actions or relations are likely to happen. This information can be used to perform a behavior decision on a better basis.

5 Application to Simulated Robotic Soccer

In the current status the qualitative mapping and the pattern learning approach are realized. The automated extraction and application of behavior prediction rules has to be done in future work. In order to evaluate our approach soccer games of the RoboCup 2D simulation league (TsinghuaAeolus vs. FC Portugal at June 22nd 2002 and FC Portugal vs. Puppets at June 21st 2002) were analyzed and a qualitative motion description as presented in Section 3 was created.

```
s; closerToGoal; 9;  8; 1                s; pass;         16; 18; 338
s; closerToGoal; 9; 11; 1                e; front;        16; 18; 338
s; front;        9;  7; 1                s; closerToGoal; 18; 19; 341
s; front;        9; 10; 1                s; closerToGoal; 18; 20; 341
s; front;        9;  2; 2                s; closerToGoal; 18; 21; 341
s; front;        9;  3; 2                s; closerToGoal; 18; 22; 341
e; front;        9;  2; 2                s; front;        18; 12; 341
e; front;        9;  3; 2                s; front;        18; 15; 341
s; closerToGoal; 9; 10; 3                e; closerToGoal; 16; 18; 341
s; closerToGoal; 9;  7; 4                e; closerToGoal; 16; 19; 341
e; front;        9;  7; 6                e; closerToGoal; 16; 20; 341
e; front;        9; 10; 6                e; closerToGoal; 16; 21; 341
s; front;        9;  8; 8                e; closerToGoal; 16; 22; 341
...                                      ...
```

Fig. 6. Sample input for the pattern mining algorithm

```
P_137
 front(X_7 X_8 )[r_1]
 front(X_15 X_16 )[r_2]
 pass(X_19 X_20 )[r_3]
 closerToGoal(X_25 X_26 )[r_4]

 [ X_7 = X_19; X_8 = X_20; X_15; X_16; X_25; X_26;]
 [X_7 = player X_8 = player X_15 = player X_16 = player
 X_19 = player X_20 = player X_25 = player X_26 = player ]
 [r_1 older r_2, r_1 younger-equal r_3
  r_1 younger-equal r_4, r_2 older r_3 ]
```

Fig. 7. Learned pattern

```
[r1] hasBall(X_1),
[r2] front(X_1, X_2),
[r3] uncovered(X_2)
 => [r4] pass(X_1, X_2)
[r1] older [r4],
[r2] older [r4],
[r3] older [r4]
```

Fig. 8. Example for a prediction rule

Fig. 6 shows a sample input for the learning algorithm. This input represents a sequence of predicates with their start and end time points. The first column identifies if a relation starts ("s") or ends ("e"). The second column is the name of the predicate (e.g., `pass`). The identifiers of the objects for which the predicate holds are the third and fourth value in each line. The last column denotes the time stamp of the start or end time point of this relation. Temporal parallelism can be recognized by identical values in the last column.

The learning algorithm performs a top-down generation of patterns. During learning the predicates in the input file are used to generate new patterns with variables. One example of a learned pattern can be seen in Fig. 7. It consists of four predicates (`front` [2x], `pass`, and `closerToGoal`). Among other information it says that a player (`X_7/X_19`) is in front of another (`X_8/X_20`) after a pass between those two was performed. The three segments below the predicates show the restrictions w.r.t. variable unification (`X_7 = X_19`), class information (`X_7 = player`), and temporal relations (`r_1 older r_2`). This pattern was learned from a snippet of the sequence from the first game. Fig. 8 shows an example of a prediction rule.

6 Conclusion

In this paper we presented an approach to situation and behavior prediction. A sequential pattern mining algorithm is applied in order to learn frequent patterns in the data. These patterns are then transformed into prediction rules which can be applied to estimate what is likely to happen in the future. One characteristic of the learning approach is high representational power with the potential of learning complex patterns with predicates and variables from relational and temporal data.

The drawback of the approach is the high complexity of the learning algorithm as discussed in [13]. The experiments support the assumption that without limiting the search space during pattern generation the algorithm cannot be used to learn complex patterns on-line due to time and space complexity. It is necessary to develop heuristics which allow an efficient learning of patterns without cutting off a large number of potentially good patterns.

We are currently working on improvements in order to handle the complexity of the learning task. In future work the performance of the learned patterns for predicting future behaviors and situations must be analyzed.

Acknowledgment

The work in this paper was partially funded by the Deutsche Forschungsgemeinschaft (DFG) in the SPP-1125.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., May 1993.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499, September 1994.
3. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.
4. J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.
5. I. Frank, K. Tanaka-Ishi, K. Arai, and H. Matsubara. The statistics proxy server. In T. Balch, P. Stone, and G. Kraetschmar, editors, *4th International Workshop on RoboCup*, pages 199–204, Melbourne, Australia, 2000. Carnegie Mellum University Press.
6. C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1–2):199–227, 1992.
7. D. Hernández, E. Clementini, and P. Di Felice. Qualitative distances. In *Proceedings of COSIT'95, LNCS 988*, Semmering, Austria, 1995. Springer.
8. F. Höppner. Learning temporal rules from state sequences. In *Proceedings of the IJCAI'01 Workshop on Learning from Temporal and Spatial Data*, pages 25–31, Seattle, USA, 2001.
9. Z. Huang, Y. Yang, and X. Chen. An approach to plan recognition and retrieval for multi-agent systems. In M. Prokopenko, editor, *Workshop on Adaptability in Multi-Agent Systems, First RoboCup Australian Open 2003 (AORC-2003)*, Sydney, Australia, 2003. CSIRO.
10. G. Kaminka, M. Fidanboylu, A. Chang, and M. Veloso. Learning the sequential coordinated behavior of teams from observation. In G. Kaminka, P. Lima, and R. Rojas, editors, *RoboCup 2002: Robot Soccer World Cup VI, LNAI 2752*, pages 111–125, Fukuoka, Japan, 2003.
11. K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases, SSD*, pages 47–66, Portland, Maine, 1995.
12. G. Kuhlmann and P. Stone. Progress in 3 vs. 2 keepaway. In *RoboCup 2003: Robot Soccer World Cup VII*. Springer, Berlin, 2004.
13. A. D. Lattner and O. Herzog. Unsupervised learning of sequential patterns. In *ICDM 2004 Workshop on Temporal Data Mining: Algorithms, Theory and Applications (TDM'04)*, Brighton, UK, November 1st 2004.
14. D. Malerba and F. A. Lisi. An ILP method for spatial association rule mining. In *Working notes of the First Workshop on Multi-Relational Data Mining*, pages 18–29, Freiburg, Germany, 2001.
15. H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

16. J. Mennis and J. W. Liu. Mining association rules in spatio-temporal data. In *Proceedings of the 7th International Conference on GeoComputation*, University of Southampton, UK, 8 - 10 September 2003.
17. A. Merke and M. Riedmiller. Karlsruhe Brainstormers - a reinforcement learning way to robotic soccer. In *RoboCup 2001: Robot Soccer World Cup V.* Springer, Berlin, 2002.
18. A. Miene, A. D. Lattner, U. Visser, and O. Herzog. Dynamic-preserving qualitative motion description for intelligent vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV '04)*, pages 642–646, June 14-17 2004.
19. A. Miene, U. Visser, and O. Herzog. Recognition and prediction of motion situations based on a qualitative motion description. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII, LNCS 3020*, pages 77–88. Springer, 2004.
20. P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*, number 2377 in Lecture Notes in Artificial Intelligence, pages 453–458, Berlin, 2002. Springer Verlag.
21. P. Riley and M. Veloso. Coaching advice and adaption. In *RoboCup 2003: Robot Soccer World Cup VII, LNCS 3020*, pages 192–204. Springer, Berlin, 2004.
22. P. Stone and R. S. Sutton. Scaling reinforcement learning toward robocup soccer. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
23. P. Stone and M. Veloso. A layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.
24. U. Visser, C. Drücker, S. Hübner, E. Schmidt, and H.-G. Weland. Recognizing formations in opponent teams. In P. Stone, T. Balch, and G. Kraetschmar, editors, *RoboCup 2000: Robot Soccer World Cup IV*, volume 2019 of *Lecture Notes in Computer Science*, pages 391 – 396, Melbourne, Australia, 2001. Springer-Verlag.
25. U. Visser and H.-G. Weland. Using online learning to analyze the opponent's behavior. In *RoboCup 2002: Robot Soccer World Cup VI, LNAI 2752*, pages 78–93. Springer-Verlag Berlin Heidelberg, 2003.

A Composite System for Real-Time Robust Whistle Recognition

Andrea Bonarini, Daniele Lavatelli, and Matteo Matteucci

Politecnico di Milano, Department of Electronics and Information,
Artificial Intelligence and Robotics Lab
{bonarini, matteucci}@elet.polimi.it

Abstract. In Robocup Middle-Size League (MSL) the challenge to recognize signals given by the referee by whistling has been introduced from this year as a way to reduce the interaction via radio-link. We present Whistle Recognizer (WR), a system able to recognize different whistling patterns, after a relatively short training done in advance. This composite system encompasses neural networks and more traditional information processing techniques. It demonstrated to be quite effective and can be easily integrated in a multi-thread control architecture, as the vast majority of those used in the league; thus, it candidates itself as a potential off-the-shelf module to be used by MSL teams not interested in research about signal processing and analysis.

1 Introduction

In Robocup Middle-Size League (MSL) the challenge to recognize signals given by the referee by whistling has been introduced from this year as a way to reduce the interaction via radio-link. Whistling has been introduced as a signal to be detected at the end of a half: detection is not compulsory, but the teams able to recognize this event will gain points for the challenge competition. Since many Robocup teams are not directly interested in signal processing and whistle analysis, an off-the-shelf package for this task would be clearly useful.

Whistle recognition has been faced in different domains to automatically detect interesting events or summarize multimedia data in sport events [1].

In this paper, we present Whistle Recognizer (WR), a composite system able to recognize different whistling patterns, after a relatively short training done in advance. This composite system (i.e., a system encompassing neural networks and more traditional information processing techniques [2]) is quite effective and can be easily integrated in any thread-based control architecture, as an off-the-shelf whistle sensor.

The tool we describe in the following sections recognizes whistle events and patterns from a raw audio data stream. With the term "pattern" we refer to a sequence of a predefined number of whistles at the rate of 1-2 whistles per second; we consider this as a possible future extension to the simple start/stop bit of information provided by a single whistle. A typical use of WR is to gather the raw audio stream using a microphone placed on the robot body and a cheap

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 130–141, 2006.

sound-card; output of the system is a message sent to the controlling process to communicate the whistle type: short single, long single, or multiple, with multiplicity.

Since whistles and referees are different from game to game, a key issue for a whistle recognition system is fast tuning and adaptation in order to reduce set-up time and increase robustness. This is obtained by implementing the recognition system using a composite approach to first extract the characteristic features from the signal, and then applying a neural classifier to detect whistling events. The system is integrated with a learning tool used to set recognition parameters basing on a short recorded sample of whistle and background noise. Section 2 gives a detailed description of the system architecture and algorithms; all design choices are explained there, while the following section gives a summary of the learning tool.

2 System Architecture

As introduced in the previous section, the system is based on a classical digital signal processing algorithm to extract signal features followed by a neural stage and an event counter. Figure 1 shows the schema of this architecture. The signal is acquired by a commercial sound-card (section 2.1) connected to a microphone; the periodogram is computed from the raw signal by a fast algorithm to extract features related to the spectral power of the whistle. To improve classification capabilities, a frequency mask has been introduced to give only the interesting samples in input to the neural stage (section 2.3). Finally, a non-linear perceptron (section 2.4) recognizes the presence of the whistle signal and passes this information to an output event counter, which produces the recognition message.

2.1 Data Acquisition and Feature Extraction

Raw data is acquired from the computer sound-card and signal level is adjusted by acting on the pre-amplifier through the IGAIN feature. Considering a whistle power spectrum concentrated below the 4 Khz we used a sampling frequency of 8 KHz and a quantization of 8 bit/sample proved to be adequate for the job. The

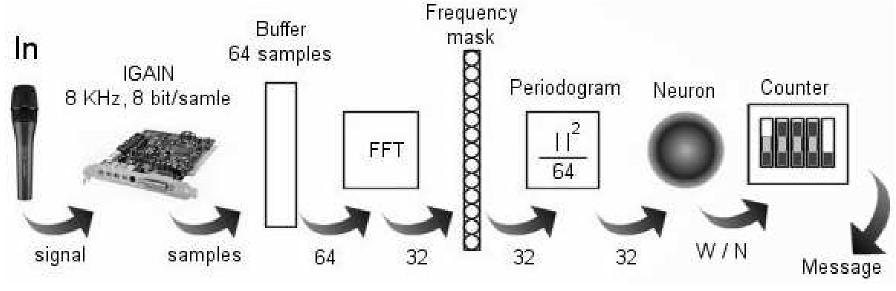


Fig. 1. System architecture

elemental block on which the system operates to detect a whistle is a 64-sample sequence, yielding a time resolution of 8 ms.

2.2 The FFT and the Periodogram

The features used by the neural classifier are taken from the signal periodogram and, as it can be expected, this computation is the system bottleneck; so, we have designer it with special attention to to make it as fast as possible. Let us consider the Discrete Fourier Transform (DFT) formula [3, 4]:

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{-nk} \tag{1}$$

where N is the number of samples, n is the time index (from 0 to $N-1$), k is the frequency index (from 0 to $N-1$) ,$W_N = e^{-j\frac{2\pi}{N}}$ and X_k is an N-sample sequence representing the original sequence x_k in the frequency domain. In general, the so-called Fast Fourier Transform (FFT) algorithm is used instead of the basic formula, to lower the N^2 complexity to $N \log N$. This is done by subdividing the original sequence in two sub-sequences, x_{2n} and x_{2n+1}, and then applying the algorithm recursively on them. In the particular case of N power of 4, the original sequence can be directly subdivided in 4 sub-sequences, so obtaining another 25% improvement ($0.75N \log N$). In this case the situation is the following:

$$\begin{bmatrix} X_k \\ X_{k+\frac{N}{4}} \\ X_{k+2\frac{N}{4}} \\ X_{k+3\frac{N}{4}} \end{bmatrix} = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -j & -1 & +j \\ +1 & -1 & +1 & -1 \\ +1 & +j & -1 & -j \end{bmatrix} \begin{bmatrix} G_k \\ W_N^{-k} H_k \\ W_N^{-2k} L_k \\ W_N^{-3k} M_k \end{bmatrix} = \underline{W} \begin{bmatrix} G_k \\ W_N^{-k} H_k \\ W_N^{-2k} L_k \\ W_N^{-3k} M_k \end{bmatrix} \tag{2}$$

where G_k,H_k,L_k and M_k are the following sequences:

$$G_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n} W_{\frac{N}{4}}^{-nk} \qquad H_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n+1} W_{\frac{N}{4}}^{-nk}$$

$$L_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n+2} W_{\frac{N}{4}}^{-nk} \qquad M_k = \sum_{n=0}^{\frac{N}{4}-1} x_{4n+3} W_{\frac{N}{4}}^{-nk}$$

Let us point out that the computation of the $\underline{W}$ matrix does not need any multiplication, so, given G_k,H_k,L_k and M_k, the X_k sequence can be computed performing only 3 complex multiplications ($W_N^{-k}*$ and so on), while the standard 2-subsequences FFT algorithm will have needed 4 complex multiplications. In doing so, we obtain a 25% gain on the total number of multiplications and this is the most important reason to select a 64-samples sequence.

The FFT of the 64-samples sequence can be easily computed with 3 stages of recursion. In fact, the 64-samples sequence is subdivided in 4 16-samples subsequences; each 16-samples sequence is subdivided in 4 4-samples subsequences, whose FFTs are then atomically calculated.

To drastically reduce the average computation time, another property of the DFT can be used: symmetry. As x_n (the elemental 64-samples sequence) is a sequence of real numbers, we can build a 64-samples sequence of complex numbers in the following way:

$$s_n = x_n + jy_n \tag{3}$$

After the calculation of the FFT of s_n, S_k, the X_k and Y_k sequences can be rebuilt using only sums and subtractions, by the following formulas:

$$\Re(X_k) = \frac{1}{2}[\Re(S_k) + \Re(S_{-k})], \quad \Im(X_k) = \frac{1}{2}[\Im(S_k) - \Im(S_{-k})] \tag{4}$$

$$\Re(Y_k) = \frac{1}{2}[\Im(S_k) + \Im(S_{-k})], \quad \Im(X_k) = \frac{1}{2}[\Re(S_{-k}) - \Re(S_k)] \tag{5}$$

This reduces by 50% the average number of multiplications, as we can get two DFTs by applying a single FFT, and, once the FFT has been calculated, the periodogram is given by:

$$P_k = \frac{|X_k|^2}{64}, \quad with \;\; k = 1, \ldots, 32. \tag{6}$$

2.3 Frequency Mask for Feature Selection

When the periodogram is computed, many of its samples represent frequencies which should not be used for recognition, as they are far from the whistle power spectrum and do not contain useful information. All these frequencies act as noise for the classifier and should be reduced to improve classification performances. So, the problem is: how many and which samples should be considered?

Since the whistle signal has a bandwidth of about 30÷40 Hz centered around its average frequency and the periodogram has a resolution of $\frac{f_{Nyquist}}{32} = \frac{4KHz}{32} = 125Hz$, a window of 3 samples can be adequate. A pure whistle signal can, in fact, cause no more than 2 samples in the periodogram to raise, in absence of background noise.

However, listener and source can move, and we need to face also the Doppler effect. Suppose a stationary source is generating sound waves with frequency $\overline{f}$ and wavelength $\overline{l} = v/\overline{f}$, being v the speed of sound. A stationary observer at a certain distance from the source will hear a sound with pitch $\overline{f}$. $\overline{f}$ times each second the observer sensor will be pushed in and pulled out as pressure crest and pressure trough reach it. The time period between two consecutive crests is $T = 1/\overline{f}$. Assume the observer in this case is a robot and starts driving away from the source. Assume that at time t_1 a pressure crest reaches the “robot ear” at position x. The next crest will be at position x at time $t_1 + T$, but the “ear” will no longer be there. In this case the crest has to travel an extra distance before it reaches the observer and this takes an extra time interval Δt. The time interval between subsequent crests reaching the ear of the observer is now $T' = T + \Delta t$.

While the observer has traveled a distance $\Delta x = v_o \cdot (T + \Delta t)$, at speed v_o, the wave has traveled a distance $\Delta x + \overline{l} = v \cdot (T + \Delta t)$. Therefore, using $\overline{l} = v/\overline{f} = v \cdot T$, we have $v_o \cdot T + v_o \cdot \Delta t + v \cdot T = v \cdot T + v \cdot \Delta t$, or $\Delta t = v_o \cdot T/(v - v_o)$. Thus, we obtain:

$$T' = T + v_o \cdot T/(v - v_o) = v \cdot T/(v - v_o), \tag{7}$$

$$f' = \overline{f}(v - v_o)/v. \tag{8}$$

The period has increased, the apparent frequency of the wave has decreased, the pitch has decreased. If the observer is driving towards the source, then the time interval between successive crests reaching the sensor will be shorter than T. The apparent frequency of the sound wave reaching the observer is thus

$$f' = \overline{f}(v + v_o)/v. \tag{9}$$

The perceived pitch of a sound wave also changes if the observer is stationary and the source is moving. Then the apparent frequency of the sound wave reaching the observer when the source is moving towards him with speed $v_s$is:

$$f'' = \overline{f}v/(v - v_s) \tag{10}$$

Whenever the source and the observer move with respect to each other, the wavelength of the sound reaching the ear will be Doppler shifted:

$$f = f''(v \pm v_o)/v = \overline{f}v/(v \mp v_s) \cdot (v \pm v_o)/v = \overline{f}(v \pm v_o)/(v \mp v_s). \tag{11}$$

The Doppler effect, computed with reasonable parameters (i.e., robot speed = 5 m/s, whistle speed = 2 m/s, as worst case hypothesis), gives a frequency shift of about 60 Hz at whistle average frequency (about 3 KHz). For this reason, the observation window can be optionally extended from 3 to 5 samples to compensate the Doppler effect.

The next problem to face is now: which frequencies should be considered? Different whistles (or different environments) will lead to different choices of frequencies and this calls for an adaptive system to reduce set-up time. To face this issue, the concept of data separation has been introduced. Given a few examples of whistle signal and background noise, these can be used to determine which are the frequencies (i.e., the periodogram samples) that better characterize the whistle sound and make it possible to distinguish it from the background noise. Data separation gives a measure of how many samples can be correctly recognized with a fixed threshold by observing a periodogram sample.

Figure 2, on the left, shows the situation for samples 21, 22, 23, and 24 of the periodogram. Whistle examples (light gray) are obtained by transforming sequences marked as "whistle" by a supervisor; the same applies to noise (dark gray) examples. For each sample, all the examples are classified in two sets: W(histle) and N(oise); so, for a generic sample, we have the situation shown in Figure 2, on the right. Data separation (DS) for the sample k is defined as:

$$DS_k \triangleq 1 - \frac{W_k \cap N_k}{W_k \cup N_k} \tag{12}$$

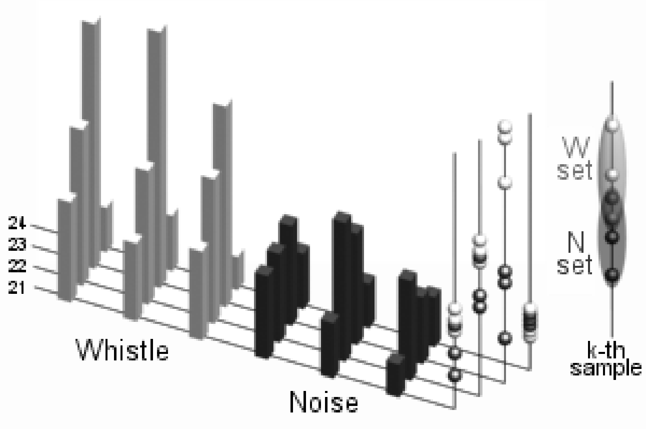


Fig. 2. Data separation

Once DS has been calculated for all 32 periodogram samples, the 3 samples with the highest DS values are selected and marked as "1" in the frequency mask. All other samples are marked as "0" in the frequency mask.

2.4 Perceptron with Hysteresis and Event Counter

The inputs used in the neural classifier are 3 (or 5, if Doppler compensation is enabled) samples of the periodogram. Being reasonable to consider low values of these samples as belonging to noise and high values to whistle signal, we can use a hyper-ellipsoid as separation surface for the perceptron. When the 3 (or 5) inputs identify a point internal to the hyper-ellipsoid, the sample will be recognized as noise, otherwise it will be recognized as whistle.

To avoid spurious commutations in noise-to-whistle and whistle-to-noise transients, we have introduced hysteresis, i.e., we implemented a recurrent perceptron that uses its previous state as switching condition. Figure 3 shows the non-linear perceptron with hysteresis used in the system. Calling s_{01} the noise-to-whistle

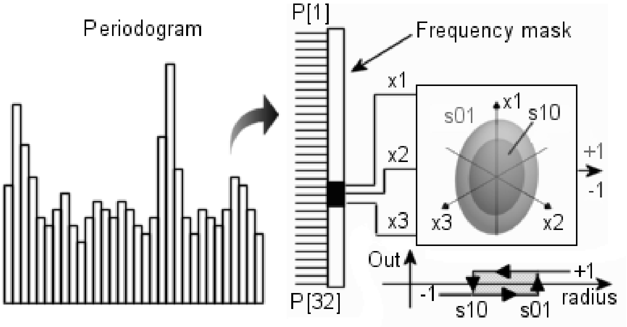


Fig. 3. Perceptron with hysteresis

(turn-on) threshold and s_{10} the whistle-to-noise (turn-off) threshold, the output of the neuron is:

$$Out = \begin{cases} if\ \ PreviousState = -1 \begin{cases} +1 & if\ \ \sum_{n=0}^{32}(\frac{x_n}{W_n})^2 > 10^{\frac{s01}{10}} \\ -1 & \quad Otherwise \end{cases} \\ if\ \ PreviousState = +1 \begin{cases} -1 & if\ \ \sum_{n=0}^{32}(\frac{x_n}{W_n})^2 \leq 10^{\frac{s10}{10}} \\ +1 & \quad Otherwise \end{cases} \end{cases} \tag{13}$$

where +1 is the output chosen to denote whistle sound detection, and -1 is the output corresponding to "noise", s_{01} and s_{10} are expressed in dB and *PreviousState* is the previous perceptron's output.

During the learning procedure, thresholds s_{01} and s_{10} are forced to 0 and the best separation surface can be found by applying the learning rule:

$$\begin{array}{l} E = \frac{1}{2}(target - output) \\ W_k = W_k - \gamma \cdot x_k \cdot E \end{array} \tag{14}$$

where γ is an appropriate learning rate, *target* is the desired output, as specified by the supervisor, *output* is the output of the perceptron to the current sample, and E is the error, which can be +1, 0 or -1. For instance, if the perceptron outputs +1 (whistle) while no whistles are being blown, we have:

- *outupt* $= +1$ (The perceptron's output)
- *target* $= -1$ (The correct output, provided by the supervisor).

This results in error E to be -1; this means that a noise point, which should be internal to the ellipsoid, has been actually classified as external. Since W_k is the ellipsoid radius on the k axis, it must be raised to let the ellipsoid include the example, in order to classify it as a noise sample. This is done for those axes of the ellipsoid (from W_1 to W_{32}) that are not filtered out by the frequency mask.

Doing the dual reasoning when error is +1, and combining the two cases, we obtain the learning rule 14.

The perceptron output refers to the analysis of 64 samples (8 ms) of data, and states whether this block of audio signal contains a whistle sound. Time elapsed between two consequent readings of elementary audio blocks is customizable, and it will be called "timestep" in the following. The perceptron is thus designed to produce an "event" (+1:whistle or -1:noise) every timestep; more precisely, the FFT is called once on a pair of 64-samples sequences, so we obtain a pair of events every 2·timestep.

A "counter" module of WR has to translate the sequences of events into one of the following messages:

- Short whistle
- Long whistle
- Multiple whistle, which multiplicity is n

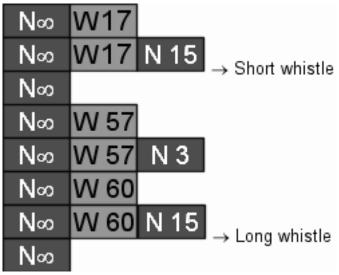


Fig. 4. A short and a long whistles

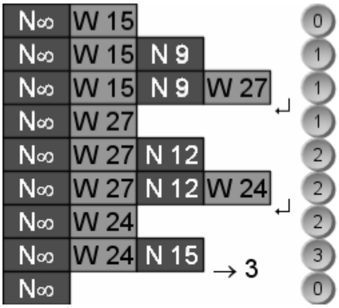


Fig. 5. A triple whistle

Table 1. Settings used in the examples

$Parameter$	$Value$
$ShortIfLess$	30 steps
$LongIfGreater$	50 steps
$IntervalBetween$	15 steps
$MinWhistleLength$	5 steps
$MinNoiseLength$	5 steps

The following parameters, expressed in numbers of timesteps, have to be provided by the user in order to define whistle timing specifics:

- Maximum length of a short whistle ($ShortIfLess$)
- Minimum length of a long whistle ($LongIfGreater$)
- Maximum interval length between two consecutive whistles ($Interval$ $Between$)
- Minimum whistle length (shorter whistles are filtered)
- Minimum noise length (shorter noises are filtered)

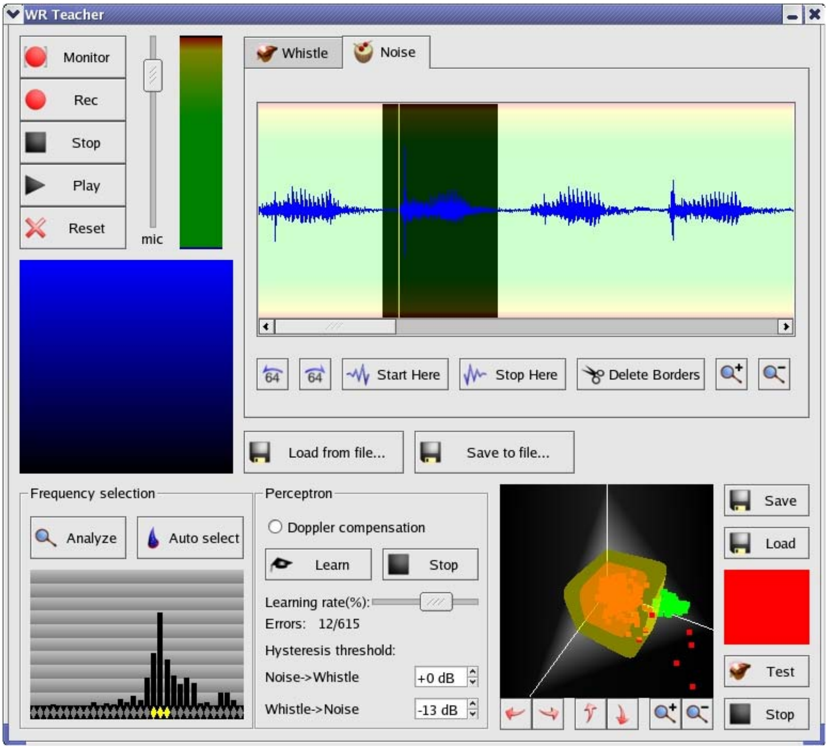
Events output by the perceptron (Whistle or Noise) are accumulated into slots, which are groups of subsequent events of the same type. Slots are represented with colored boxes in figures 4 and 5. The generated slots are accumulated into an event stack, as shown in figures 4 and 5: each line corresponds to the status of the stack up to the occurrence of an event of different type.

The starting slot is "noise" from a sufficiently long time (N_∞). So, for instance, in figure 4, we have on the first line 17 "whistle" events (represented by the W17 block). Then, on the second line, we have the situation of the stack once the subsequent 15 "noise" events (N15) have been detected. Due to the settings for this example, the "W17" slot is not filtered out (17 is greater than 15, the minimum whistle length parameter), so the subsequent noise events are accumulated up to the maximum length of an interval between two whistles in a multiple whistle (in this example, 15), as defined by the user (see settings in Table 1). The stack situation shown on the second line is thus recognized as a single whistle, and the corresponding notification message is issued. Then we have the recognition of a long whistle. Notice that the number of noise events detected (3) is less that the minimum required to interrupt the sequence of whistle events, so the N3 sequence is filtered out.

Figure 5 shows a triple whistle; the number on the right is the whistle multiplicity. Every time a new noise event occurs and the current whistle has not to be filtered, multiplicity is incremented; in order to save space in the stack, each pair of whistle-noise slots is eliminated when a new whistle slot is completed (see the 3rd and 4th row in figure 5: "W15" and "N9" slots are eliminated and the "W27" slot becomes the second slot); the system knows how many whistles were blown in the past, memorized as multiplicity. As with the previous example, the correct event is notified after a "N15" slot; here, the message is the multiplicity of the whistle.

3 The Learning Tool

A software tool is provided to let the user set system parameters and neuron weights. The higher part in the tool main window (Figure 6) is a sound recorder with two buffers: one for whistle recording and one for background noise



○ Doppler compensation

Learning rate(%):

Errors: 12/615

Hysteresis threshold:

Noise->Whistle +0 dB

Whistle->Noise -13 dB

Fig. 6. The learning tool

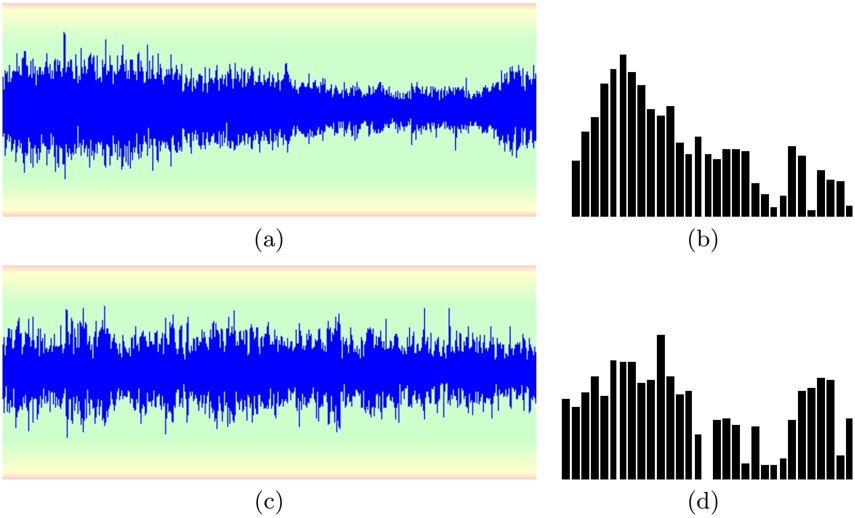


Fig. 7. Background noise (a) and whistle (c) signals captured from outside the field during a match. In (b) and (d) respective periodograms are reported.

recording. This separation is a simple way to let the user be the supervisor of the acquisition process.

The lower part of the window integrates a set of tools to control and test the perceptron. We can see a plot of data separation (on the left), which helps the user in the selection of the frequency mask, a section dedicated to the perceptron learning with a 3D visualization of the separation surfaces (in the center), and a test section (on the right).

4 Results

The system has been tested both with samples recorded from outside the field during competitions and on the robot during execution in our laboratory in Milano–Bovisa[1]. In Figure 7(a) and (c) you can notice respectively the noise and whistle signals captured during a match in Padova during the 2003 Robocup competition. In this case, we do not have a pure whistle signal so we trained the recognition system using background noise and a quite noisy whistle. Even in these conditions, the system has been able to recognize perfectly the two short whistles and the long whistle present in the sample.

Whenever we are interested on on-board whistle recognition, we should notice that the highest level of background noise is caused by the robot motors and body, which strongly vibrates while the robot is moving. This is clear in Figure 8(a) where noise has been captured on-board during robot operation.

[1] All samples used for experimental validation in this paper are available from `http://robocup.elet.polimi.it/MRT/WR.html`.

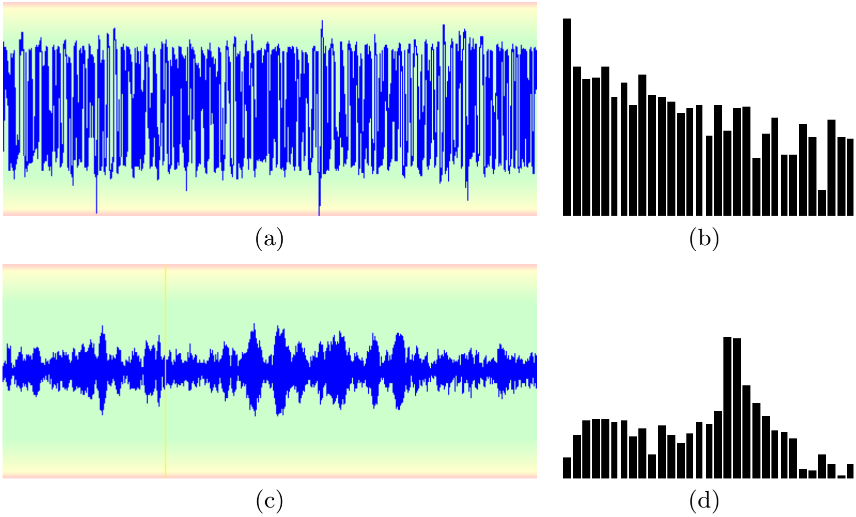


Fig. 8. Background noise (a) during execution and pure whistle (c) signals captured from the robot. In (b) and (d) respective periodograms are reported.

This results in a very strong white noise[2], which could be partially lowered by mechanically isolating the microphone from the structure. In this case, we could acquire the pure whistle signal, reported in Figure 8(c), but it has an intensity lower than noise. If the whistle-to-noise and noise-to-whistle thresholds are adequately set, the system proved to have a good behavior in rejecting noise in this especially adverse situation, resulting in a 70% correct classification rate.

Eventual shocks of the robot during the match are well filtered by adequately setting the whistle and noise minimum lengths. Human voice frequencies have been measured and they are quite distant from the whistle characteristic frequency, so this kind of background noise did not cause any problem. We also tested the system using a sample with a spurious whistle coming from a different field and it was able to reject it.

5 Conclusions

Aim of this paper was to present the design and features of WR, a system enabling robots to detect the sound of the referee whistle during a match. A composite approach, based on a spectrum analyzer followed by a neural output stage and a counter was chosen to achieve the goal providing a flexible and easily tunable system. Particular care has been taken to design a system as simple and fast as possible. We have also implemented a software tool to support the user to tune the system. During the tests, the system proved to be fast and accurate, even in presence of quite strong background noise.

[2] Actually the recorded signal resemble more to a pink noise due to the low-pass effect of the microphone.

Another important feature to be mentioned is that the system requires a very low CPU load; in fact, the whole process (FFT, perceptron and counter) takes about 150 μsec to complete on a P4 2.1 MHz, while it is called, with the default timestep, every 40 ms.

Acknowledgments

This work has been partially supported by MADSys, P.R.I.N. 2003 project "Software technologies and knowledge models for the design, implementation and experimentation of robotic multi-agent systems in real environments", co-funded by the Italian Ministry of Instruction, University, and Research.

References

1. Tjondronegoro, D., Chen, Y.P.P., Pham, B.: Integrating highlights for more complete sports video summarization. IEEE MultiMedia (2004) 22–37
2. Alippi, C., D'Angelo, G., Matteucci, M., Pasquettaz, G., Piuri, V., Scotti, F.: Composite techniques for quality analysis in automotive laser welding. In: Proceedings International Symposium on Computational Intelligence for Measurement Systems and Applications, Lugano, Switzerland (2003)
3. Oppenheim, A.V., Schafer, R.W.: Digital Signal Processing. Prentice Hall, Englewood Cliffs, NJ, USA (1975)
4. Oppenheim, A.V., Schafer, R.W.: Discrete-Time Signal Processing. Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ, USA (1989)

Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization

Martin Lauer, Sascha Lange, and Martin Riedmiller

University of Osnabrück, Institute of Cognitive Science and
Institute of Computer Science, 49069 Osnabrück, Germany
{martin.lauer, sascha.lange, martin.riedmiller}@uos.de

Abstract. The paper develops a new approach for robot self-localization in the Robocup Midsize league. The approach is based on modeling the quality of an estimate using an error term and numerically minimizing it. Furthermore, we derive the reliability of the estimate analyzing the error function and apply the derived uncertainty value to a sensor integration process. The approach is characterized by high precision, robustness and computational efficiency.

1 Introduction

Autonomous robots need to know their position and heading to be able to solve a given task like driving to a certain position. Especially in the Robocup Midsize league reliable position estimates are essential for higher level behavior like path planning, strategy and multi-agent coordination. Since autonomous robots cannot refer to global sensors which are fixed with respect to a global coordinate system but all sensors are on-board they need a procedure of self-localization, i.e. an algorithm to calculate their position and heading.

In this paper, we focus on a camera-based self-localization approach for the Robocup Midsize league. Three main difficulties have to be faced: (a) the self-localization process must be robust. The soccer field is not encircled by a board that allows to distinguish objects inside and outside the field like spectators. This may lead to misinterpretations of the visual information.

(b) Position estimates must be accurate: images of standard camera systems exhibit a poor resolution of objects located more than a few meters away. Hence, distance estimates are very noisy. Furthermore, the dynamics of the game with large accelerations and collisions between robots leads to vibrations that further affect the quality of self-localization (see fig. 1).

(c) The self-localization approach needs to be computationally efficient since the robot control program must satisfy strict real time conditions: our goal is to reduce the computation time to less than 15 milliseconds.

Mainly three approaches [10] have been used so far to solve the self-localization task: (a) the use of colored landmarks combined with geometrical calculation, e.g. [4], (b) the detection of white field markings combined with a Hough-transform, e.g. [6], and (c) the detection of landmarks or field markings combined with a sequential importance sampling [3] approach like *Particle filtering* [2].

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 142–153, 2006.

All of these approaches have some merits but none of them solves all objectives of self-localization satisfactorily: approaches using landmarks are easily mislead by colored objects outside the field. Secondly, the large size of the field ($8 \times 16m$) and the small number of landmarks restricts the use of landmarks.

Using the Hough-transform needs the calculation of a three dimensional accumulator array. Hence, a lot of calculation is done for positions of no interest and any increase in precision implicates a heavy increase of computation time.

Monte Carlo approaches like Particle filtering also spend a lot of time in evaluating positions of no interest since they follow a blind search paradigm. In experiments we made with a Particle filtering approach of our Robocup team [7] we observed that approximately 98% of the examined positions did not contribute to the final position estimate since positions are evaluated even if neighboring places have already been evaluated as poor estimates.

We therefore want to propose a new algorithm for robot self-localization that overcomes the problems stated before and that fulfills all requirements: robustness, accuracy and efficiency. It is based on guided update steps modeling the localization problem as an error minimization task and using an efficient numerical minimizer. Additionally, we derive a measure of reliability of the calculated position analyzing the structure of the error function so that we can apply a stochastic sensor fusion process that increases the accuracy of the estimate.

An extension of the algorithm described in a further section also allows to solve the global localization problem, i.e. to find a robot's position without any prior knowledge. Finally, we compare the new approach with an existing implementation of Particle filtering for self-localization.

We assume the robots being equipped with an omnidirectional color camera on top that perpetually takes pictures from the field area around the robot. We further assume the case of omnidirectional driving capabilities although the calculations can also be done for a differential drive in simplified form.

2 Interpreting the Pictures from the Camera

2.1 Image Preprocessing

The pictures from the camera (fig. 1) are preprocessed using a detector of line points based on an efficient search along pre-defined radial scanlines. A description of this approach can be found in [7]. The result of preprocessing is a list of positions relative to the robot position and robot heading where scanlines intersect white field markings that have been observed in the image. In the following, we will call these points *detected line points* or, simply, *line points*.

An example of such a list is given by the gray circles in figure 4 (left). The detected line points are not linked, i.e. the list does not preserve the neighborhood relationship of line points that belong to the same line.

2.2 Matching Visible Information with Position Estimates

To find the position and heading of the robot with respect to the information we get from image preprocessing we define an error function that describes the

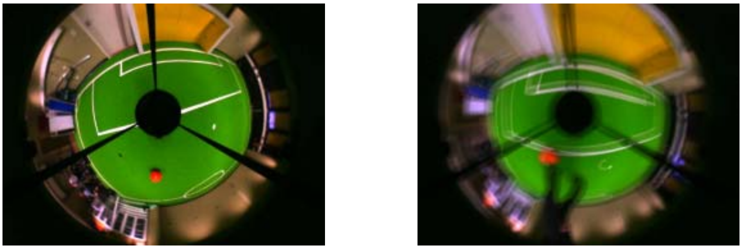


Fig. 1. Pictures taken by the omnidirectional camera. Left: a neat picture taken when the robot was not moving. Right: a blurred picture affected by vibrations when the robot was moving. All field markings are blurred and some of them occur twice.

fitness of a certain estimate. The idea is, that the detected line points and the known field markings match as best as possible if we assume the true robot position and robot heading. Hence, maximizing the fitness (=minimizing the error) yields the best estimate.

Let $(\boldsymbol{p}, \phi)$ be a pair of a possible robot position $\boldsymbol{p} = (p_x, p_y)$ and heading ϕ in a global coordinate system. The list of detected line points is given relative to the robot's pose as vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ (see fig. 2). Its position in world coordinates therefore is given by $\boldsymbol{p} + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i}$. Minimizing the error between detected line points and true field markings means to solve:

$$\underset{\boldsymbol{p},\phi}{\mathit{minimize}} \;\; E := \sum_{i=1}^{n} \mathit{err}(d(\boldsymbol{p} + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i})) \tag{1}$$

The mapping $d(\cdot)$ gives the distance from a certain point on the field to the closest field marking. It is continuous and piecewise differentiable and can be calculated from the knowledge of the field markings that are defined in the Robocup rules.

err is an error function that punishes deviations between detected line points and the model lines. The squared error function $e \mapsto \frac{1}{2}e^2$ which is standard for many applications is not appropriate for the given task since it is not robust with

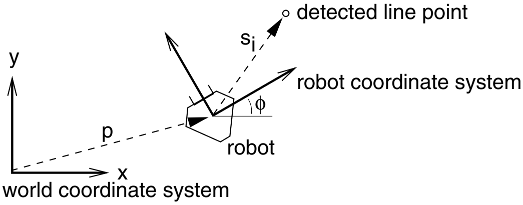


Fig. 2. Sketch of the fixed world coordinate system, the robot relative coordinate system and a vector s_i pointing to a detected line point

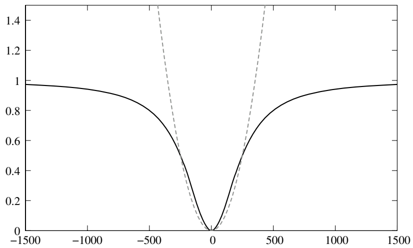


Fig. 3. Comparison of the squared error function (dashed line) and the more robust M-estimator $e \mapsto 1 - \frac{c^2}{c^2+e^2}$ (solid line)

respect to outliers [9]. Due to image noise and imperfect image preprocessing we are faced with a substantial amount of erroneously detected line points that would distort the estimate. Instead, we use the error function $e \mapsto 1 - \frac{c^2}{c^2+e^2}$ with parameter $c \approx 250$, see fig. 3. This error function is very similar to the squared error function for errors $e \leq c$ and is bounded above by a constant for larger errors, thus the influence of outliers onto the estimate is bounded.

Figure 4 (right) shows the error function for a certain set of detected line points. Obviously, the error function exhibits a large number of local minima. Due to the non-linearity of the minimization problem (1) we cannot analytically calculate its solution but we need a numerical minimizer.

Since d is almost everywhere differentiable we can build its gradient almost everywhere and interpolate the gradient at the non-differentiable places. Hence, we can use gradient descent to solve (1). Due to quick convergence and high robustness we use 10 iterations of *RPROP* to solve the minimization task [8]. *RPROP* was originally developed as learning rule for multi layer perceptrons but it can also be used to solve other types of unconstrained optimization problems.

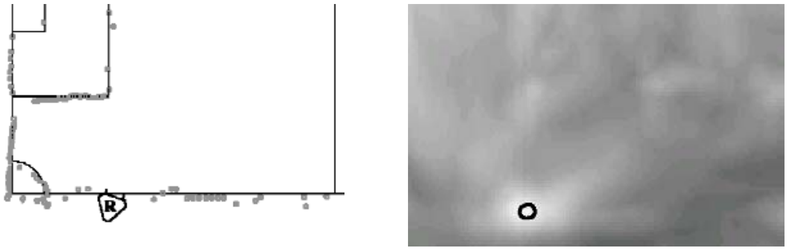


Fig. 4. Left: The set of line points (gray circles) and the field markings (solid lines) for an optimal position estimate. The estimated robot position and heading is indicated by the symbol "R". Right: A graylevel plot of the error function for the same clipping as in the left-hand figure. The 3D-error function was projected onto the two-dimensional field assuming optimal heading of the robot. Dark areas indicate positions with large error, bright areas positions with small error. The black circle indicates the optimal position estimate. The error function exhibits a distinctive global minimum.

Using the idea of error minimization we can realize the draft version of a robot localization algorithm:

1. start with a known position estimate
2. calculate the movement of the robot since the latest update of the position estimate and add it to the latest estimate
3. optimize the position applying the error minimization approach
4. repeat 2. and 3. every time a new camera image is received

2.3 Dealing with the Aperture Problem

Figure 4 shows a situation with a distinctive global minimum, i.e. the optimization task is well-posed and all parameters can be estimated reliably. Unfortunately, situations occur in which the optimization task is ill-posed due to a small number of line points or a poor structure of the line points. Such a situation is depicted in figure 5: the robot is located next to the touch-line of the field and all line points refer to the touch-line. Hence, the distance to the touch-line, i.e. the y-coordinate, can be estimated very reliably while the x-coordinate remains vague. The error function is characterized by a long valley of similar small values.

To tackle the aperture problem we have to recognize three possible situations: (a) the error function exhibits a distinctive global minimum. Hence, we can estimate $\boldsymbol{p}$ and ϕ reliably. (b) The error function is completely flat due to a small number of line points. Thus, we cannot estimate any parameter. (c) The error function exhibits a valley structure around the minimum. Here, we can estimate parameters robustly which refer to a coordinate axis orthogonal to the valley but we cannot estimate parameters that refer to a coordinate axis parallel to the valley. E. g. in fig. 5 we can estimate p_y and ϕ but not p_x.

To determine the structure of the error function around the minimum we propose the analysis of the second order derivatives of the error function: the value of $\frac{\partial^2 E}{(\partial p_x)^2}$ is small in the case of a valley parallel to the x-axis and in a

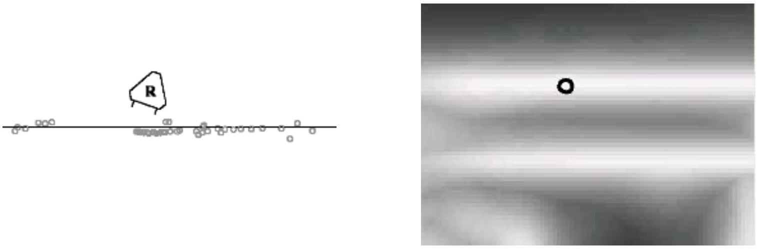


Fig. 5. Aperture problem: The left hand figure shows the mapping of the line points onto the known field markings while the right hand figure shows the error function in the same way as in figure 4. The error function shows a valley of small error values. Hence, the position estimate is very reliable with respect to the y-coordinate but unreliable with respect to the x-coordinate.

completely flat case while it is large if E shows a distinctive minimum with respect to the x-axis. Analogously we can analyze $\frac{\partial^2 E}{(\partial p_y)^2}$ and $\frac{\partial^2 E}{(\partial \phi)^2}$.

From a practical point of view the calculation of the second order derivatives simplifies: The function d is build out of the line markings on the field. Most of them are lines parallel to the coordinate axis, only the corner arcs and the center circle are no straight geometrical objects. Hence, the function d is piecewise linear in most parts of the field and its second order derivatives are zero in these areas. Using this simplification we get:

$$\frac{\partial^2 E}{(\partial p_x)^2} \approx \sum_{i=1}^{n} err''(s_i) \cdot (\frac{\partial d(s_i)}{\partial x})^2 \tag{2}$$

$$\frac{\partial^2 E}{(\partial p_y)^2} \approx \sum_{i=1}^{n} err''(s_i) \cdot (\frac{\partial d(s_i)}{\partial y})^2 \tag{3}$$

$$\begin{aligned}\frac{\partial^2 E}{(\partial \phi)^2} \approx \sum_{i=1}^{n} \Big(& err''(s_i) \cdot (\frac{\partial d(s_i)}{\partial x}(-\sin\phi \quad -\cos\phi)\boldsymbol{s}_i + \frac{\partial d(s_i)}{\partial y}(\cos\phi \quad -\sin\phi)\boldsymbol{s}_i\Big)^2 \\ & + err' \cdot (\frac{\partial d(s_i)}{\partial x}(-\cos\phi \quad \sin\phi)\boldsymbol{s}_i + \frac{\partial d(s_i)}{\partial y}(-\sin\phi \quad -\cos\phi)\boldsymbol{s}_i)\Big)\end{aligned} \tag{4}$$

where err' and err'' denote the first and second order derivative of err.

Unfortunately, the error function $e \mapsto 1 - \frac{c^2}{c^2+e^2}$ used in (1) is not completely positive definite and therefore the curvature criterion may be mislead, i.e. the second order partial derivative may be small also the minimum is distinctive. However, this problem is caused only from outlying observations since the error function is positive definite in the interval $(-\frac{c}{\sqrt{3}}, \frac{c}{\sqrt{3}})$. To avoid this problem we adopt the following artifice: in (2)–(4) we replace the original error function by the squared error function $e \mapsto \frac{1}{2}(\frac{e}{c})^2$ and ignore outlying observations. Hence, E becomes positive definite and the curvature criterion yields sound results.

3 Tracking and Smoothing

The approach described so far estimates the robot position and heading that matches optimally to the information extracted from the camera image. Due to vibrations of the robot, especially in the case of high velocity or due to collisions, this position is affected by a reasonable amount of noise and inaccuracy (see fig. 1 (right)). The dotted line in figure 6 (left) shows an example of a trajectory build out of the positions calculated only from the image information. Obviously, the noise of the self-localization process is severe.

To reduce the noise we propose to evaluate the temporal dependency of positions estimated from subsequent images. Since subsequent positions of the robot are neighbored and linked using some transition depending on the robot velocity we can use a stochastic weighted averaging approach that is in fact a simplified application of the Kalman filter (see e.g. [5]).

We thereto enclose all estimates with variances that model the degree of uncertainty. We don't use covariances to simplify the modeling. Let denote $(\boldsymbol{r}_t, \psi_t)$

the estimate of the robot's position and heading at time t and $\sigma^2_{r_x,t}$, $\sigma^2_{r_y,t}$, $\sigma^2_{\psi,t}$ the respective variances. After a robot movement the position estimate and variances are updated using the motion model of an omnidirectional robot with velocity $\boldsymbol{v}$ and rotational velocity[1] ω:

$$\hat{\psi}_{t+\tau} = \psi_t + \omega \cdot \tau \tag{5}$$

$$\hat{\boldsymbol{r}}_{t+\tau} = \begin{cases} \boldsymbol{r}_t + \boldsymbol{v} \cdot \tau & \text{if } \omega = 0 \\ \boldsymbol{r}_t + \frac{1}{\omega} R_{\psi_t} \begin{pmatrix} \sin(\omega\tau) & \cos(\omega\tau) - 1 \\ 1 - \cos(\omega\tau) & \sin(\omega\tau) \end{pmatrix} R_{-\psi_t} \boldsymbol{v} & \text{if } \omega \neq 0 \end{cases} \tag{6}$$

with R_ψ denoting the rotation matrix by the angle ψ. The velocity and rotational velocity is measured by odometers. The update of the variances takes into account the inaccuracy of the movement:

$$\sigma^2_{\hat{\psi},t+\tau} = \sigma^2_{\psi,t} + \alpha(\hat{\psi}_{t+\tau} - \psi_t)^2 \tag{7}$$

$$\sigma^2_{\hat{r}_x,t+\tau} = \sigma^2_{r_x,t} + \alpha(\hat{r}_{x,t+\tau} - r_{x,t})^2 \tag{8}$$

$$\sigma^2_{\hat{r}_y,t+\tau} = \sigma^2_{r_y,t} + \alpha(\hat{r}_{y,t+\tau} - r_{y,t})^2 \tag{9}$$

The parameter $\alpha > 0$ controls the assumed accuracy of the movement.

In (8) and (9) we ignore the non-linear dependency between rotational and translational movements of a robot. Ignoring it keeps the statistical modeling efficiently tractable while considering the dependency would require time-consuming statistical techniques like e.g. sequential importance sampling. As long as the frequency of updates remains high, the additional error made by the assumption of independence remains small.

After receiving an image from the camera and calculating the optimal estimate with respect to the image information $(\boldsymbol{p}, \phi)$ we are able to calculate a smoothed position estimate combining $(\boldsymbol{p}, \phi)$ and $(\boldsymbol{r}, \psi)$. Therefore we introduce variances for $(\boldsymbol{p}, \phi)$ that model the uncertainty of the image-based estimator.

The reliability of the image-based estimator is influenced by several aspects: the precision of the optical system, mechanical vibrations, camera calibration errors, the accuracy of image preprocessing and the structure of detected line points. While we can only make crude assumptions about the accuracy of the former aspects we need to model the latter aspect, i.e. the structure of line points, carefully to avoid erroneous estimates.

In section 2.3 we discussed the aperture problem recognizing that the estimate may be reliable in some parameters and unreliable in others. This means, the assumption of uncertainty is different for each of the parameters p_x, p_y and ϕ. We therefore propose to use the curvature criterion to individually determine the variance of each parameter: a small second order partial derivative should be related to a large variance while a large second order partial derivative should be related to a small variance. We use a heuristic function to map second order partial derivatives onto variances. It was determined from a set of experiments by visual inspection.

[1] $\boldsymbol{v}$ and ω refer to the global coordinate system and describe the movement at time t.

The sensor fusion step consists of averaging two independent Gaussian distributions. Denoting with σ_ϕ^2 the variance of ϕ we get:

$$\psi_{t+\tau} = \frac{\sigma_\phi^2 \hat{\psi}_{t+\tau} + \sigma_{\hat{\psi},t+\tau}^2 \phi}{\sigma_\phi^2 + \sigma_{\hat{\psi},t+\tau}^2} \tag{10}$$

$$\sigma_{\psi,t+\tau}^2 = \frac{\sigma_\phi^2 \cdot \sigma_{\hat{\psi},t+\tau}^2}{\sigma_\phi^2 + \sigma_{\hat{\psi},t+\tau}^2} \tag{11}$$

The sensor fusion steps for $\boldsymbol{r}_{t+\tau}$ can be calculated analogously.

Using the filtered estimates helps to improve both, robustness and precision. A single misleading camera image does not lead any more to a loss of track since the filter does not allow to jump to a completely different position which would be possible using the simple algorithm shown in section 2.2.

Additionally, the aperture problem is tackled appropriately: even if the image-based estimate is unreliable with respect to some coordinate axis sensor fusion leads to reliable estimates for all parameters. Moreover, the precision is increased by the sensor fusion due to its implicit smoothing. Hence, erroneous image information do not have a strong impact on the final estimate.

4 Solving the Global Localization Problem

Section 3 discussed the problem of tracking a robot's position starting with a known initial position, i.e. to look for the locally optimal estimate. To solve the global localization problem means to find the global minimum of the error E from (1) even if no initial estimate is available. Certainly, it is not possible to solve the global minimum search under hard real time constraint every cycle.

We therefore propose to apply the tracking approach several times in parallel with random initial positions. The estimates converge very quickly to the next local minima of E. Hence, we can compare the different estimates and choose the best one as our main estimate while the sub-optimal estimates remain under inspection as possible alternatives. By repeated random reinitialization of the alternative estimates we successively scan the whole parameter space.

To avoid premature switches between main estimate and an alternative due to random effects we introduced a discounted scoring system: the best estimate in a cycle gets one point while the others don't get any point. By comparing the discounted sums of points we can evaluate which of the estimates is the overall best one over a longer period of time. Switching the main estimate to an alternative happens only if the score of the alternative becomes greater than the score of the current main estimate.

5 Experiments and Comparison

5.1 Accuracy

All experiments explained here were made on the robots of the *Brainstormers Tribots* team. The algorithms were implemented in C++ under Linux and ran on JVC sub-notebooks with 1GHz Pentium processor. We used a test field which had the dimensions of the fields used in the Robocup 2004 competition.

We made experiments using the joystick to control the robot. In all cases the self-localization approach worked fine. Figure 6 (left) shows a trajectory of self-localization positions that were calculated while the robot drove on a curved trajectory of $6m$ length. The trajectory based on the Kalman filtered positions is very smooth. In contrast, the trajectory build out of the image-based estimates without Kalman filtering exhibits deviations up to $43cm$ and erratic outliers orthogonal to the direction of movement.

In figure 6 (right) we repeated the same experiment with a Particle filtering based self-localization [7]. Both approaches worked on exactly the same input so that they can be compared directly. Obviously, the Particle filter exhibits large deviations from the curved trajectory that are even worse than the deviations of the image-based trajectory in figure 6 (left). These examples show the high precision of the new approach that clearly outperforms the hitherto used Particle filter. In further experiments these results have been confirmed.

5.2 Computational Efficiency

We also measured the computation time and compared it to the Particle filter with 200 and 500 particles. The average computation time is given in table 1. The Particle filter needed four times (with 200 particles) and ten times (with 500 particles) as much computation time as the error minimizing algorithm. In the given framework with hard real-time constraints this saving of time allowed us to increase the number of camera images analyzed per second to 30 which was far not possible with the Particle filtering approach.

Figure 7 shows the cumulative distribution function of the time needed by the error minimizing self-localization algorithm. The computation time linearly depends on the number of line points. It varied between none and 300 per cycle. The maximal computation time was $11ms$ while the average was $4.2ms$. One possibility of restricting the maximal computation time is therefore to restrict

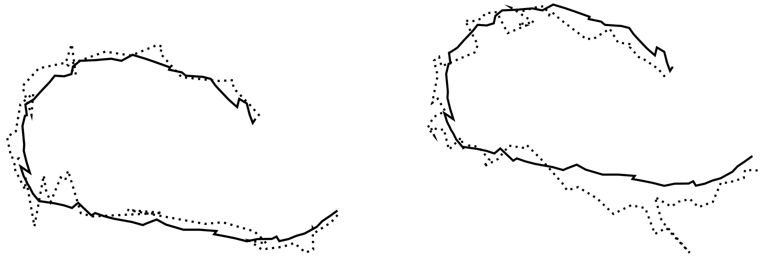


Fig. 6. Left: example of a robot driving with $2\frac{m}{s}$ on a curved trajectory of $6m$ length. The dotted line shows the trajectory of positions which are evaluated optimal considering the camera image. The solid line shows the smoothed trajectory using the Kalman filter. Right: comparison of the error minimizing approach (solid line) with the Particle filter with 500 particles (dotted line) on the same trajectory.

Table 1. Computation time of different methods for self-localization in milliseconds

Approach	Average computation time per cycle
Particle filter with 500 particles	48.3
Particle filter with 200 particles	17.9
Error minimizing self-localization	4.2

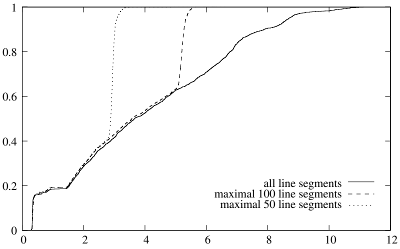


Fig. 7. Cumulative distribution function of the computation time per cycle of self localization in milliseconds (x-axis). The solid line refers to the case of unlimited number of line points while the dashed (dotted) line shows the case of a maximum of 100 (50) line points per cycle.

the number of line points used. E.g. restricting the number of line points to 100 (50) yields a maximal computation time of $6ms$ ($4ms$) without reducing the accuracy of estimates considerably.

5.3 Global Localization

In a third experimental setup we measured the time that was needed to globally localize the robot. Thereto we repeatedly activated a random reset of self-localization and measured the time needed to find the robot's position again.

The global localization approach used one main estimate and three alternative estimates which were repeatedly reinitialized after between $100ms$ and $2000ms$, depending on their quality. Although in every iteration only four positions were evaluated the self-localization found the global optimum in most cases quickly. On average, it took 2.9 seconds. The time needed for global localization heavily depended on the number of line points and their structure: it was far easier to localize in the penalty area in front of a goal with lots of horizontal and vertical field markings than next to the touch line where only a few line points of only a single field marking could been detected.

6 Related Work

There are two different approaches that are closely related to the error minimizing approach: the two-step approach of Cox [1] and the so-called *MATRIX*-approach [11]. The work of Cox uses a range finder to detect walls instead of

Table 2. Comparison of three approaches for robot self-localization

	Cox	MATRIX	Error minimizer
sensory system	range finder/walls	omnidirectional camera/field markers	omnidirectional camera/field markers
principle	2-step	force-field	gradient descend
error function	squared error	≈squared error	M-estimator
deals with outliers	remove in advance	weighted observations	M-estimator
optimizer	analytically/2-step	ad hoc	RPROP
variance estimation	analytically, only for straight lines	none	analyzing the Hessian
sensor integration	fusion of Gaussians	none	fusion of Gaussians
global localization	none	exhaustive search at beginning	randomized parallel search
experiments	$\frac{1}{40}\frac{m}{s}$ robot velocity, frame rate of $\frac{1}{8}Hz$	none	$3\frac{m}{s}$ robot velocity, frame rate of $30Hz$

field markers. Self-localization is based on assigning every observed wall point to the closest true wall and minimizing the squared error. Since this approach does not consider curved walls solving the optimization problem can be done analytically. Experiments are presented only for very slowly moving robots.

The *MATRIX*-approach models the task using an artificial force field which resembles a gradient vector field for the squared error measure. This approach does not consider the aperture problem. Experimental results are missing.

Comparing the error minimizing approach with both alternatives (see tab. 2) shows that the new approach completes its alternatives: in contrast to *MATRIX* it tackles the aperture problem and allows sensor integration while in contrast to Cox' approach it can even deal with noisier sensors like optical systems and with higher robot velocities which cause slippage and imprecise odometer signals.

7 Summary

We proposed a new approach to efficiently solve the robot self-localization problem in the Robocup Midsize league. The approach is based on an efficient numerical approach to find the locally best match between the camera image and the model of the field. Additionally, a stochastic sensor fusion step similar to the Kalman filter is used to link the position estimates calculated from subsequent images and to smooth the trajectory.

This approach enables a low-noise tracking of a robot's position. Experiments comparing the new approach with an existing Particle filtering approach point out the immense noise reduction. Position estimates become more reliable and more precise. Hence, they are much better suited for further calculations like path planning. Thus, using the error minimizing self-localization enables the development of higher level capabilities of robot control and team play.

While an increase in accuracy implicates an increase of computation time using methods like Particle filtering and Hough transform the new approach is very efficient. In experiments we could show that it needs only a tenth of the computation time of a Particle filter. By restricting the number of line points it was possible to restrict the maximal computation time to 4 milliseconds. Hence, the new approach can be used even under hard real time constraints.

Although the basic modeling is not guaranteed to find the overall optimal position we proposed an extension that also solves the global localization problem. We could show by experiments that the robot found its position on average in only 2.9 seconds. Hence, even if the tracking approach is mislead by heavily erroneous sensor information the robot is able to quickly find its position again.

The error minimizing algorithm for self-localization that is presented in this paper is characterized by three properties: high accuracy, robustness and efficiency. It outperforms Particle filtering in all of these aspects. It therefore is a step towards a completely autonomous and robust soccer playing robot.

Acknowledgments. This work was supported by the German Research Foundation (DFG) SPP 1125.

References

1. Ingemar J. Cox. Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.
2. Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, 1999.
3. Arnaud Doucet. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR.310, University of Cambridge, 1998.
4. Hikari Fujii, Yusuke Ohde, Masayuki Kato, Fumitaka Otsuka, Naoka Sema, Marie Kawashima, Eisuke Sugiyama, Shuichi Niizuma, and Kazuo Yosida. EIGEN team description. In *Robocup-2004*, 2004.
5. Arthur Gelb, editor. *Applied Optimal Estimation.* Cambridge, 1974.
6. Luca Iocchi and Daniele Nardi. Self-localization in the robocup environment. In *Proceedings of the 3rd Robocup Workshop*, pages 116–120, 1999.
7. Artur Merke, Stefan Welker, and Martin Riedmiller. Line based robot localization under natural light conditions. In *ECAI 2004 Workshop on Agents in Dynamic and Real Time Environments*, 2004.
8. Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591, 1993.
9. Werner Stahel. Robust alternatives to least squares. Technical Report 78, ETHZ Seminar für Statistik, 1996.
10. Hans Utz, Alexander Neubeck, Gerd Mayer, and Gerhard Kraetzmar. Improving vision-based self-localization. In *Robocup-2001*, 2001.
11. Felix von Hundelshausen, Michael Schreiber, Fabian Wiesel, Achim Liers, and Raúl Rojas. MATRIX: A force field pattern matching method for mobile robots. Technical Report B-09-03, FU Berlin, 2003.

Comparing Sensor Fusion Techniques for Ball Position Estimation

Alexander Ferrein[1], Lutz Hermanns[2], and Gerhard Lakemeyer[1]

[1] Knowledge-Based Systems Group
Computer Science Department
RWTH Aachen
Aachen, Germany
{ferrein, gerhard}@cs.rwth-aachen.de
[2] SMA Technologie AG
Niestetal, Germany
lutz.hermanns@sma.de

Abstract. In robotic soccer a good ball position estimate is essential for successful play. Given the uncertainties in the perception of each individual robot, merging the local perceptions of the robots into a global ball estimate often results in a more reliable estimate and helps to increase team performance. Robots can use the global ball position even if they themselves do not see the ball or they can use it to adjust their own perception faults. In this paper we report on our results of comparing state-of-the-art sensor fusion techniques like Kalman filters or the Monte Carlo approach in RoboCup's Middle-size league. We compare our results to previously published work from other Middle-size league teams and show how the quality of perceiving the ball position is increased.

1 Introduction

Estimating the ball position accurately and reliably is one of the central problems in robotic soccer (RoboCup, [18]), especially in the Middle-size league, where the robots often occlude the ball due to their size. Knowing where the ball is located is central for the cooperation and coordination task of the team. As the rules of the Middle-Size league allow for communication between the robots or to an external host global sensor fusion can be applied for calculating a global ball estimate. Merging the single estimates of the robots into a global ball position is one of the keys to robust team-play as the perception of a single robot might be wrong. With the knowledge of the team-mates the wrong estimate of a single robot can be adjusted. This leads to a better overall team performance.

In this paper we evaluate state-of-the-art sensor fusion techniques for merging the global ball position from the robot's local perceptions, filtering out wrong estimates and false positives. The methods comprise simple approaches based on averaging and more sophisticated ones like the Kalman filter, or the Monte Carlo approach. We tested the global sensor fusion with our team AllemaniACs during the World Cup 2003 in Padua, Italy, and 2004 in Lisbon, Portugal, and the

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 154–165, 2006.

German Open 2004 in Paderborn observing a significant increase in the quality of the ball position estimate. Moreover, we use the global ball information to detect when a robot is dis-localized.

We show that the best methods yield better results than reported before in the literature for the RoboCup setting and that the one reported in [3] is outperformed by an even simpler one. The presented experiments were conducted in real game situations showing a significant increase in the availability and quality of ball position estimates.

The paper is organized as follows. In Section 2 we give a brief overview about the related work on sensor fusion in the robotic soccer domain. In Section 3 we describe our hardware platform and the software system. Section 4 describes the methods we use for merging the local robot perceptions to a global world model and show their results in Section 5. We conclude with Section 6.

2 Related Work

The soccer domain is an interesting domain for research on sensor fusion. Most of the work concentrates on merging perceptions of the ball to one consistent estimate. The methods commonly used are probabilistic method as Kalman filters [9] or Markov Localization [6]. Here, we will concentrate on the related work in the field of fusing ball estimates in the RoboCup domain. One can distinguish between the so-called *local sensor fusion* and *global sensor fusion.* In the former case the perceptions of several sensors on one robot are combined. The latter refers to merging the perceptions of different robots.

Dietl et al. [3] present a ball tracking algorithm, which combines a Kalman filter with Markov localization. They assign a new measurement to an existing track of observation by minimizing the sum of squared error distances. For predicting the ball position a Kalman filter is used. For this Kalman filter they use Markov localization as an observation filter. They report a mean error of 38 cm for a moving ball while the robots did not move.

Stroupe et al. [16] represent each ball estimate as a two-dimensional Gaussian in a canonical form. This allows to merge the single estimates of the robots simply by multiplying them. For predicting the ball position they use a Kalman filter approach.

Pinheiro and Lima [13] also represent sensor information about the ball as a Gaussian applying *Bayesian Sensor Fusion.* They assume that the last position is known and that the single estimates are close by each other.

The team *Mostly Harmless* [14] use local sensor fusion to integrate the perceptions from the different sensors their robots are equipped with. They use a Monte Carlo approach [2] to merge the data from the different sensors into a local world model. They also provide a merged global world model.

The *Milan RoboCup Team* use an *anchoring approach* [1]. The Sensor data are represented symbolically and are anchored with objects from the environment. For the sensor fusion of the symbolical sensor data they use fuzzy logics.

Several other teams participating at the world championships are using local sensor fusion.

3 The Platform

In this section we give a brief overview of the relevant parts of our hard and software system.

3.1 Hardware

The hardware platform is a self-development [20] with the aim of having robots which are competitive in RoboCup and can also be used in office domains for service robotics applications. The platform has a size of 39 cm × 39 cm × 40 cm (Fig. 1). For power supply we have two 12 V lead-gel accumulators with 15 Ah each on-board. The battery power lasts for approximately one hour at full charge. The robot has a differential drive, the motors have a total power of 2.4 kW. This power provides us with a top speed of 3 m/s and 1000 °/s by a total weight of approximately 60 kg.

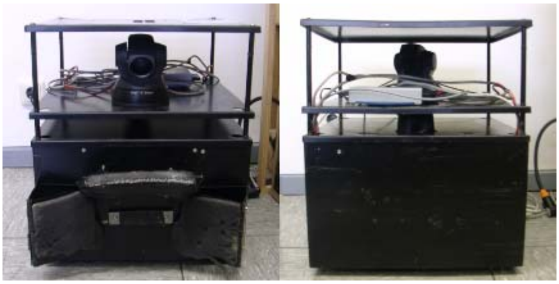


Fig. 1. The hardware platform

On-board we have two Pentium III PC's at 933 MHz running Linux, one equipped with a frame-grabber for a Sony EVI-D100P camera mounted on a pan/tilt unit. Our other sensor is a 360° laser range finder from Sick Ibeo with a Gaussian error distribution and a deviation $\sigma \approx 3$ cm. It runs with a resolution of 1 degree at a frequency of 10 Hz. For communication a WLAN adapter using IEEE 802.11a is installed.

3.2 On-Board Software

For our software architecture we are using a three layered architecture consisting of a low-level layer where sensory inputs and actuator outputs are controlled, a mid-level where modules like collision avoidance and localization are located and a high-level controller for making plans about future courses of actions. Important aspects for the process of merging local ball estimates are the *localization*, the *vision module* which provides the ball estimates, and the *world model* which communicates the local estimates to a global world model where the fusion process takes place.

For self-localization our software provides the *localize* module using the 360 ° laser range finder following a Monte Carlo approach. As for RoboCup it is very important to have continuity in the position updates we track the robot's position with odometry in between two position estimates from the Monte Carlo localization module. The localization uses KLD sampling and cluster-based sampling (e.g. cf. [4,11]). The position estimates provided by the localization have an accuracy of about ±2.5 cm, position losses are very rarely. For more information about the localization we confer to [15].

The ball, goals and flag-posts are localized with the *vision* module using color segmentation and knowledge about the objects' shape. Color segmentation is inexpensive, but the mapping between single objects and their colors or chrominances, if reduced to two dimensions, must be known in advance. For each shape detected object, we compute the chrominance histogram. The histograms are combined based on the Bayes Theorem to obtain a lookup table [7], which is used for color segmentation. To profit from the speed of color segmentation, shape detection is only applied to regions containing the interesting object's color. The circular ball is detected by a randomized hough transform. The goals and flag-posts are modeled as quadrilaterals, which are bordered by straight lines. Object detection takes about 50 ms on a Pentium III with 933 MHz. The entire vision system including the color segmentation and color recalibration step runs at 10 Hz.

The *world model* consists of information like the ball and player position as well as some other internal state information. We provide two kinds of world model, a local one which is constructed from the own perception and a global one which is the result of a world model fusion on an external computer.

For inter-process communication we use a blackboard communicating via shared memory between the two on-board computers and via UDP between the robots.

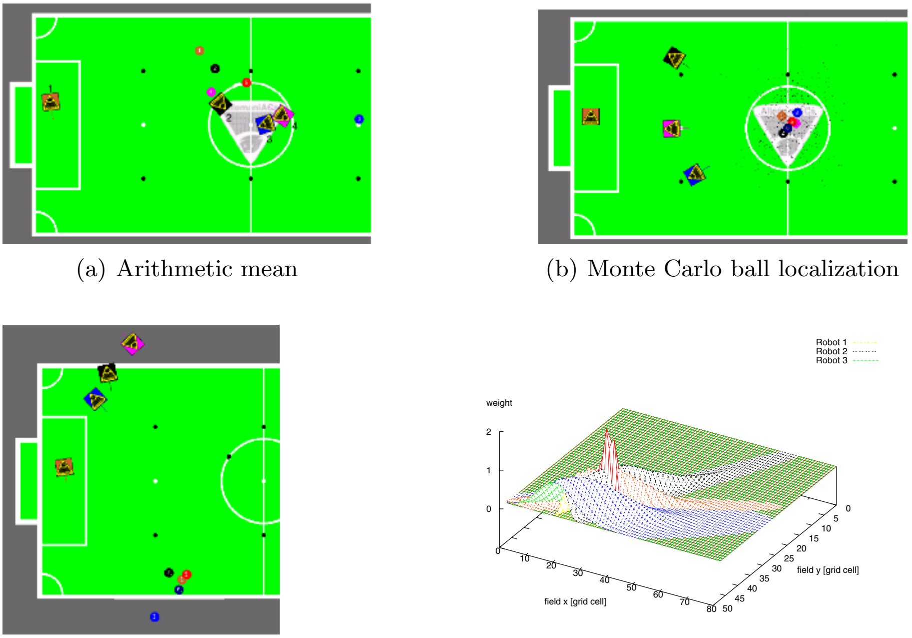
4 Sensor Fusion Techniques

In this section we give a brief overview of the sensor fusion techniques we use in this case study. We start with simple ones like mean methods and go over to more sophisticated ones like the Weight grid method, the Kalman filter, and the Monte Carlo method. Finally, we test a combination between the Weight grid and the Kalman filter. This method is similar to the one proposed in [3].

4.1 Arithmetic Mean

Each robot i seeing the ball contributes his local estimate $(lb_x, lb_y)^{(i)}$ about the ball position by communicating it to a central server. The global ball estimate (gb_x, gb_y) is calculated by averaging over the estimates from each robot, i.e. $gb_x = \frac{1}{n}\sum_{i=0}^{n} lb_x^{(i)}$, $gb_y = \frac{1}{n}\sum_{i=0}^{n} lb_y^{(i)}$.

Here, every local ball estimate has the same importance. The estimate of a robot which is far away from the ball should not be weighted as much as the estimate of a robot being close to the ball. Therefore, in a variant we weight the



(a) Arithmetic mean

(b) Monte Carlo ball localization

(c) Example situation for the weight grid fusion

(d) The weight grid for the situation in Fig. 2(c)

Fig. 2. Fusion techniques

estimates according to the distance from the robot to the ball and a time factor which denotes how long ago the robot has seen the ball for the last time:

$$w_i = 1/dist_i \cdot conf_b{}^{(i)} \cdot conf_p{}^{(i)}. \qquad (1)$$

$conf_b$ is the ball confidence provided by the vision system. The role of this confidence is to give some means of persistence to the ball estimate. If the ball is not seen in one frame it is not reasonable to assume that the ball disappeared from the previously detected position as the vision system has some detection error. The confidence is modeled by a rapidly decreasing function over the time, which is set to 1 if the ball is detected. This confidence helps stabilizing the local ball estimates. The other confidence $conf_p$ is provided by the localization system and represents the confidence that the robot is located at the given position. The weighted mean is then calculated as $gb_x = \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} w_i \cdot lb_x{}^{(i)}$ and $gb_y = \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} w_i \cdot lb_y{}^{(i)}$. An example of the weighted mean estimation is depicted in Fig. 2(a). The local ball estimates are enumerated and marked with the respective robot's color. The merged estimate is numbered as estimate 5 (red ball).

4.2 Weight Grid

This method uses a grid to represent positions on the field. The representation is similar to an occupancy grid representation [12], but each cell can take weights greater than 1. For each ball estimate a 2-dimensional Gaussian distribution is calculated. The parameters for this Gaussian are taken from the *distance error* and the *pan error* which are provided by the vision system. Fig. 3 shows the parameters.

The cell update works as follows: each local ball estimate is weighted by the weight function given in the previous section (Eq. 1). These weights are then multiplied with the Gaussian distribution and the result is stored in the respective grid cells. Fig. 2(c) shows an example situation where the ball can be seen by the goal keeper, the blue, and the black robot. The perception of the goal keeper (brown) is closest to the real ball position (dark blue) and its distribution is due to the weighting more narrow than ones from the other robots. The merged ball position is depicted in red.

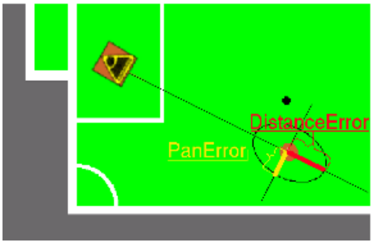


Fig. 3. Ball confidence

4.3 Kalman

A Kalman filter [9] is used to estimate the state of a process variable $x \in \mathbb{R}^n$ in a dynamic system. As we want to estimate the position of the ball we have $x = (gb_x, gb_y)^T$. The basic equation describing the stochastic process is $x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$ with measurements $z_k = Hx_k + v_k$, where w_k and v_k represent the process and measurement noise, resp. They are assumed to be independent of each other and normally distributed, i.e. $p(w) \sim N(0, Q)$ and $p(v) \sim N(0, P)$.

The matrix A represents the motion model relating the old process state with the new one. In our case we do not integrate a motion model directly as the local ball measurements from all robots are from the same point in time. Instead we propagate the global ball position by the ball velocity which is calculated using the last global ball estimates each time the Kalman filter is called. This simplifies the application of the motion model. Otherwise a complex motion model has to be found and applied. It turned out that applying the motion model this way works fine for the ball tracking task. To integrate control inputs in the estimate the variable $B \cdot u$ is added to x_k. In our case $u = 0$. Therefore, the state equation results in $\begin{pmatrix} gb_{x,k} \\ gb_{y,k} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} gb_{x,k-1} \\ gb_{y,k-1} \end{pmatrix} + w_{k-1}$.

To integrate the sensor measurements z_k the matrix H is used denoting the observable components of x. In our case it is also the identity matrix as we can observe both coordinates of the ball. The process noise covariance matrix Q was empirically found as $Q = diag(3, 3)^T$, the measurement noise covariance matrix P is initially set to $P = diag(4, 4)^T$. For the first time when the filter is started we take the very first measurement from one robot as initial value for

x_0. Applying the time update and measurements update equations (cf. eg. [10]) we estimate the global ball position from the robots' local estimates.

As another variant we use the *Kalman reset* filter. Here, the matrix P is reset to its initial value each time a global ball position is calculated, which in our case happens 10 times a second. This means that the actual measurement receives more attention.

4.4 Monte Carlo Localization

Monte Carlo Localization was introduced in [5] as an improvement of the Markov Localization [6]. Both techniques originally were developed for the self-localization of a robot.

The main idea of *Markov Localization* is to provide a probability distribution $Bel(l)$ over the space of possible positions. New sensor readings are used to re-calculate the distribution with the help of Bayes' rule. The distribution is represented in an occupancy grid.

The *Monte Carlo (Ball) Localization* represents the distribution $Bel(l)$ with a set of weighted samples instead of the grid which is used in the Markov localization. For the ball fusion we take samples that represent the ball position hypotheses. Therefore, a *sample* x_i consists of a possible ball position $l = \langle x, y \rangle$ and a weight w_i which is also called *importance factor*: $x_i = \langle \langle x, y \rangle, w_i \rangle$.

In Fig. 4 the Monte Carlo Algorithm is shown based on [19] using a set of samples X the ball velocity vel and a set of local ball estimates LB. In one iteration of the algorithm m samples from the sample set are drawn by chance according to their importance factor. A sample with a high weight is drawn more often than one with a lower weight. Additionally, some new samples are generated around new local ball estimates. Then, the algorithm works in two steps. First the motion model is applied to the drawn samples. This is done by promoting the samples according to the ball velocity and some noise. In the second step the samples are re-weighted with the new local ball estimates using the measurement error covariance matrix around the estimated ball position.

In practice it turned out that Monte Carlo is applicable with a set size of 1000 samples in our time constraints. With more than 1500 samples we are not able to fulfill our time constraints of 100 ms of computation time any more. In

```
AlgorithmMCL($X, vel, LB$):
  $X' = \emptyset$
  for $i = 0$ to $m$ do
      generate random $x$ from $X$ according to $w_1, ..., w_m$
      generate random $x' \sim p(x'|vel, x)$
      $w' = p(LB|x')$
      add$\langle x', w' \rangle$ to $X'$
  endfor
  normalize the importance factors $w'$ in $X'$
  return $X'$
```

Fig. 4. MCL

Fig. 2(b) the samples of the distribution approximation are depicted. The dark blue ball resembles the true position of the ball, the red one is the fusion result. The black dots show the samples.

4.5 Combined Fusion

Inspired by the work of the CS Freiburg Team [3] we combined the weight grid technique (Sect. 4.2) with the Kalman filter (Sect. 4.3). All local ball estimates are used to calculate a *reference ball* by using our weight grid algorithm. Only those local balls are used as input for the Kalman filter that are in between a radius of 1 meter to the reference ball. If the resulting ball of the Kalman filter is too far away from the calculated reference ball the Kalman filter receives a reset.

5 Experiments and Results

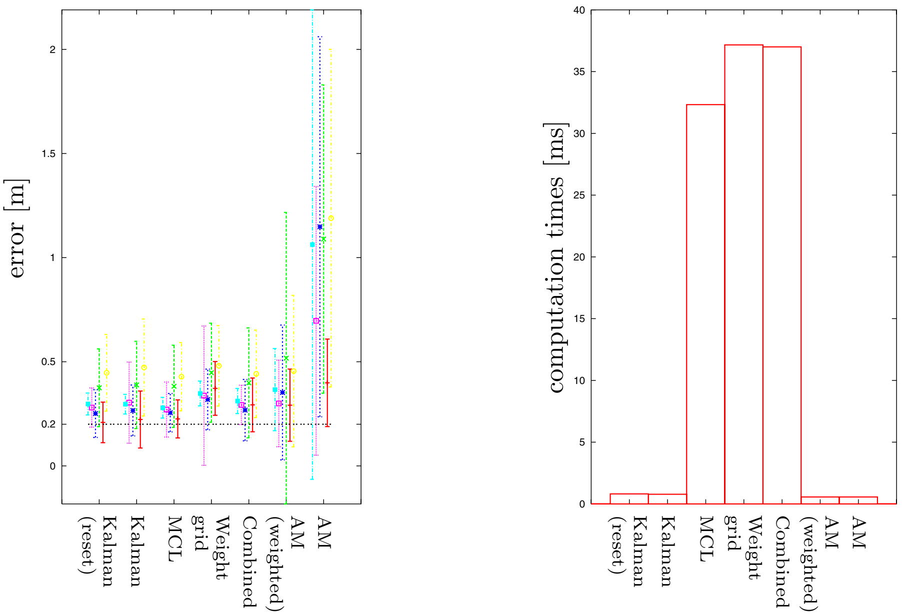
To get significant and realistic results about the quality of the presented fusion techniques we tested them in real game situations. We made several test games at the Philips RoboCup team in Eindhoven. To get ground truth for the real ball position we used a ceiling camera. All relevant data were logged and can be replayed. The ground truth data were manually added to the log data. As this is rather elaborate, we do not have any ground truth data from the German Open or the RoboCup championships.

In the game against Philips, we picked 6 different sequences lasting between one and three minutes with significant action on the field. For example, in one sequence the ball was blocked between two robots so that the other robots of the team did not see the ball at all. One sequence is a typical game start situation, in another one the Philips robot kicked very often (and very hard, as usual) resulting in a rapidly moving ball. During some of the sequences one or more robots were dis-localized reporting wrong ball estimates.

The evaluation results are shown in Fig. 5(a). The data represent the mean error in meters together with its deviations over each test run[1]. Not very surprisingly, the arithmetic-mean method yields the worst results, followed by the weighted arithmetic mean. The reason is that one outlier is enough to drag the merged position into the wrong direction. Even with weighting the estimates according to their distance to the ball this bias cannot be prevented. The grid-based method (combined and weight grid) are in the midfield wrt. to their accuracy.

The first three places are taken by the Kalman, Kalman with reset, and Monte Carlo. Depending on the game sequence one of the three scored first place. One can see that in the mean these method have an error of about 20 cm. This is better than the results reported on in [3]. They reported on a position error of about 38 cm. Moreover, they conducted their experiments in a static setting, where the robots did not move during the experiments.

[1] For the whole testing time we had 8.520 cycles where the fusion module calculated a global ball estimate.



(a) The mean error for the different methods

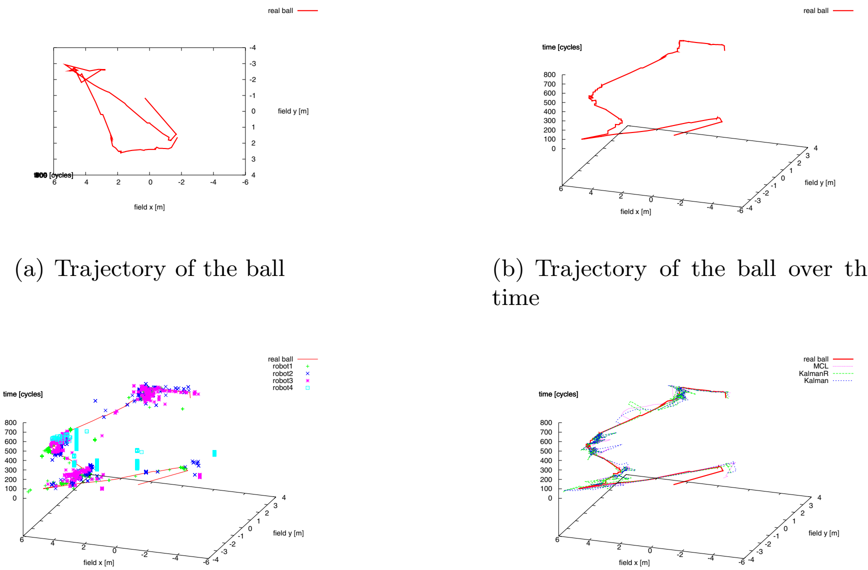
(b) Computation times in ms

Fig. 5. Comparison of the methods

With respect to the computation times one can state that the Kalman filter methods clearly outperform the other methods. With an average computation time of 0.8 ms and an error of about 20 cm the Kalman filter methods are accurate and fast. What surprised us most was that combining a Kalman filter with a weighted grid which is similar to the method proposed in [3] does not seem to pay off, as a simple Kalman performed better, both in accuracy and computation time. To give an impression about the tracking of these methods we show one sample trajectory in Fig. 6(a). Fig. 6(b) shows the same trajectory over the time from a reverse angle. The tracking results of the two Kalman methods and the Monte Carlo method are depicted in Fig. 6(d). To get a feeling about the quality of the tracking methods we moreover depicted the robots' local ball estimates in Fig 6(c).

In the next experiment we tested another typical RoboCup scenario: it happens very often in RoboCup that the referee picks up the ball, for instance after the ball passed the side line or was stuck between robots. Then, no robot can see the ball. The referee places the ball at some restart spot again. In this situation it is important to get stable ball estimates as soon as possible. In our second experiment, the robots took their kick-off positions and two helpers staying at the opposing restart points randomly placed a ball at one of these points.

In this experiment, again the Kalman reset method showed the best results. The Kalman reset had a mean error of 0.45 m, which is reasonable with respect



(a) Trajectory of the ball

(b) Trajectory of the ball over the time

(c) Sensor Data

(d) Tracking results of the Kalman and Monte Carlo methods

Fig. 6. Sample run

to a minimum distance of 5 meters between the robots and the ball. Moreover, the computation time of 0.6 ms is a good result, especially compared to Monte Carlo which takes in the order of two magnitudes longer.

6 Conclusion

In the RoboCup domain it is very important to have a good estimate about the ball position. As the ball is perceived by only some robots most of the time a stable global ball estimates helps to increase the team performance. The result for RoboCup's Middle-size league is that the Kalman reset filter shows the best performance.

The quality gain using a ball fusion technique is significant. The availability of a global ball estimate lies at over 80 % in our experiments while each robot itself has seen the ball only 50 % of the time.

Another nice effect of using a ball fusion technique is that one is able to detect when a robot is dis-localized. Comparing local and global estimates it is easy to notify when those values differ too much in order to detect a wrong localization position. Of course, it is crucial to find that value as false positives must be avoided. It turned out that using the Kalman reset method we detected all dis-localizations during the test runs having only one false positive. More details about that can be found in [8].

Another problem is to gather ground truth data for empirical evaluation. In our experiments we installed a ceiling camera and associated the real ball position with the logged data by hand. This is a very time consuming process. Recently, Stulp et al. [17] proposed a ceiling camera system to acquire ground truth data for the robots and the ball. With these data, we could evaluate the methods on a wider data basis. For the future we would like to conduct further experiments with real tournament data.

As the presented results are very specific to the RoboCup domain in general it turns out that one should try out several methods for the specific application domain in order to find the most appropriate method. Further, one can observe that applications of the Bayes filter (Kalman, Monte Carlo) provided the best estimates for the fusion task.

Acknowledgment

This work was supported by the German National Science Foundation (DFG) in the Priority Program 1125, *Cooperating Teams of Mobile Robots in Dynamic Environments* and by a grant from the NRW Ministry of Education and Research (MSWF).

References

1. A. Bonarini, M. Matteucci, and M. Restelli. Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem? In *Proc. AAAI Fall Symposium on Achoring Symbols to Sensor Data in Sungle and Multiple Robot Systems*, 2001.
2. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, May 1999.
3. M. Dietl, J.-S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS-2001)*, 2001.
4. D. Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.
5. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, pages 343–349, 1999.
6. D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11, 1999.
7. C. Gönner, M. Rous, and K.-F. Kraiss. Robust color based picture segmentation for mobile robots. In *Autonome Mobile Systeme*, pages 64–74, Karlsruhe, Germany, Dec. 2003. Springer. (in German).
8. L. Hermanns. Fusing uncertain world information of cooperating robots into a global world model. Diploma thesis, Knowledge-based Systems Group, Computer Science V, RWTH Aachen, Aachen, Germany, (in German), 2004.
9. R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, pages 35–45, Mar. 1960.

10. P. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic Press, 1979.
11. A. Milstein, J. N. Sànchez, and E. Williamson. Robust Global Localization Using Clustered Particle Filtering. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 581–586, 2002.
12. H. Moravec and A. Elfes. High resolution maps from wide angular sensors. In *Proc. of the IEEE International Conference On Robotics and Automation (ICRA)*, pages 116–121, 1985.
13. P. Pinheiro and P. Lima. Bayesian sensor fusion for cooperative object localization and world modeling. In *Proc. 8th Conference on Intelligent Autonomous Systems*, 2004.
14. G. Steinbauer, M. Faschinger, G. Fraser, A. Mühlenfeld, S. Richter, G. Wöber, and J. Wolf. Mostly harmless team description. In *Proc. RoboCup Symposium*, 2003.
15. A. Strack, A. Ferrein, and G. Lakemeyer. Laser-based localization with sparse landmarks. to appear in Proc. RoboCup Symposium 2005, 2005.
16. A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proc. of 2001 IEEE Int. Conf on Robotics & Automation (ICRA-01)*, 2001.
17. F. Stulp, S. Gedikli, and M. Beetz. Evaluating multi-agent robotic systems using ground truth. In *In Proceedings of the Workshop on Methods and Technology for Empirical Evaluation of Multi-agent Systems and Multi-robot Teams (MTEE)*, 2004.
18. The RoboCup Federation. http://www.robcup.org, 2004.
19. S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.
20. J. Wunderlich. Technical description of the allemaniacs soccer robots. Technical report, LTI / KBSG, Aachen University, 2002.

RoboCup X: A Proposal for a New League Where RoboCup Goes Real World

Tijn van der Zant and Thomas Wisspeintner

Fraunhofer Institute for Autonomous Intelligent Systems,
Schloss Birlinghoven,
53754 St. Augustin, Germany
tijn@aisland.org,
thomas.wisspeintner@ais.fraunhofer.de

Abstract. To put more emphasis on real-world problems, the authors propose to extend the RoboCup competitions. In order to foster progress in the desired abilities the authors propose to expand the existing challenges by a set of simple tests. The passing of the entire set should lead to robots that are capable of working both autonomously and in cooperation with humans in different realistic scenarios. Robots from all RoboCup leagues but also from outside of RoboCup should be allowed to participate. The new league especially aims at fostering the development of practical solutions and applications for supporting humans in everyday life.

1 Introduction

1.1 The Aim of RoboCup

On the first page of the official RoboCup web-site [1] it is stated literally that:

RoboCup is an international joint project to promote AI, robotics, and related field. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined.

Furthermore it says:

The ultimate goal of the RoboCup project is, ***By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer.***

It ends with:

One of the major application of RoboCup technologies is a search and rescue in large scale disaster. RoboCup initiated RoboCupRescue project to specifically promote research in socially significant issues.

1.2 Achieving the Aim

To achieve the RoboCup mission statement the closed environments of present day leagues are not enough. The social interaction between humans and robots is, due to the 'no human intervention' rules in most leagues, almost non existing.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 166–172, 2006.

But soccer playing and rescue also include social aspects. For example, robots have to cooperate with humans in rescuing efforts. The authors think that after almost a decade of RoboCup it is about time to take the first steps out of the artificial environment. The rescue league is making a good progress in this direction but the disaster scenario in the Rescue league is for sure not a usual but an exceptional case humans encounter. There is a need for application based progress, for robots cooperating with humans. The RoboCup X-games are more AI based than the other leagues, since social intelligence and interaction with both humans and robots will play a big role.

2 RoboCup eXtends

The RoboCup X-games are a proposal to advance robotic technologies beyond what nowadays is or seems possible. It is also meant to inspire people from different leagues to use and recombine existing technology and work together on new solutions and useful applications. The X in the name of the league stands for eXtended or eXpanded, eXperimental, eXtreme and for the celebration of the $X^{th}/10^{th}$ anniversary of RoboCup. It is not about a single specific topic, but about the building of re-usable real-life robotic solutions. It is extended because it widens the scope of RoboCup beyond rescue and soccer, experimental because new technologies can be applied and extreme because it puts robots in uncertain real-life situations where they have not been before.

To retain the necessary attention and acceptance of society and economy to continue high quality research, we have to prove that the results of our work are useful and promising. Coping with self-explaining real-life tasks seems a good way to do this. The RoboCup initiative has the quality needed, but not all elements are there yet.

2.1 Competition

RoboCup is a versatile set of challenges where the robots compete against each other. The element of competition accelerates the process of research and offers the possibility of benchmarking. A drawback of the competition element is that in some leagues the approaches used in hardware and software seem to converge to a specific set of solutions, pushed by the rules of the competition. In the RoboCup X-games, the real world robot league, this should be avoided, while retaining the competitive elements. The proposal is to have a group of tests that benchmark a robot on its ability to fulfill useful tasks in applied, real-life scenarios with a lot of human intervention. As a consequence, effective human-machine interaction will play a big role in this league.

The tests in this league are rated with points. In the end, the one with the highest amount of points wins. By choosing new tests the community is able to rapidly steer the research in desired directions. The idea to propose a *new* league instead of adjusting an existing one is that it is probably tough to change a league so dramatically that it includes social human interaction. Many researchers have put a lot of effort in one of the existing leagues and would like to continue with it.

3 The RoboCup X Tests

RoboCup X is a benchmark for RoboCup and non-RoboCup robots. If a robot wins the RoboCup X-games this does not mean that it also wins in another competition, but it does suggest that its application-related qualities are better than of others. By actually using the benchmark tests of RoboCup X the RoboCup community is able to set the standards for research groups and the industry.

The areas that are covered by RoboCup X should include, but are not limited to: animal like/intelligent behavior (bio-inspired robotics), adaptivity, appearance, applicability, autonomy, endurance, ergonomy, human-robot interaction, modularity, navigation, out-door abilities, precision work, reliability, reusability, safety, speed and transportation.

The criteria of the tests are the following:

- The scenarios/tests are easy to set up, so that researchers can test at home. They are low cost and available world-wide.
- Tests should preferably not be boring to watch.
- Tests take a finite and not too long amount of time.
- The tests should have definite and easy rules, to avoid long-lasting rule-specific discussions.
- The test should be self-explaining
- The tests should preferably point at applications which are possible with the new abilities.
- The tests should have a strong relation to real-life problems and applications, partly driven by industrial demands.

3.1 The RoboCup X-Games

We expect many of the tests in the RoboCup X-games to have a good show element. Competing and socially interacting robots make a versatile show for both scientists and the public watching. It also attracts the industry by focusing on real-world applications for every day use. The stakes are a bit higher in the RoboCup X contests because real world scenarios are less predictable. Some test might even involve interaction with the public!

The size, weight and other dimensions of the robots are specified in the rules of the other leagues. Any robot that is allowed in another league is allowed in the RoboCup X-games, but also modification or rebuilding a robot within a certain range of size should be possible.

The authors think that the technical committee of the RoboCup X-games should allow *all* technologies, unless it requires a *special external setup* or is harmful to other robots or humans. The RoboCup X-games should encourage innovative ideas on autonomous systems, even if they are not allowed in the other leagues. This rule should prevent external camera use, special navigational markings, radio waves for local positioning etc. Though it should allow GPS, sonar, laser range finders, vocal communication etc.

3.2 Points

Autonomy: The robots have to work autonomously. The problem is to define autonomously. The authors suggest that a restricted period of time (e.g. 10 minutes) is given, where human-intervention is allowed before the test (e.g. for calibration and set-up) on the test-site. There should be *no* flexibility in these time limits once they are defined. Exceeding this time-limit, or controlling the robot during the test remotely by a human operator is considered a failure. If one cannot comply with the specifications the team faces disqualification for the specific test.

Difficulty multiplier: If less than $\frac{1}{4}$ of the participating teams succeeds in a test it gets a multiplier of 2. If more than $\frac{3}{4}$ of the participants succeed then it gets a multiplier of $\frac{1}{2}$. The last situation indicates that the test could be revised for the next games, either by changing it or not having it.

General applicability: A test consists of 2 parts. First a test is done with settings specified by the robots team to show that the system works. After a successful demonstration of the capabilities the same test is done with generalized and therefore more difficult settings specified by the RoboCup X committee. If the robot also succeeds here the amount of points is doubled.

Human intervention: During a test human intervention (like touching, restarting or helping a robot out of a stuck situation) is not allowed and results in disqualification for the test, unless human interaction is an essential and wanted feature of the test itself (like giving speech or gesture commands).

3.3 Specific Tests

The general idea: A list of possible tests with a short description is presented here. The list is a proposal and debatable. Of course for all tests it is obligatory that the robot remains (fully) operational during the entire test. A maximum of 3 robots per team is allowed to participate. Only one of these robots is allowed to participate in a single test. Calibration periods are very restricted, just as repair time. The idea is that the robot goes through as many tests as possible in a very short period of time. In the first years a tame version of the rules and the tests can be used.

The list presented here are proposals for tests. Further in this article a few tests are highlighted to be used in the first edition of the RoboCup X-games and described in more detail. On request and after consulting the RoboCup community the committee of the RoboCup X-games league can add or change a test.

Bar. Can the robot serve drinks in a bar?

Burglar. Does it recognize its owner and give an alarm when it detects an unknown intruder?

Elderly. Can it aid elderly people, for example support a person while walking?

Falling. Does the robot survive a 10 cm drop? And 25 cm? And 50 cm ? And 100 cm?

Guide robot for the blind. Lead a blind person from place A to B.

Light. The robot should, without manual recalibration, go to an object that it has seen in a dark place, but which is now put outside or the other way around.

Looks. Does it look nice to interact with or does it look very technical with sharp edges and wires sticking out?

Open challenge. Demonstrate a new and challenging ability not yet covered in another test.

Racing. Can the robot go on a (difficult) race track? The early version can consist of a race track with an uneven floor. Touching the border reduces the score. Putting more than one robot on the track gives a very nice racing character to this test.

Rain. The robot should be capable of functioning in rainy environments. A shower or watering can should suffice.

Robot agility. Similar to competitions with dogs, walk, talk or point a robot through an obstacle parcours.

Stairs. Can it climb stairs autonomously? And a stairway?

Supermarket. Can it find a certain product in the supermarket (on a shelf)?

Suitcase. Carry a suitcase and avoid obstacles while following a human. Sound an alarm if the suitcase gets stolen or lost.

Traffic. The robots drive in a miniaturized street where the robots have to follow a certain path (e.g. marked by arrows on the ground) the path consists of one mayor crossing and some streets on the side. The robots have to obey the traffic rules, recognize signs, etc. The robot which does the most rounds in a certain time, without bumping in other robots or going besides the track, wins.

3.4 Tests for the First Time

Out of the list presented in the previous paragraph we have chosen the following for consideration, due to the ease of set-up and possible success of the robots.

Terminology for the remainder of the article: The terminology in the rest of the article is as follows: 'Ranking' means three points for the first place, two for the second and one for the other succeeding robots. 'Boolean' means that every robot that succeeds gets one point, no success is no points.

Elderly. Can it aid elderly people, for example support a person while walking? The elderly person must be able to steer the robot, while leaning on it, by simple commands such as voice commands. The test is boolean.

Guide robot for the blind. Lead a blind person from place A to B in the RoboCup area. The test is boolean.

Humans. Can the robot recognize different humans (bodies and/or faces) and express this in some way, preferably with sound (speech). To pass the robot has to recognize five different humans and has to be trained on the spot in unknown light conditions. The test is boolean.

Light. An object (unknown in advance) is shown inside the building on a random spot. The robot is taken outside and may not be re-calibrated. The object is put within within a 15 meter radius of the robot three times. If the robot touches the object all three times then the robot passes the test. The test is boolean.

Looks. The appearance of the robot is rated by a jury. Among the criteria can be: smoothness of movements, integration of components (no wires), easiness of charging, attractiveness of shape, safeness of shape, human-machine interaction, type of applications,... The test is ranking based.

Open challenge. Demonstrate a new and challenging ability not yet covered in another test. A jury decides on the ranking. It can act as a test generator for the years to come.

Rain. The robot has to keep driving around (slowly) for one minute avoiding black objects while cold water out of a shower or watering can is sprayed on top of it. The test is boolean.

Race. A circular race track with white borders is used to race on. Several robots race at the same time. Bumping can lead to disqualification. The race lasts 10 minutes and the robot that has traversed the biggest distance wins. The test is ranking based.

Stairs. The robot has to climb stairs somewhere in the RoboCup building. The test is boolean.

Suitcase. Carry a suitcase and avoid obstacles while following a human through the RoboCup area. Sound an alarm if the suitcase gets stolen or lost. A bit more difficult would be to take a walk from the inside to the outside of the building and back. This is a boolean test.

3.5 The Final Scenario

At the end of the test round the teams are ranked according to their points they have scored. The teams with the highest points go to the finale which consists of a scenario with an open challenge. An example could be a living room with a person watching television. Here the best robots have ten minutes (five for the setting up, five for the application) to show some capabilities. The ranking of this final show, combined with the ranking from the tests (and maybe even from the audience!), determines the final ranking. The jury should not only consists of roboticists, but also others such as journalists, famous persons from television, government officials, people from the industry, ...

The idea behind this is to promote the open challenge and stimulate the creativity of researchers. The entire finale lasts about an hour and is a nice show for spectators, journalists and participants. A professional host who can entertain the audience could present the entire show, have some interviews, talk to the jury etc. This should preferably be arranged professionally (maybe even by a network station) so it can be broad casted as a nice robot show.

In a previous email discussion (initiated by one of the authors) on the mid-size league list a few years ago, there were roughly two opinions on this issue. One opinion was that we are doing science and should not bother about this, also because it might degrade the work. The other opinion was that such a show does not diminish the importance of the scientific work, but focuses on the fun side for the layman audience watching. The professional will recognize the scientific qualities and/or can read about the work in articles. We think it is worth a try.

4 Discussion

We all know that the RoboCup has to go real world, in order to remain interesting. Playing soccer and winning from humans will probably not bring us all the problems we would like solve in order to have robots capable of functioning autonomously in the real world, which consists of human societies and rough nature.

RoboCup X provides a natural way of extending the current competitions. The tests are not all or nothing but build up in complexity. Also a team does not have to bring several robots to the competition so the overall costs for participation can remain relatively low.

Besides the test cases, which are interesting by themselves, RoboCup X also provides a show for the layman, who is not interested usually in the technical details of the system. By covering a broad area (from science and technology to entertainment and fun) we should be able to get more attention from the media, which increases the chance of more funding by governments and increased interest from the industry. Hopefully the new league can bring some of the excitement that participants feel over to the layman.

In 2006 in Germany the tenth RoboCup world championships will be held. To celebrate this, we should show that we, as a community, are dedicated to improve the lives of human beings; that we work toward a bigger goal that transcends economic pressure and short term visions. With RoboCup X we can show that the community is working toward applicable robots for general use. Hopefully the community can demonstrate, as a gift for the 10^{th} anniversary of RoboCup and to honor the people who have so dedicatedly devoted their careers to this magnificent world-wide event, that we are truly caring for society at large.

References

1. http://www.robocup.org
2. http://www.Xprogramming.com

SimRobot – A General Physical Robot Simulator and Its Application in RoboCup*

Tim Laue, Kai Spiess, and Thomas Röfer

Bremer Institut für Sichere Systeme, Technologie-Zentrum Informatik, FB 3,
Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
{timlaue, kspiess, roefer}@tzi.de

Abstract. This paper describes SimRobot, a robot simulator which is able to simulate arbitrary user-defined robots in three-dimensional space. It includes a physical model which is based on rigid body dynamics. To allow an extensive flexibility in building accurate models, a variety of different generic bodies, sensors and actuators has been implemented. Furthermore, the simulator follows an user-oriented approach by including several mechanisms for visualization, direct actuator manipulation, and interaction with the simulated world. To demonstrate the general approach, this paper presents multiple examples of different robots which have been simulated so far.

1 Introduction

When working with robots, the usage of a simulation is often of significant importance. On the one hand, it enables the evaluation of different alternatives during the design phase of robot systems and may therefore lead to better decisions and cost savings. On the other hand, it supports the process of software development by providing an replacement for robots that are currently not on-hand (e. g. broken or used by another person) or not able to endure long running experiments (e. g. learning tasks [1]). Furthermore, the execution of robot programs inside a simulator offers the possibility of directly debugging and testing them. This is a great benefit when working with platforms that do not offer any direct debugging facilities, e. g. the Sony AIBO robot.

In the past, several robot simulators have been developed with different main focus on complexity, accuracy, and flexibility. There are also differences in the possibility of creating and integrating own robot models and virtual environments. Some of the simulators are restricted to a two dimensional environment or are only approximating dynamics and realistic interaction of the robots with the environment. In the following, we give a short overview of current related works on robot simulators with accurate dynamics simulation for three dimensional environments and with relevance to the RoboCup domain. These will be compared with SimRobot.

* The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 173–183, 2006.

UCHILSIM [1] is a robot simulator developed by the University of Chile. It is specially designed for the RoboCup Four-legged League. The simulator contains dynamics simulation using the Open Dynamics Engine (ODE) [2], a graphics engine, and has a window based graphical interface. The environment and the robots are described in a VRML structure extended by nodes for simulator elements and physical attributes. It has interfaces to their UChile1 software package and their learning component. At the current stage, this simulator is rather specific for one RoboCup league.

Another 3-D simulator used in the RoboCup domain is Übersim [3], which has a focus on vision-centric robots in dynamic environments. It has a client/server based architecture, where clients communicate with the server over TCP/IP. The simulator also uses ODE for dynamics simulation. Own robots can be modeled by programming their structure in C classes. In the current release [4], only two sensors are predefined: a camera sensor and an inclinometer. At the moment there seems to be no graphical user interface for interacting directly with the robots or the environment during simulation time.

A more general 3-D multi-robot simulator with graphical interface and dynamics simulation is Gazebo [5], a part of the Player/Stage project [6]. This simulator has a large variety of sensors and comes with models of existing robots such as the Pioneer2DX or the SegwayRMP. The robots and sensors can be controlled by the Player server or controllers can be written using a library provided with the simulator. The simulated environment is described in XML files using several predefined elements and robots. It also offers the possibility of creating and integrating own robots as plug-ins but this has to be done by code-based modeling in C.

Webots [7, 8] is a commercial robotic simulator developed by the Cyberbotics Ltd. It has an ODE-based physics simulation and provides several robot models such as Sony Aibo, Khepera, or Pioneer2. The robots and the environment are described using the VRML standard for graphical models, extended by nodes for the Webots elements, sensors, and physical attributes. Controllers can be programmed in C++ or Java and connected to third party software through a TCP/IP interface.

In comparison with these simulators, the following features of SimRobot may be pointed out and will be described in this paper: The simulator is not limited to any special class of mobile robots[1]. By using an XML-based modeling language, users are enabled to specify robots and their environments completely without any additional use of other programming languages. A large set of body elements, actuators and generic sensors allows the free composition of arbitrary robot models. An important element of simulations, which is ignored in many cases, is the support of the work of the user. This is addressed by SimRobot by providing several possibilities of visualization and interaction with the simulated world.

To simulate rigid body dynamics, SimRobot also uses ODE, since this engine has a wide variety of features and has been used successfully in many other

[1] Admittedly, SimRobot will not support the special domains of underwater robotics and aircrafts.

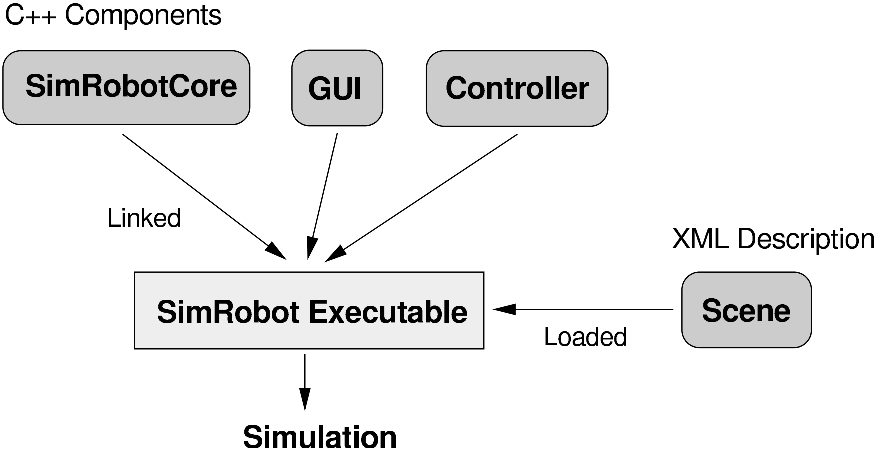


Fig. 1. The modules of SimRobot and their dependencies

projects. The visualization as well as the computation of imaging sensor data is based on OpenGL, because this industry standard offers the best performance on modern hardware on different platforms.

Previous works on this simulator project have been a kinematic robot simulation [9] and a preliminary release (without dynamics) for the GermanTeam in the Sony Four-legged Robot League [10].

This paper is organized as follows: Section 2 describes the general architecture of SimRobot, Section 3 shows several special features of the user interface, the Sections 4 and 5 describe the elements which SimRobot is able to simulate, some applications of the simulation are shown in Section 6, the paper ends with the conclusion in Section 7.

2 Architecture

2.1 Components of the Simulator

As shown in Fig. 1, SimRobot consists of several modules that are linked to one single application. This approach, which is different from many other client/server-based simulation concepts, has been chosen because it offers the possibility of halting or stepwise executing the whole simulation without any concurrencies. It allows also a more comprehensive debugging of the executed robot software.

The main components of SimRobot are:

SimRobotCore. The simulation core, which may also be qualified as engine or kernel, is the most important part of the application. It models the robots and the environment, simulates sensor readings, and executes commands given by the controller or the user. Even most parts of the visualization are integrated into the simulation core.

```
<Hinge name="subWheelAxis">
    <AnchorPoint x="0.022" y="0" z="0"/>
    <Axis x="0" y="1" z="0"/>
    <Elements>
        <Cylinder radius="0.006" height="0.004">
            <Rotation x="90" y="0" z="0"/>
            <Appearance ref="VeryDarkGray"/>
            <PhysicalAttributes>
                <Mass value="0.02"/>
            </PhysicalAttributes>
        </Cylinder>
    </Elements>
</Hinge>
```

Fig. 2. An excerpt from a scene description using the `RoSiML` language. A sub-wheel of a Small Size robot (see Sect. 6) is modeled via a hinge joint and a cylinder.

The kernel is platform independent. It is connected to a user interface and a controller via a well-defined interface. This enables an easy porting to other platforms as well as the embedding into other applications. A previous version had been used in the *RobotControl* software of the GermanTeam [10]. At the moment, the current kernel becomes integrated in the Linux framework of our Small Size team (see Sect. 6).

GUI. The user interface is responsible for the display of information (e. g. different views of the simulated scene) and for the interaction with the user. It is described in more detail in Sect. 3.

Controller. The controller implements a sense-think-act cycle. In each simulation step, it is called by the simulation, reads the available sensors, plans the next action, and sets the actuators to the desired states. A controller which is suitable for the modeled scene has to be provided by the user. Normally, it contains the control software of the simulated robots, but it may also be left empty.

Scene. The specification of the robots and the environment, in the context of SimRobot named as *scene*, is modeled via an external XML file and loaded at runtime. It is described in the following section.

2.2 Specification of Robots and Their Environment

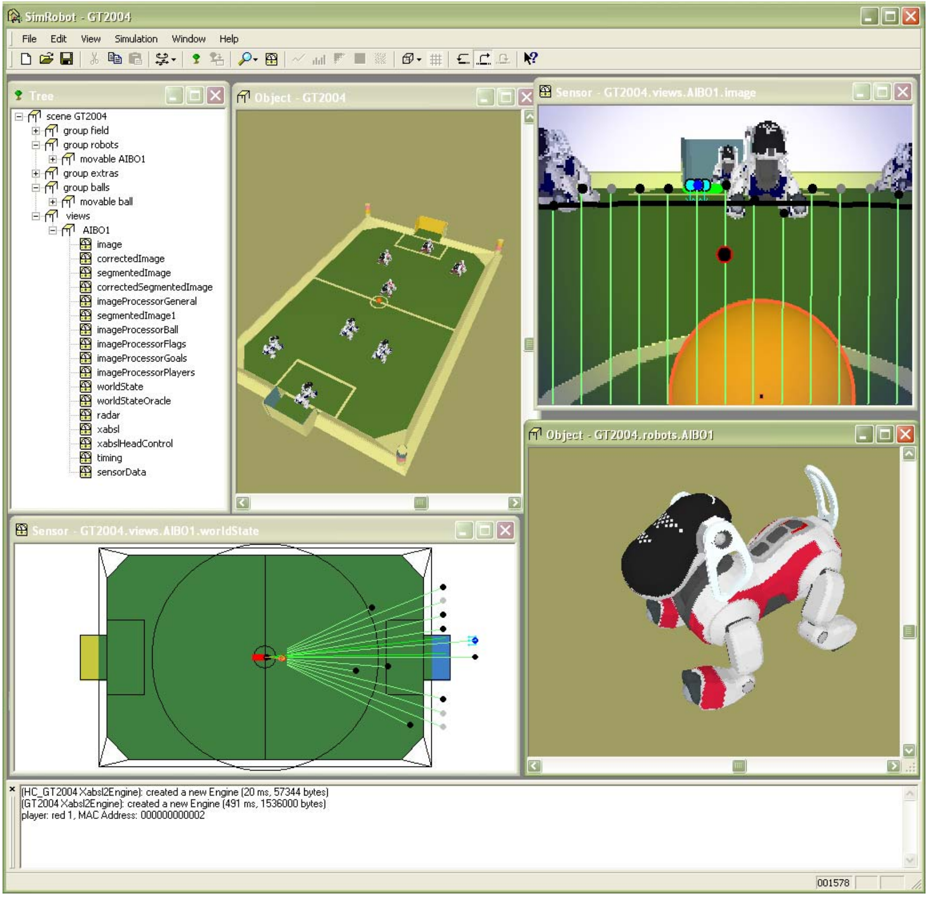
The use of an external specification language allows the modeling of scenes without any modifications or extensions of the source code of the simulator. Thus the modeling process becomes simpler and people without programming skills are also enabled to use the simulator.

Together with researchers from the Fraunhofer Institute for Autonomous Intelligent Systems, the specification language `RoSiML` (Robot Simulation Markup Language) [11] has been developed (see example in Fig. 2). It is a part of a joint effort to establish common interfaces for robot simulations. The aim is to exchange components between different simulators and to allow the migration of robot models among simulators without any complicated adaptions.

The language by itself has been specified in XML Schema. This is a popular choice, since XML is supported by a variety of editors and there also exist many ready-to-use parsers. The cascaded structure of XML documents is also quite suitable to reflect the structure of scenes inside the simulator, as it uses the scene graph approach from computer graphics to organize and process all elements.

3 User Interface

The user interface (Fig. 3) of SimRobot has been designed to allow as much visualization and interaction as possible as well as to be flexible enough to handle



```
(HC_GT2004 Xabsl2Engine): created a new Engine (20 ms, 57344 bytes)
(GT2004 Xabsl2Engine): created a new Engine (491 ms, 1536000 bytes)
player: red 1, MAC Address: 000000000002
```

Fig. 3. The user interface of SimRobot while simulating the German Team 2004. The internal frames show (from left to right, top to bottom): the object tree, a view of the whole scenario, a simulated image of an AIBO camera including overlays from the image processor of the simulated software, a user defined view of a robot's world model, a close view of a single robot, and the console.

simulations of any different kind of environments. Therefore a tree of all objects of the scene is the starting point for all user operations. Each node of that tree may be selected to open a view for that kind of object. In case of actuators (e. g. a hinge joint), a control for direct manipulation is opened. For sensors, several different visualization modes are implemented. Through this concept, it is also possible to open several views of arbitrary subsets of the scene graph (as shown in Fig. 3). These views offer a zoom, rotation, panning and different grades of detail as well as the possibility to switch between the appearance and the physical model of single objects, as shown in Fig. 4. Furthermore, it is possible to interactively drag and drop and rotate objects inside the scene or to apply a momentum to an object (e. g. to let a ball roll). This is quite useful to arrange different settings while testing e. g. a robot behavior.

To add own views to a scene (as e. g. the world state in Fig. 3), an interface for user-defined views has been implemented which enables the definition of own debug drawings from inside the controller.

Other elements of the user interface are an editor for the scene description files and a console for text output from the controller.

4 Physics and Actuators

As aforementioned, ODE is used for simulating rigid body dynamics. Therefore, the set of simulated objects results from the abilities of that engine. Nevertheless, we had to implement several extensions to fulfill the requirements of our general approach.

4.1 Rigid Bodies

For the design of robot shapes and the environment, several rigid bodies (mostly adopted from ODE) may be used. The basic bodies are: Box, Cylinder, Capped-Cylinder and Sphere. Each of them has divers attributes describing its appearance (e. g. color) and physical behavior (e. g. mass or friction).

This set has been extended by the so-called ComplexShape. A body of that class has a graphical representation based on a number of geometric primitives and a physical representation based on a set of basic bodies which may approximate the shape. This approach allows the use of sightly detailed elements with an accurate dynamic and collision behavior. The parts of the AIBO model in Fig. 4 have been described by ComplexShape objects.

4.2 Actuators

To achieve a high grade of flexibility in creating robots, SimRobot supports various kinds of joints corresponding to the joints provided by ODE. There are simple rotational joints with one axis or two perpendicular fixed axes, a translational joint, a ball and socket joint and a simple wheel suspension like joint with free rotation about one axis and rotation and compression along the other axis. The joints can connect two movable rigid bodies or one body with

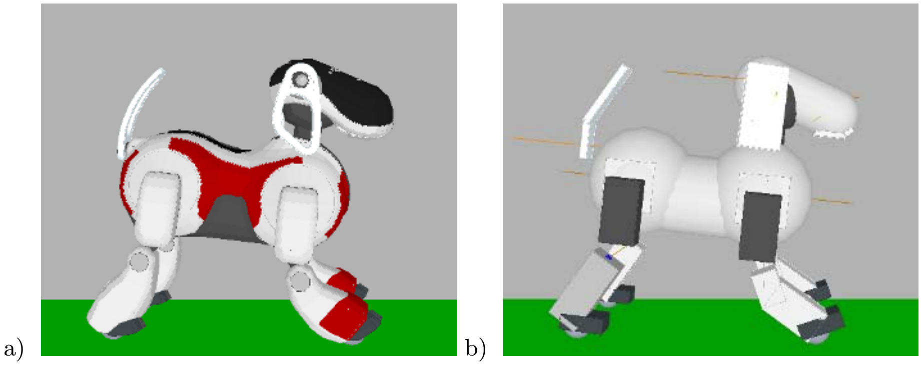


Fig. 4. A Sony AIBO model consisting of a set of ComplexShape objects. a) The graphical representation with details such as toes, tricot elements and LEDs on the head. b) The physical representation consisting of simple boxes, spheres, and capped cylinders.

the static environment. The range of motion of a joint can be limited. Joints can be unpowered or a motor can be attached to each axis. Due to the fact that the motor provided by ODE has no specific controller, we implemented a P and a PID controller to simulate servo motors. Additional controllers can easily be integrated, if necessary. Besides the servo motor, a simple velocity controlled motor is also provided. In ODE, joints are frictionless, i. e. a pendulum will never stop swinging if no collision occurs. So we implemented a simple friction model for damping motion in unpowered joints.

4.3 Automatic Generation of Object Compounds

In SimRobot it is possible to specify complex rigid bodies through using the provided simple elements without explicitly summarizing them with additional XML tags in the scene description or performing additional calculations by the user. The structure of the scene tree is used for this automatism. In the scene tree, all elements of a subtree beneath a joint down to the leaf nodes or to other joints are treated as one single physical body. All elements in this compound are stuck together and behave as one single object. For correct dynamical behavior, the masses, centers of masses and inertia tensors of all contained objects are combined to a single rigid body. The geometrical representations of all combined objects build the collision behavior of the compound object. In addition, single objects or object groups can be excluded of the above described automatism of combining and are treated as independently moving objects or compound objects.

5 Sensing

SimRobot realizes sensor simulation via a set of generic sensor classes, i. e. it does not include specific sensors such as a special laser range finder or similar devices. The available sensor classes are:

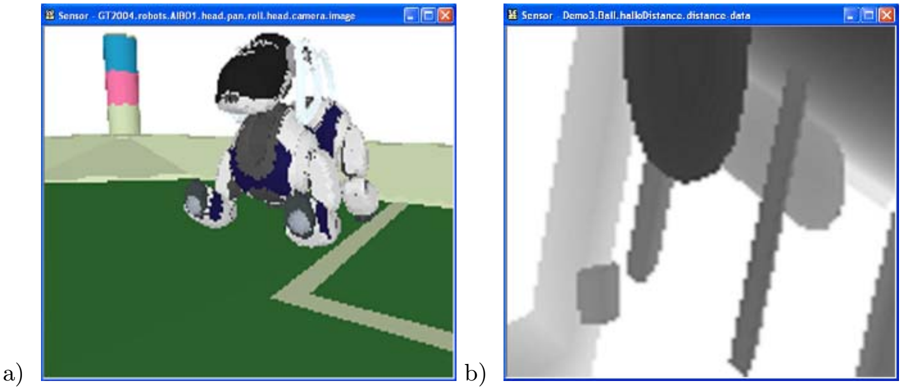

Sensor - GT2004.robots.AIBO1.head.pan.roll.head.camera.image
a)
Sensor - Demo3.Ball.halloDistance.distance.data
b)

Fig. 5. The results from two different sensors: a) An image from a simulated AIBO camera and b) a depth image inside a demo scene mainly consisting of boxes and cylinders

Camera. The camera sensor generates a two-dimensional array of RGB pixels which have a color depth of 24 bits (one example is shown in Fig. 5a). Aside from the standard perspective projection, it is also possible to use a spherical projection with an equal angular distance between all pixels. To speed up the process of image generation significantly, SimRobot is able to support hardware accelerated off-screen rendering. This is a feature which several manufacturers implement on their graphics hardware nowadays.

Distance Sensor. This sensor is quite equal to the camera, but instead of pixels, it returns distances gained from the depth buffer of rendered images. Due to its generic approach, there exist several applications for this sensor: the generation of depth images (Fig. 5b), simulating a laser range finder (Fig. 6b), or being modeled as a single value PSD sensor in an AIBO robot.

Bumper. For detecting the collision of objects, e. g. to model a touch sensor, the so-called Bumper has been implemented. Unlike all other sensors, it is not a special object in the scene graph. Furthermore, each body or group of bodies may be assigned to be collision sensitive. This allows the creation of arbitrarily shaped touch sensors. The information about collisions is directly gained from the dynamics engine. As an addition, it is also possible for the user to interactively provoke the sensing of a collision. We have used this feature e. g. for pressing the buttons of simulated AIBO robots.

Actuator State. There also exist interfaces for inquiring the current states of actuators. This includes the angles of joints as well as the velocities of motors.

6 Applications

As aforementioned, SimRobot has been used by the GermanTeam to simulate the robots and the environment of the Sony Four-legged League. In previous years, a kinematic version of SimRobot has been used, which needed several work arounds

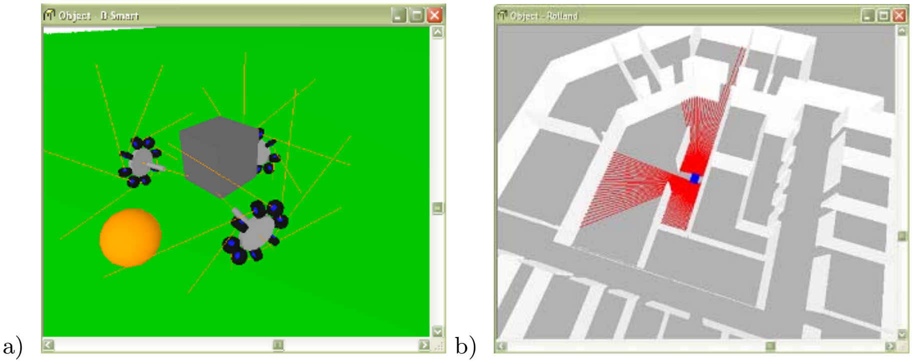


Fig. 6. Applications of SimRobot: a) The physical representation of a Small Size platform with an omnidirectional drive. The lines around the wheels denote the axes of the sub-wheels. b) A simulated office environment with a navigating robot. The lines starting at the robot denote the rays of its laser range finders.

to simulate correct body postures e. g. when executing kick motions. Due to the absence of a collision handling, the ball had also to be moved manually. Together with the physical simulation, a new model of the AIBO ERS-7 robot, which is shown in Fig. 4, has been created. The appearance was kept, but all elements are now additionally represented by rigid bodies, which are connected with adequate joints to allow an accurate simulation of all 20 degrees of freedom. This model enables a direct execution of the desired motions as well as an accurate handling of collisions with other robots or the ball.

A completely different kind of robots is used in the RoboCup Small Size League. To allow fast and flexible motions, wheel-based robots with omnidirectional drives are used. These consist of three or four wheels which are arranged in a triangle or a rectangle respectively. Each of these wheels is surrounded by a set of small passive sub-wheels which enable the robot to move sidewards. This approach has also found its way into the Middle Size League. To integrate SimRobot into the environment of our Small Size team B-Smart [12], a platform with such a drive has been modeled, as shown in Fig. 6a. The XML description of a single sub-wheel has already been presented in Fig. 2. Though having a completely different structure than the previously modeled walking robot, the platform has been simulated successfully, showing the expected motion behavior when driving around.

An application outside the RoboCup domain is the simulation of *Rolland* - The Bremen Autonomous Wheelchair [13]. This robot has a differential drive and is equipped with two laser range finders which are connected to a standard notebook which executes the control software. Rolland performs navigation tasks in office environments, as shown in Fig. 6b. Since this robot currently has a two dimensional model of the world and is programmed to never collide with other objects, the physical simulation is of minor usefulness for this platform. Nevertheless, it demonstrates the application of SimRobot in a large-scale environment and the usage of distance sensors.

7 Conclusion and Future Works

This paper presented SimRobot, a robot simulator which supports rigid body dynamics and the simulation of a variety of sensors and actuators. Through its generic concept and the use of a special modeling language, it is able to simulate arbitrary user-defined robots without any modifications of the simulator itself. This has been shown by means of several examples of completely different robot platforms which have been simulated successfully. The flexible approach of the user interface has been able to offer a variety of visualizations and options for interaction for each simulated environment.

The German Team as well as the B-Smart team will use this simulator in 2005. It will also be made available to other RoboCup teams, as described in the following section.

Among other things, future works will concentrate on the simulation interface standardization efforts described in Sect. 2.2. These will include generic plug-in interfaces for sensors and actuators as well as the definition of a common controller interface.

Availability of SimRobot

The simulator presented in this paper will be released under an open source license in the near future. The most current available version, which does not include the dynamics engine but most other features, has been released as a part of the German Team 2004 code release [14]. An up-to-date binary version for Microsoft Windows which includes several examples is also available [15]. A release of a Linux version is planned.

Acknowledgements

The authors would like to thank all people who contributed to the development of SimRobot by directly providing code or by testing and reporting problems, the authors of the Open Dynamics Engine for making available the basis for our physical simulation, and the researchers from Fraunhofer AIS for their contributions to the Robot Simulation Markup Language.

References

1. Zagal, J.C., del Solar, J.R.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Artificial Intelligence, Springer (2004)
2. Smith, R.: Open Dynamics Engine - ODE (2005) www.ode.org.
3. Go, J., Browning, B., Veloso, M.: Accurate and flexible simulation for dynamic, vision-centric robots. In: Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04). (2004)
4. Go, J., Browning, B., Veloso, M.: Carnegie Mellon UberSim Project (2004) http://www-2.cs.cmu.edu/ robosoccer/ubersim/.

5. Koenig, N., Howard, A.: Gazebo - 3D multiple robot simulator with dynamics (2004) http://playerstage.sourceforge.net/gazebo/gazebo.html.
6. Gerkey, B., Vaughan, R.T., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: Proceedings of the 11th International Conference on Advanced Robotics, Coimbra, Portugal (2003) 317 – 323
7. Michel, O.: Cyberbotics Ltd. - WebotsTM: Professional Mobile Robot Simulation. **1** (2004) 39–42
8. Cyberbotics Ltd.: Cyberbotics Webots (2005) `http://www.cyberbotics.com/products/webots/`.
9. Röfer, T.: Strategies for using a simulation in the development of the Bremen Autonomous Wheelchair. In Zobel, R., Moeller, D., eds.: Simulation-Past, Present and Future, Society for Computer Simulation International (1998) 460–464
10. Röfer, T., Laue, T., Burkhard, H.D., Hoffmann, J., Jüngel, M., Göhring, D., Lötzsch, M., Düffert, U., Spranger, M., Altmeyer, B., Goetzke, V., v. Stryk, O., Brunn, R., Dassler, M., Kunz, M., Risler, M., Stelzer, M., Thomas, D., Uhrig, S., Schwiegelshohn, U., Dahm, I., Hebbel, M., Nisticó, W., Schumann, C., Wachter, M.: GermanTeam RoboCup 2004 (2004) Only available online: http://www.robocup.de/germanteam/GT2004.pdf.
11. Ghazi-Zahedi, K., Laue, T., Röfer, T., Schöll, P., Spiess, K., Twickel, A., Wischmann, S.: Rosiml - robot simulation markup language (2005) http://www.tzi.de/spprobocup/RoSiML.html.
12. Kurlbaum, J., Laue, T., Lück, B., Mohrmann, B., Poloczek, M., Reinecke, D., Riemenschneider, T., Röfer, T., Simon, H., Visser, U.: Bremen Small Multi-Agent Robot Team (B-Smart) Team Description for RoboCup 2004. In: RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Artificial Intelligence, Springer (2005)
13. Mandel, C., Hübner, K., Vierhuff, T.: A Demonstrator for Cognitive Aspects in Service Robotics. In: XXVII Annual Meeting of the Cognitive Science Society (submitted). (2005)
14. German Team: German Team web site. (2005) http://www.germanteam.org.
15. Röfer, T.: SimRobot Website (2005) http://www.tzi.de/simrobot.

Agent Community Extraction for 2D-RoboSoccer

Ravi Sankar Penta and Kamalakar Karlapalem

Center for Data Engineering,
International Institute of Information Technology, Hyderabad
ravi_s@research.iiit.net, kamal@iiit.net

Abstract. Agents perform tasks to maximize their benefits. There are several instances where the agent can not perform a task individually. In these situations, agents need to cooperate and coordinate with other agents effectively and efficiently to maximize their benefits in a limited time. In several domains, we can analyze the behavior of successful agents and the way they interact with other agents forming strong communities or coalitions. This knowledge can be used by a new or unsuccessful agent to collaborate with other agents that gives maximum benefit under strict time constraints. This paper proposes a generic procedure for extracting these hidden communities that can be used by the agents in a productive manner. We tested the framework on robosoccer simulation environment and our experiments indeed show drastic increase in both agent and team performance.

1 Introduction

A multi-agent system consists of a number of autonomous homogeneous or heterogeneous agents which interact with each other to achieve a common goal or to maximize their own benefit. The agents in a multi-agent system perform activities. An activity can be a simple task that can be done by a single agent or it can be a complex task involving two or more agents. A multi-agent system has the perspective at macroscopic level and at microscopic level. At macroscopic level, issues such as starvation or overloading agents with many tasks are addressed. Whereas at microscopic level, issues such as agent's local decision making and increasing the agent's profit are discussed.

In this paper, we present, how to use the history of the system to extract hidden and non-overlapping agent communities of various strengths based on both agent capabilities and their interactions. The hidden agent communities can be used by a new or less successful agent to participate in a specific community with agents having similar/complementary capabilities and behavior for improving its performance.

1.1 Related Work

To improve the performance of the agent and multi-agent system, considerable amount of work has gone in forming appropriate coalitions [7, 1] based on the environment, set of tasks to be achieved and agent capabilities with reasonable time consumption. Modeling of the agent communities by grouping the agents with similar objectives is discussed in [8, 2]. Extracting reputation of agents in the electronic communities using

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 184–195, 2006.

only local information without relying on feedback in [3]. Improving agent's decision making using coordination graphs via a context-specific decomposition of the problem into smaller sub-problems is demonstrated in [5]. Using the history (logfile), behavior patterns of an individual agent with the help of self-organizing maps (SOMs) is addressed in [9]. We also used the history, to obtain behavior patterns for a group of agents (team) by extracting communities using reputation of agents and interactions among the agents for improving both the agent and the team performance.

1.2 Motivation

In many real world domains, there are several restrictions on the system like incomplete information about environment and restrictions on the agents like limited thinking time, resource constraints, etc. Agents need to be more reactive in these kind of situations. For example, consider the role of a coach in robosoccer simulation environment. The coach needs to form the best team and guide the players. Best team need not be the one having players with maximum individual capabilities. The way the players cooperate in achieving the goals, play a vital role in the formation of best team. These interactions can only be studied by analyzing the previous matches of the players. This analysis also helps in guiding the players in achieving their goals. Extracted knowledge can be used by the agent in taking better possible actions within strict time constraints. For example, in robosoccer, the thinking time for an agent is limited. Consider the case, where the agent wants to pass the ball to one of its teammates and there is no time for choosing the best teammate. If the agent has knowledge through the discovered agent community about a particular teammate who is good at receiving the ball in that situation, then the agent can safely pass to that teammate instead of passing to some random agent.

Our main contributions in this paper are as follows:

- Extraction of hidden agent communities for an activity,
- Ranking of the extracted communities,
- Extraction of grand communities (community consisting of agents which can perform many of the activities),
- Best fit community for any new agent,
- Selection of top 'k' agents in a system for an activity or set of activities, and
- Identification of drag-agents in a community.

The paper is organized as follows. The problem and our assumptions are stated in section 2. In subsections 2.1 to 2.6, we illustrate, how to achieve the above mentioned contributions. Robosoccer simulation environment, for validating our approach is briefly described in section 3. The results obtained and their analysis are presented in section 3.3. Finally, we conclude the paper in section 4.

2 Problem Statement

Let $A = \{A_1, A_2, ...A_n\}$ be the set of agents and $Act = \{act_1, act_2, ...act_m\}$ be the possible activities executed by them in the multi-agent system. Let $S = \{S_1, S_2, ...S_p\}$ be the sessions of the system. Sessions can be static or dynamic, complete or incomplete. A session can be of fixed or varied time intervals. In robosoccer, a session can be a match between the teams. Given the history, the problem is how to extract hidden agent communities for helping a new or unsuccessful agent in improving its performance.

We assume that the agents goals remain constant across all sessions. We find communities among the common set of agents across all sessions.

2.1 Extraction of Hidden Communities

Graphs are used to model the agent interactions and its capabilities. Since any activity is a series of interactions in some specific order, an activity is modeled as a graph. Let $DG^{S_j}_{act_i}(A, E)$ be a directed graph (a forest of directed graphs) for an activity act_i during the session S_j. The nodes of the graph represents agents A and any directed edge $A_p{\rightarrow}A_q$ in E represents an interaction between the agents A_p and A_q. We call this directed graph as activity graph. Let e_{pq},s_{pq},f_{pq} be the total number of interactions, successful interactions and failed interactions respectively among the agents A_p and A_q. The weight of any edge $A_p{\rightarrow}A_q$ gives the strength of interaction between the agents A_p and A_q, given by

$$w_{pq} = \frac{s_{pq}}{e_{pq}}, \quad 0 \leq \ w_{pq} \leq 1$$

If all the interactions required to fulfill the activity among the agents are successful, then the activity is considered as success at that instant during the session. For example, agents A_p,A_q,A_r can complete activity Act_k if they interact in the order given in the Figure 1. Suppose, they got 2 chances to perform Act_k during the session. Act_k failed in the first chance because of failed interaction between A_q and A_r. In the second chance, they fulfilled Act_k. The corresponding successful interactions and the weights they obtained are shown in the Figure 1.

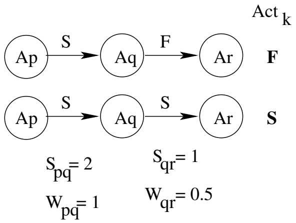


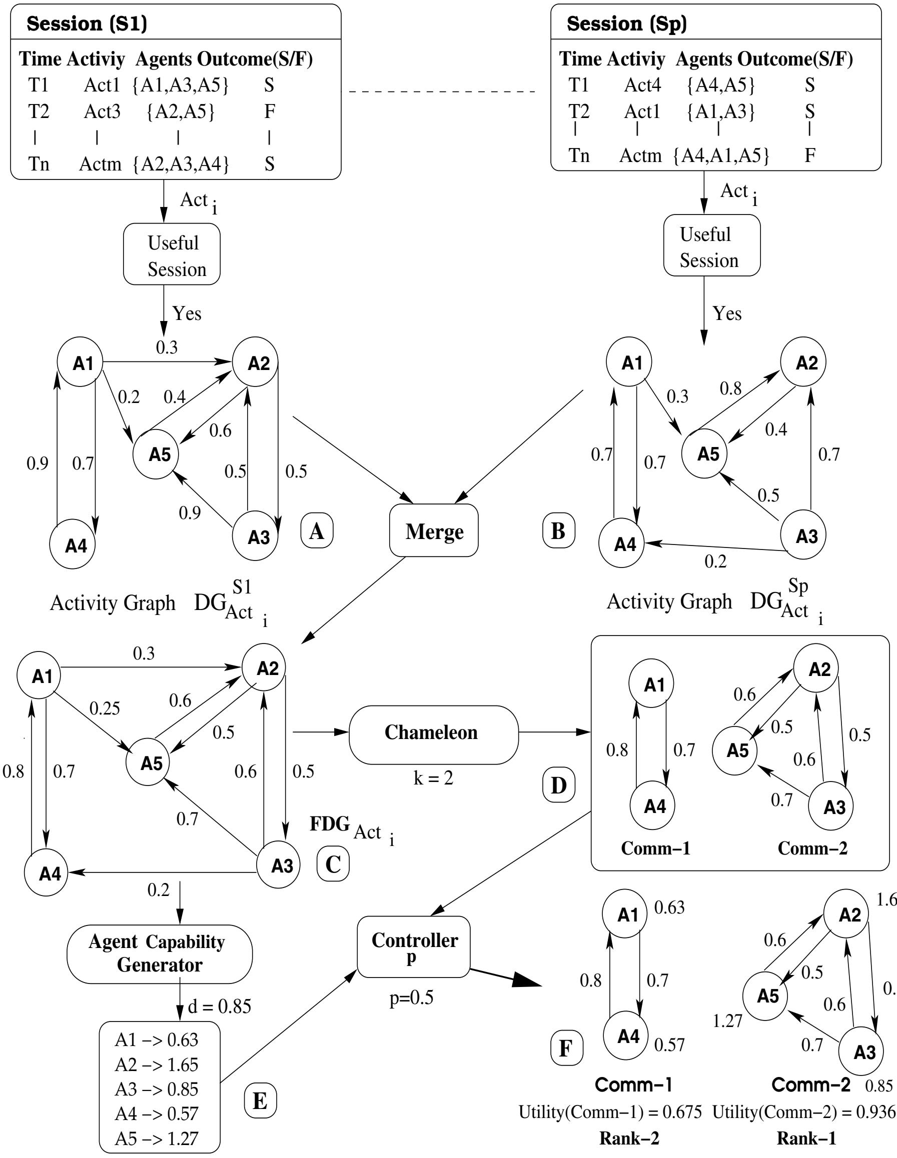
Fig. 1. Success or Failure of Activity Act_k

Generation of Final Activity graph. Final activity graph for an activity is generated by merging all the activity graphs of successful sessions. A session is successful for an activity, if the success rate of the interactions is above some specified threshold. In robosoccer, a successful session can be registering a win in the match. A session S_j for an activity act_i is successful if it satisfies

$$\frac{\sum\limits_{\forall A_p \rightarrow\ A_q} s_{pq}}{\sum\limits_{\forall A_p \rightarrow\ A_q} e_{pq}} > \lambda_i, \quad A_p, A_q \in DG^{S_j}_{act_i}$$

where λ_i is the minimum confidence decided by the user for an activity act_i. In robosoccer λ_i=0 implies, considering all the matches for analysis for an activity act_i. Taking λ value around 0.5 means that the user is interested in those matches which are competitive.

Let $S = \{S_1, S_2...S_k\}$ be the successful sessions for an activity act_i and FDG_{act_i} be its final activity graph. FDG_{act_i} is generated by merging all the successful activity



Session (S1)

Time	Activiy	Agents	Outcome(S/F)
T1	Act1	{A1,A3,A5}	S
T2	Act3	{A2,A5}	F
⋮	⋮	⋮	⋮
Tn	Actm	{A2,A3,A4}	S

Session (Sp)

Time	Activiy	Agents	Outcome(S/F)
T1	Act4	{A4,A5}	S
T2	Act1	{A1,A3}	S
⋮	⋮	⋮	⋮
Tn	Actm	{A4,A1,A5}	F

Fig. 2. Procedure for Extraction and Ranking of agent communities for an activity Act_i

graphs. Let $w_{pq}^{S_j}$ be the weight of an edge $A_p \rightarrow A_q$ in the session S_j. Weight of any edge $A_p \rightarrow A_q$ in FDG_{act_i} gives the strength of the interaction between the agents A_p and A_q. It is given by

$$w_{pq} = \frac{\sum_{j=1}^{k} w_{pq}^{S_j}}{\sum_{j=1, w_{pq}^{S_j} \neq 0}^{k} 1}$$

Consider a system consisting of 5 agents A_1,A_2,A_3,A_4,A_5 and 2 successful sessions S_1 (Figure 2(A)), S_p (Figure 2(B)). Generated final activity graph FDG_{act_i} for an activity act_i using the activity graphs $DG_{act_i}^{S_1}$ and $DG_{act_i}^{S_p}$ is shown in the Figure 2(C).

Agent Community Extraction. Extracting communities helps in identifying similar or complementary group of agents performing some activity at a particular success level. Agent communities for an activity are extracted by breaking weak edges in the corresponding final activity graph. Using fixed threshold for cutting an edge extracts only strong communities disregarding weak communities. But Chameleon [4] clustering algorithm can be used for extracting both strong and weak communities. Chameleon is a dynamic hierarchical clustering algorithm that measures the similarity of two clusters based on relative intra-closeness and inter-closeness between the clusters. The clusters obtained by chameleon are our agent communities.

The communities obtained by applying chameleon on FDG_{act_i} is shown in the Figure 2(D). Since we have only 5 agents in FDG_{act_i}, the value of k is set to 2 to obtain 2 communities. The value of k can be varied to obtain very strong or weak communities.

2.2 Ranking of Agent Communities

Generation of Agent Capabilities. The strength of the community depends on the agent capabilities and their interactions with other agents. Any successful or failed interaction $A_p \rightarrow A_q$ depends on both agents, A_p and A_q. Number of agents which A_p has interacted with, is given by $OutDegree(A_p)$. It denotes the total number of interactions initiated by agent A_p and its success rate is given by

$$InitSuccess(A_p) = \frac{\sum_{\forall A_p \rightarrow A_q} w_{pq}}{OutDegree(A_p)}$$

Refer to the Figure 2(C), $OutDegree(A_1)$ in FD_{act_i} is 3
$InitSuccess(A_1)$ in FD_{act_i} is (0.3+0.25+0.7)/3 = 0.416

Interaction $A_p \rightarrow A_q$ denotes agent A_p's trust on A_q. Trust is cumulative and the number of agents who trust agent A_q is given by $InDegree(A_q)$. Trust gained by any agent A_q is given by

$$TrustGain(A_q) = \sum_{\forall A_p \rightarrow A_q} w_{pq}$$

In robosoccer, suppose the active player A_p gains equal payoff by passing the ball to either A_q or A_r. When A_p chooses player A_q it implies player A_p has more trust on A_q compared to player A_r.

Refer to the Figure 2(C), $InDegree(A_5)$ in FD_{act_i} is 3
$TrustGain(A_5)$ in FD_{act_i} is (0.25+0.5+0.7) = 1.65

Importance of any agent in the system depends on the trust it got from others. Trust contribution to an agent A_q by different agents need not be equal. Trust obtained by important or successful agents will count more compared to unsuccessful or failed agents. This is similar to PageRank [6], where the rank of the page depends on its back-links. The links which come from important pages conveys more importance to a page compared to other pages.

The capability of an agent is calculated based on the capabilities of all the trusting agents. The capability accumulated by the agent is equally shared among all the agents with whom the agent can initiate an interaction. Let the parameter 'd' be the damping factor which can be set between 0 and 1. Then the capability of an agent A_q is given by

$$C(A_q) = (1-d) + d \times \sum_{\forall A_p \rightarrow A_q} \frac{C(A_p)}{OutDegree(A_p)} \tag{1}$$

Converged value of possible recursive equation (1) will give the capability of an agent.

Refer to the Figure 2(C), let all the initial agent capabilities in FD_{act_i} be 1.0 and 'd' is set to 0.85. The capabilities obtained from FD_{act_i} using equation (1) for the agents A_1, A_2, A_3, A_4, A_5 are 0.63, 1.65, 0.85, 0.57, 1.27 respectively as shown in the Figure 2(E).

Utility for any pair of Agents. Two factors namely individual capabilities and success rate of interactions determine the strength of any pair of agents who can interact between themselves. The relative importance between these two is a subjective issue. The utility or strength of any two agents A_p and A_q is given by

$$Utility(A_p \rightarrow\ A_q) = p \times \frac{(C(A_p) + C(A_q))}{2} + (1-p) \times\ w_{pq}$$

Where the first term of $Utility$ function represents utility of agent capabilities and second term represents utility of interactions among themselves. p is the controlling parameter and it decides the relative importance among the agents capabilities and their interactions. p is domain dependent and its value for a system is decided by the user.

Utility of the Community. The strength of all pairs of agents who can interact denotes the strength of the whole community. The utility of any community is given by

$$Utility(Comm) = \frac{\sum\limits_{\forall A_p \rightarrow A_q \in\ Comm} Utility(A_p \rightarrow\ A_q)}{\sum\limits_{\forall A_p \rightarrow A_q \in\ Comm} 1}$$

Ranking of the communities is done by sorting $Utility(Comm)$ in descending order. The procedure for extraction and ranking of agent communities for an activity Act_i is shown graphically in the Figure 2.

2.3 Extraction of Grand Communities

Agents in the grand community can do many activities in the system. This is similar to all-rounders in a game. Grand communities are generated from grand activity graph using chameleon (discussed in section 2.1). Grand activity graph is generated by merging all the final activity graphs. Since, some activities are important compared to others for achieving the final goal, there is a need to differentiate these activities. Let α_i be the relative weight for an activity act_i that is decided by the user depending on the relative importance between the activities. A simpler method for assigning α_i values is to sort all the activities in the ascending order of preference and number these activities. α_i value for a particular activity can be its corresponding number.

Let GDG be the grand activity graph obtained by merging FDG_{act_i} $\forall$ i. Weight of any edge in GDG is given by

$$w_{pq} = \frac{\sum\limits_{\forall i} \alpha_i \times w_{pq}^{act_i}}{NumActivities}.$$

2.4 Best Fit Community for Any New Agent

Any new agent desires to join the community that closely matches its own behavior and capabilities. A community will be relatively small compared to the complete multi-agent system. So an agent can study the general behavior of the other agents and the interactions among them within the community. A new agent can't predict the outcome of interactions with the other agents. So we consider only the capability of the new agent for finding its best fit community. Initial capability of a new agent is set to 0.5. Over time, the capability of the new agent will change, reflecting its contribution to various communities. We know the capabilities of all the agents within a community (section 2.2). The best community for a new agent will be the community having least variance with respect to the new agent capability. The best community $Comm^\star$ for any new agent A_q is given by

$$Comm^\star = argMin_{Comm}\left[\frac{\sum\limits_{\forall A_j \in Comm} (C(A_j) - C(A_q))^2}{NumAgents\ in\ Comm}\right]$$

Suppose a new agent having capability 0.5 wants to join a best fit community from the communities obtained in the Figure 2(F).

$argmin\{[\frac{(0.13)^2+(0.07)^2}{2}], [\frac{(1.05)^2+(0.35)^2+(0.77)^2}{3}]\}$
= $argmin\{0.01, 0.60\}$

So Comm-2 is the best fit for this new agent.

2.5 Selection of Top 'k' Agents in a System

Selection of top 'k' agents is a subjective issue. For example in robosoccer environment, the captain or the coach generally prefer 3 or 4 specialized defenders, 3 or 4 specialized attackers and 4 or 5 all-rounders (who defend and attack reasonably). It makes the team well balanced and competitive. The specialized defenders are selected from the final activity graph of $Activity_{defending}$, specialized attackers are selected from $Activity_{attacking}$ and all-rounders are selected from grand activity graph which involves most of the activities in the system. The agents relative capabilities and activity graphs for each activity and for all activities which are considered can be calculated (sections 2.1, 2.2). Using this, we can select the agents accordingly.

2.6 Identification of Drag-Agents in a Community

Agents who get high amount of trust from others and have many unsuccessful interactions initiated by them are called drag-agents. Drag-agents reduce the performance of the community due to many unsuccessful interactions with other agents in the community. Agents individual benefits will be decreased if they trust any of the drag-agents. Drag-agents are the root cause for the failure of the activity at many instants. By identifying the drag agents, community performance can be increased either by giving least preference to drag agents or by replacing the drag agents with appropriate agents.

Any agent having considerable amount of trust from other agents should perform at least better than an average agent in the community. The amount of trust got by any agent depends on the percentage of agents in the community who are trusting the agent. Therefore, $Drag(A_p)$ is defined as

$$Drag(A_p) = \left[\frac{\sum_{i=1}^{N} InitSuccess(A_i)}{N} \right] \times (1 + \frac{InDegree(A_p)}{N}) - InitSuccess(A_p)$$

$InitSuccess(A_p)$ is the successful interactions initiated by A_p (defined in section 2.2). $Drag(A_p)$ signifies the weakness of the agent A_p within the community. All the agents whose $Drag > \tau$, are the drag-agents in the community. By sorting $Drag$ values for all the agents in decreasing order, the threshold τ can be decided by the user.

Consider Comm-2 in the Figure 2(F),
$InitSuccess(A_5)$= 0.6, $InitSuccess(A_2)$= 0.5
$InitSuccess(A_3)$= 0.65
Average $InitSuccess$ = (0.6 + 0.5 + 0.65)/3 = 0.58
$Drag(A_5)$= 0.58 * (1 + 2/3) - 0.6 = 0.366
$Drag(A_2)$= 0.58 * (1 + 2/3) - 0.5 = 0.466
$Drag(A_3)$= 0.58 * (1 + 1/3) - 0.65 = 0.123
When τ=0.4, then the agent A_2 is the drag agent.

3 Experiments

We tested our approach to extract hidden agent communities for 2D robosoccer environment.

3.1 RoboSoccer Simulation Environment

Soccer match is played between two teams on a simulation environment which is controlled by a centralized server. 11 agents from each team connect to the server and perceive sensory information like aural, visual and body state in each cycle. The match duration is 10 minutes. Each minute is divided into 600 cycles. Each agent has to decide the actions within a cycle (100ms). If it is valid, the server simulates the requested action. The possible actions can be kick, dash, move, turn neck, turn body, say and hear. The environment is unreliable (sensory information contains noise), dynamic and incomplete (limited vision for each agent). Like real world situations, here the agents are heterogeneous and have limited stamina. Heterogeneous in the sense that the agents with more kick power lose stamina at higher rate and the agents who can perceive the environment accurately can only view up-to limited distance. Each team tries to win the match, for which agents needs to take an action that maximizes their winning chance. To achieve this in an effective manner, agents need to cooperate.

3.2 Dataset Generation

The dataset is generated by playing 50 matches with the team $Kshitij$ against other teams in robosoccer simulation league. There are several activities possible in this system like ball possession, dribbling, passing, defending, attacking etc. The team $Kshitij$ is used to extract communities for the desired activities to analyze its strengths and the weakness based on the history of games played. The activities that are considered is shown in the Table 1. These activities are considered since our main concentration is on the ball control and scoring the goals. Snapshot of the dataset is shown in the Table 2.

Table 1. Activities

Act Num	Act Name	Parameters
Act_1	Ball Possession	$\lambda = 0.5$, $timeSlot = 30cycles$
Act_2	Straight Pass	$\lambda = 0.70$
Act_3	Leading Pass	$\lambda = 0.62$

Table 2. Snapshot of the dataset

TimeCycle	ActNum	Agents	OutCome(S|F)
4	Act_3	A_9, A_{11}	S
18	Act_1	A_4, A_8, A_{11}	F
|	|	|	|
5994	Act_2	A_9, A_7	S

3.3 Results

The team formation of $Kshitij$ is as follows. Agent A_1 acts as goalie, agents $\{A_2$,A_3, A_4,$A_5\}$ are defenders, agents $\{A_6$,A_7,$A_8\}$ are mid-fielders and the agents $\{A_9$,A_{10},$A_{11}\}$ are attackers (can be seen either in the Figure 3 or 4). The agents have

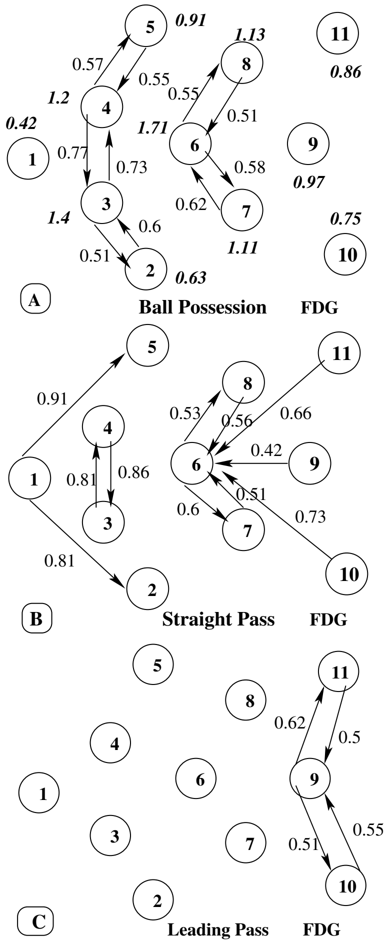


Fig. 3. FDGs of Original $Kshitij$ team

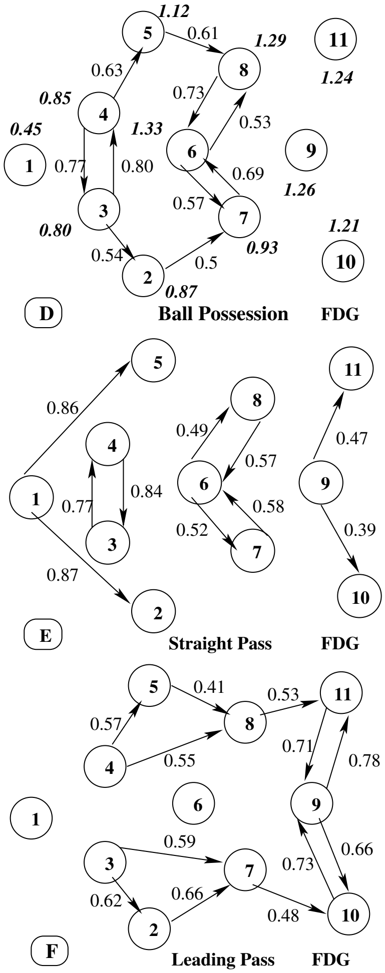


Fig. 4. FDGs of Modified $Kshitij$ team

average quality visual sensors and can view the environment for a considerable distance. Attackers have more kick power and defenders have more stamina.

We applied our approach on the history (Table 2) considering the activities in the Table 1. For ranking the communities, damping factor 'd' is taken as 0.85. For calculating the utility of any pair of agents, equal importance is given to the agent capabilities and their interactions with other agents i.e p=0.5. The communities obtained are shown in the Figure 3. Node values represents agent capabilities. Analysis of these communities for each activity is given below.

Ball Possession. Activity $BallPossession$ is successful if the control of the ball is in our team for a given time period $timeSlot$. Control of the ball can be achieved either by holding the ball for a long time or passing the ball correctly to the teammate or dribbling in the safe direction. The two communities that are obtained for this activity is shown in the Figure 3(A). One contains all the defenders and the other contains all the mid-fielders. So most of the time, the ball is within the defenders or the mid-fielders. But more successful passes between the defenders and the mid-fielders is desired who can pass the ball to the attackers for scoring the goal. There is no coordination among the agents A_9,A_{10},A_{11}. This may be due to strong opponent defense or weak passing among these agents.

Straight Pass. If an agent passes the ball directly to some teammate who is able to receive the ball successfully, then the activity $StraightPass$ is successful. Communities formed for this activity is shown in the Figure 3(B). The success rate of $A_1 \rightarrow A_2$ or $A_1 \rightarrow A_5$ signifies that most of the time, the goalie is passing to the appropriate agent. Agent A_6 is the drag agent, overloaded by the agents $A_7, A_8, A_9, A_{10}, A_{11}$. Since every agent has limited stamina, the agent can't perform all the tasks like kicking, moving etc. continuously for a longer period. Several times, Agents A_9, A_{10}, A_{11} are preferring backward passes. This may be due to unavailability of the agents A_{10}, A_{11} in the appropriate position for receiving the ball.

Leading Pass. Leading pass is passing the ball in front of the agent so that the agent receives the ball before the opponents. Activity $LeadingPass$ is successful if the desired agent receives the ball successfully. In the team Kshitij, leading pass is used only by the attackers and average success is achieved in scoring the goal (Figure 3(C)). This may be due to the inaccurate information possessed by the attackers.

To overcome the problems faced in the above activities, the following modifications are made in the team. Leading pass can be used by the defenders and the mid-fielders whenever possible to avoid excess passes among themselves. Restricting back passes for the attackers to avoid overloading of agent A_6. Dynamic positioning of the agents A_{10} and A_{11} whenever possible, to receive the ball from the agent A_9. Quality of visual sensors changed from average to high for the attackers, as quality is very important for the attackers (near to the goal) compared to the vastness. These modifications improved the team performance, mainly the average number of goals scored against other teams improved by 25-30%.

The new communities obtained by the modified team is shown in the Figure 4. Considerable number of passes between the defenders and the mid-fielders can be seen in the Figure 4(E). No agent in the team is overloaded. The success rate of passes between A_9 and A_{10}, A_9 and A_{11} improved significantly (compare Fig. 3 with Fig. 4).

4 Conclusions

In this paper, we gave a generic procedure for extracting hidden agent communities, ranking them, identifying drag agents within a community, finding best fit community for a new agent and selecting top 'k' agents in the system for an activity. To validate our procedure, we tested for robosoccer simulation environment and got useful results for improving the agent and system performance. Possible future work can be on testing our procedure in different domains, on scalability issues and developing approaches for the cases where the agents can enter or leave the communities (evolving communities).

References

1. B. Blankenburg and M. Klusch. On safe kernel stable coalition forming among agents. In *AAMAS*, 2004.
2. M. van Steen E. Ogston, B. J. Overeinder and Frances M. T. Brazier. A method for decentralized clustering in large multi-agent systems. In *AAMAS*, pages 789–796, 2003.
3. R. Sangüesa J. M. Pujol and J. Delgado. Extracting reputation in multi agent systems by means of social network topology. In *AAMAS*, pages 467–474, 2002.
4. G. Karypis, E. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.
5. Jelle R. Kok, Matthijs T. J. Spaan, and Nikos Vlassis. Multi-robot decision making using coordination graphs. In A.T. de Almeida and U. Nunes, editors, *Proceedings of the 11th International Conference on Advanced Robotics, ICAR'03*, pages 1124–1129, Coimbra, Portugal, june 2003.
6. L. Page and S. Brin. The pagerank citation ranking: Bringing order to the web. In *WWW*, 1999.
7. O. Shehory and S. Kraus. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *AAAI/IAAI, Vol. 1*, 1996.
8. W. Truszkowski and J. Karlin. A cybernetic approach to the modeling of agent communities. In *CIA*, 2000.
9. Michael Wünstel, Daniel Polani, Thomas Uthmann, and Jürgen Perl. Behavior classification with self-organizing maps. In *RoboCup*, pages 108–118, 2000.

An Online POMDP Algorithm Used by the PoliceForce Agents in the RoboCupRescue Simulation

Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-draa

DAMAS Laboratory, Laval University
{spaquet, tobin, chaib}@damas.ift.ulaval.ca

Abstract. In the RoboCupRescue simulation, the *PoliceForce* agents have to decide which roads to clear to help other agents to navigate in the city. In this article, we present how we have modelled their environment as a POMDP and more importantly we present our new online POMDP algorithm enabling them to make good decisions in real-time during the simulation. Our algorithm is based on a look-ahead search to find the best action to execute at each cycle. We thus avoid the overwhelming complexity of computing a policy for each possible situation. To show the efficiency of our algorithm, we present some results on standard POMDPs and in the RoboCupRescue simulation environment.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) provide a very general model for sequential decision problems in a partially observable environment. A lot of problems can be modelled with POMDPs, but very few can be solved because of their computational complexity (POMDPs are PSPACE-complete [1]), which motivates the search for approximation methods. Recently, many approximation algorithms have been developed [2, 3, 4, 5, 6] and they all share in common the fact that they try to solve the problem offline. While these algorithms can achieve very good performances, they are still not applicable on large multiagent problems, like the RoboCupRescue simulation.

This paper presents a novel idea for POMDPs that, to our knowledge, have never received a lot of attention. The idea is to use an online approach based on a look-ahead search in the belief state space to find the best action to execute at each cycle. By doing so, we avoid the overwhelming complexity of computing a policy for every possible situation the agent could encounter. We present results showing that it is possible to achieve relatively good performances by using a very short amount of time online. To make our solution even better, we have incorporated a factored representation in order to speed up the computation time and reduce the amount of memory required.

In this article, we first describe the formalism of our RTBSS (Real-Time Belief Space Search) algorithm, followed by some results on standard POMDPs, then we present an adaptation of our method for the RoboCupRescue simulation environment and some results showing its efficiency.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 196–207, 2006.

2 POMDP

In this section we briefly describe POMDPs [7, 8] and then we introduce factored POMDPs. Formally, a POMDP is a tuple described as $\langle S, A, T, R, \Omega, O\rangle$ where:

- S is the set of all the environment states;
- A is the set of all possible actions;
- $T(s, a, s')$ is the probability of ending in state s' if the agent performs action a in state s;
- $R(s)$ is the reward associated with being in state s.
- Ω is the set of all possible observations;
- $O(s', a, o)$ is the probability of observing o if action a is performed and the resulting state is s'.

Since the environment is partially observable, an agent cannot perfectly distinguish in which state it is. To manage this uncertainty, an agent can maintain a belief state b which is defined as a probability distribution over S. $b(s)$ means the probability of being in state s according to belief state b.

The agent also needs to choose an action to do in function of its current belief state. This action is determined by the policy π, which is a function that maps a belief state to the action the agent should execute in this belief state. To construct a policy, the agent has to evaluate the expected reward of a belief state. To do so, the agent can use the following value function of a belief state for an horizon of t:

$$V_t(b) = R(b) + \gamma \max_a \sum_{o\in\Omega} P(o \,|\, b, a)\, V_{t-1}(\tau(b, a, o)) \tag{1}$$

$$R(b) = \sum_{s\in S} b(s)R(s) \tag{2}$$

$R(b)$ is the expected reward for the belief state b and the second part of equation 1 is the discounted expected future rewards. $P(o \,|\, b, a)$ is the probability of observing o if action a is performed in belief state b:

$$P(o \,|\, b, a) = \sum_{s'\in S} O(s', a, o) \sum_{s\in S} T(s, a, s')b(s) \tag{3}$$

Also, $\tau(b, a, o)$ is the belief state update function. It returns the resulting belief state if action a is done in belief state b and observation o is perceived. If $b' = \tau(b, a, o)$, then:

$$b'(s') = \eta O(s', a, o) \sum_{s\in S} T(s, a, s')b(s) \tag{4}$$

where η is a normalizing constant. Finally, the policy can be obtained according to:

$$\pi_t(b) = \operatorname*{argmax}_a \left[R(b) + \gamma \sum_{o\in\Omega} P(o \,|\, b, a)\, V_{t-1}(\tau(b, a, o)) \right] \tag{5}$$

$$b = \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.25 \\ 0.05 \\ 0.50 \\ 0 \\ 0.10 \\ 0.10 \end{bmatrix}, \begin{bmatrix} 0.10 \\ 0.30 \\ 0.15 \\ 0.45 \end{bmatrix} \right) \qquad b = \left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} P(X_2 = x_1 \wedge X_3 = x_1) \\ P(X_2 = x_1 \wedge X_3 = x_2) \\ \cdots \\ \cdots \\ \cdots \\ P(X_2 = x_6 \wedge X_3 = x_4) \end{bmatrix} \right)$$

Fig. 1. Belief states without dependance (left) and with dependance (right)

2.1 Factored POMDP

The traditional POMDP model is not suited to big environments because it requires explicitly enumerating all the states. However, most environments can be described by a set of different features which allows representing them much more compactly. Let $X = \{X_1, X_2, \ldots, X_M\}$ be the set of M random variables that fully describe a system state. We can then define a state by assigning a value to each variable: $s = \{X_1 = x_1, \ldots, X_M = x_M\}$ or more compactly $s = \{x_i\}_{i=1}^{M}$.

With such a factored representation it is then possible to compactly represent the transition and observation functions as a dynamic Bayesian network [9]. Also, if the state variables can be partitioned into probabilistically independent subsets, then the joint distribution representing the belief state can be compactly represented by the product of the marginal distributions of each subset [4]. Therefore, by maintaining the probabilities on variables instead of states, it is much easier to update the belief state and use it for approximation methods.

Figure 1 shows an example of two belief states. On the left, the variables are all independent, thus there is one vector of probabilities for each variable. On the right, the last two variables are dependent and thus there are only two vectors. Moreover, even if all variables are dependent, it is still possible to factorize the belief state with minimal degradation of the solution's quality. Some methods have been developed to automatically generate groups of variables that offer an interesting compromise between compactness of the representation and the performance of the agent [4, 10]. It is important to mention that the factorization is not necessary for our method, but if it can be used, it can accelerate the calculations, thus helping our algorithm to search deeper.

2.2 Formalization

More formally, to take advantage of the factored representation of the belief state, we define a function $\omega : B \rightarrow \mathbb{P}\, S$:

$$\omega(b) = \{\{x_i\}_{i=1}^{M} \mid (\forall\, x_i)\ P_b(X_i = x_i) > 0\} \tag{6}$$

This function returns, according to a belief state, all the states the agent could be in. We know that a state is impossible if one of the variables has a probability of zero according to b. Moreover, if the variables are ordered approximately according to their certainty (see Figure 1), this subset of states can be rapidly constructed because each time we encounter a variable with a probability equal to zero, we can immediately exclude all the corresponding states. The following equation can then be computed much more rapidly than equation 2:

$$R(b) = \sum_{s \in \omega(b)} R(s)b(s) \tag{7}$$

In fact, the less uncertainty the agent has, the smaller the subset of possible states is and the faster the computation of equation 7 is compared to equation 2.

Now that we consider only the states that an agent can be in, we would also like to have a function that returns the states that are reachable from a certain belief state. To do so, we define a new function $\alpha : A \times B \times \Omega \rightarrow \mathbb{P}\, S$ that takes as parameters the current belief state b, the action a that was performed and the observation perceived o and returns all the states that the agent can reach:

$$\alpha(a, b, o) = \{s' \mid (\forall\, s \in \omega(b))\;\; T(s, a, s') \neq 0 \wedge O(s', a, o) \neq 0\} \tag{8}$$

The probability of making an observation $(P(o|a, b))$ can also be expressed using α and ω:

$$P(o \mid a, b) = \sum_{s' \in \alpha(a,b,o)} O(s', a, o) \sum_{s \in \omega(b)} T(s, a, s')b(s). \tag{9}$$

3 Online Decision Making

Instead of computing a policy offline, we adopted an online approach where the agent rather performs a local search at each step in the environment, thus avoiding a lot of computations. The advantage of such a method is that it can be applied to problems with a huge state space.

3.1 Belief State Value Approximation

In section 2, we described how it was possible to exactly compute the value of a belief state (equation 1). In this section, we instead explain how we estimate the value of a belief state for our online approach by using a look-ahead search. The main idea is to construct a tree where the nodes are belief states and where the branches are a combination of actions and observations (see Figure 2). To do so, we have defined a new function $\delta : B \times \mathbb{N} \rightarrow \mathbb{R}$ that looks like equation 1, but that is based on a depth-first search instead of dynamic programming. The function takes as parameters a belief state b and a remaining depth d and returns an estimation of the value of b by performing a search of depth d. For the first call, d is initialized at D, the maximum depth allowed for the search.

$$\delta(b, d) = \begin{cases} U(b) & \text{, if } d = 0 \\ R(b) + \gamma \max\limits_{a} \sum\limits_{o \in \Omega} (P(o \mid b, a) \times \delta(\tau(b, a, o), d-1)) & \text{, if } d > 0 \end{cases} \tag{10}$$

where $R(b)$ is computed using equation 7 and $P(o\,|\,b, a)$ using equation 9.

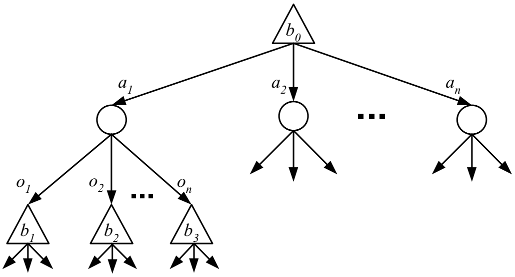


Fig. 2. A search tree

When $d = 0$, we are at the bottom of the search tree. In this situation, we need a way to estimate the value of the current belief state. To do so, we need a utility function $U(b)$, that gives an estimation of the real value of this belief state (if the function $U(b)$ was perfect, their would be no need for a search). If it is not possible to find a better utility function, we can use $U(b) = R(b)$.

When $d > 0$, the value of a belief state at a depth of $D - d$ is simply the immediate reward for being in this belief state added to the maximum discounted reward of the subtrees underneath this belief state.

Finally, the agent's policy which returns the action the agent should do in a certain belief state is defined as:

$$\pi(b, D) = \operatorname*{argmax}_a \sum_{o \in \Omega} P(o \mid b, a)\delta(\tau(b, a, o), D - 1). \tag{11}$$

3.2 RTBSS Algorithm

We now describe our RTBSS (Real-Time Belief Space Search) algorithm that is used to construct the search tree and to find the best action. Since it is an online algorithm, it must be applied each time the agent has to make a decision.

To speed up the search, our algorithm uses a "Branch and Bound" strategy to cut some sub-trees. The algorithm first explores a path in the tree up to the desired depth D and then computes the value for this path. This value then becomes a lower bound on the maximal expected value.

Afterwards, for each node of the tree visited, the algorithm can evaluate with an heuristic function if it is possible to improve the lower bound by pursuing the search. This is represented by the Prune function at line 10. The heuristic function returns an estimation of the best utility value that could be found if the search was pursued. Thus, if according to the heuristic, it is impossible to find a value that is better than the lower bound by continuing the search, the algorithm backtracks and explores another action. If not, the search continues because there is a non-zero probability that the best solution hides somewhere in the sub-tree.

Moreover, the heuristic function used in the Prune function must be defined for each problem. Also, to work well it has to always overestimate the true value.

Algorithm 1. The RTBSS algorithm

```
 1: **Function** RTBSS($b$, $d$ , $rAcc$)
    **Inputs:** $b$: The current belief state.
                $d$: The current depth.
                $rAcc$: Accumulated rewards.
    **Statics:** $D$: The maximal depth search.
                 $bestValue$: The best value found in the search.
                 $action$: The best action.
 2: if $d = 0$ then
 3:    $finalValue \leftarrow rAcc + \gamma^D \times U(b)$
 4:    if $finalValue > bestValue$ then
 5:       $bestValue \leftarrow finalValue$
 6:    end if
 7:    return $finalValue$
 8: end if
 9: $rAcc \leftarrow rAcc + \gamma^{D-d} \times R(b)$
10: if PRUNE($rAcc$, $d$) then
11:    return $-\infty$
12: end if
13: $actionList \leftarrow$ SORT($b$, $A$)
14: $max \leftarrow -\infty$
15: for all $a \in actionList$ do
16:    $expReward \leftarrow 0$
17:    for all $o \in \Omega$ do
18:       $b' \leftarrow \tau(b, a, o)$
19:       $expReward \leftarrow expReward + \gamma^{D-d} \times P(o \mid a, b) \times$ RTBSS($b', d-1, rAcc$)
20:    end for
21:    if $(d = D \wedge expReward > max)$ then
22:       $max \leftarrow expReward$
23:       $action \leftarrow a$
24:    end if
25: end for
26: return $max$
```

If it does not, it would have the effect of pruning some parts of the tree that might hide the best solution. On the other hand, if the heuristic always overestimates, we are guaranteed that all the pruned sub-trees do not contain the best solution.

To link the algorithm with the equations presented, notice that the line 19 of Algorithm 1 corresponds to the last part of equation 10, where δ is replaced by RTBSS. Also, the function $\tau(b, a, o)$ at line 18 returns the new belief state if o is perceived after the agent has done action a in belief state b. Note that the actions are sorted at line 13. The purpose of this is to try the actions that are the most promising first because it generates more pruning early in the search.

With RTBSS the agent finds at each turn the action that has the maximal expected value up to a certain horizon of D. As a matter of fact, the performance of the algorithm strongly depends on the depth of the search. The complexity of our algorithm is in the worst case of: $O((|A| \times |\Omega|)^D)$. This is if no pruning is possible, thus with a good heuristic, it is possible to do much better. As we can see, the complexity of our algorithm depends on the number of actions and

Table 1. Comparison of our approach

Problem	Reward	Time (s)
Tag (870s,5a,30o)		
Q_{MDP}	-16.75	11.8
RTBSS	-10.56	0.23[1]
PBVI [3]	-9.18	180880
BBSLS [2]	∽ -8.3	∽100000
BPI [4]	-6.65	250
HSVI [5]	-6.37	10113
Perséus [6]	-6.17	1670
RockSample[4,4] (257s,9a,2o)		
RTBSS	16.2	0.1[1]
PBVI [5][2]	17.1	∼ 2000
HSVI [5]	18.0	577
RockSample[5,5] (801s,10a,2o)		
RTBSS	18.7	0.1[1]
HSVI [5]	19.0	10208
RockSample[5,7] (3201s,12a,2o)		
RTBSS	22.6	0.1[1]
HSVI [5]	23.1	10263
RockSample[7,8] (12545s,13a,2o)		
RTBSS	20.1	0.2[1]
HSVI [5]	15.1	10266

observations, but not on the number of states. Therefore, even if the environment has a huge state space, our algorithm would still be applicable, if the number of actions and observations are kept small.

4 Experiments on Standard POMDPs

This section presents the results obtained in two problems: *Tag* [3] and *RockSample* [5]. If we compare RTBSS with different existing approaches (see Table 1), we see that our algorithm can be executed much faster than all the other approaches. Our algorithm does not require any time offline and takes only a few tenths of a second at each turn. On small problems the performance is not as good as the best algorithms but the difference is generally not too important.

Moreover, the most interesting results are obtained when the problem becomes bigger. If we look at the RockSample problems, RTBSS is really close on the first three smaller problems, but it is much better on the biggest instance of the problem. RTBSS is better because HSVI has not had time to converge. This shows the advantage of our approach on large environments.

[1] It corresponds to the average time taken by the algorithm at each time it is called in a simulation.

[2] PBVI was presented in [3], but the result on *RockSample* was published in [5].

Another huge advantage of our algorithm is that if the environment changes, we do not need to recompute a policy. Let's suppose that we have the RockSample problem but at each new simulation, the initial position of the rocks changes. With offline algorithms, it would require recomputing a new policy for the new configuration while our algorithm could be applied right away.

5 Experiments on RoboCupRescue

The RoboCupRescue simulation environment consists of a simulation of an earthquake happening in a city [11]. The goal of the agents (representing firefighters, policemen and ambulance teams) is to minimize the damages caused by a big earthquake, such as civilians buried, buildings on fire and roads blocked.

For this article, we consider only the policeman agents. Their task is to clear the most important roads as fast as possible, which is crucial to allow the other rescuing agents to perform their tasks. However, it is not easy to determine how the policemen should move in the city because they do not have a lot of information. They have to decide which road to prioritize and they have to coordinate themselves so that they do not try to clear the same road.

In this section, we present how we applied our approach in the RoboCupRescue simulation. In fact, we are interested in only a subproblem which can be formulated as: Having a partial knowledge of the roads that are blocked or not, the buildings in fire and the position of other agents, which sequence of actions should a policeman agent perform?

5.1 RoboCupRescue Viewed as a POMDP

We present how we modelled the problem of the RoboCupRescue as a POMDP, from the point of view of a policeman agent. The different actions an agent can do are: *North*, *South*, *East*, *West* and *Clear*. A state of the system can be described by approximately 1500 random variables, depending on the simulation.

- *Roads*: There are approximately 800 roads in a simulation and each road can either be blocked or cleared.
- *Buildings*: There are approximately 700 buildings in a simulation. We consider that a building can be on fire or not.
- *Agents position*: An agent can be on any of the 800 roads and there's usually 30-40 agents.

If we estimate the number of states, we obtain $2^{800} \times 2^{700} \times 800^{30}$ states. However, a strong majority of them are not possible and will not ever be reached. The state space of RoboCupRescue is too important to even consider applying offline algorithms. We must therefore adopt an online method that allows finding a good solution very quickly.

5.2 Application of RTBSS on RoboCupRescue

This section presents how we have applied RTBSS to this complex environment. In RoboCupRescue, the online search in the belief state space represents a search in the possible paths that an agent can take. In the tree, the probability to go from one belief state to another depends on the probability that the road used is blocked. One specificity of this problem is that we have to return a path to the simulator, thus the RTBSS algorithm has been modified to return the best branch of the tree instead of only the first action.

Furthermore, the key aspect of our approach is that we consider many variables of the environment to be static during the search in order to minimize the number of states considered. In the RoboCupRescue, all variables are considered static except the position of the agent and the variables about the roads. For the other variables, like the position of the other agents and the position of the fires, the agent considers that they keep the last value observed. Consequently, all those fixed variables are represented in the belief state by a vector containing only zeros except for the last value observed which has a probability of one. Therefore, the function ω (equation 6) only returns a small subset of states.

More precisely, the beliefs are only maintained for the road variables. Those variables are the most important for the agent decisions. In other words, the agent focuses on the more important variables, maintaining beliefs as precisely as possible, and it abstracts the other variables by considering that they are fixed and it relies only on its observations to maintain them.

We update the value of the fixed variables only when the agent perceives a new value. In our model, we consider the observations to be both the direct agent's observations and the information received by messages. We are in a cooperative multiagent system, therefore all agents have complete confidence in the information received from the other agents.

In complex dynamic multiagent environments, it is often better to rely on observations than to try to predict everything. There are just too many things moving in the simulation. Therefore, the agent should focus on the more important parts of the environment. To efficiently take all the unpredicted parts of the environment into consideration, the agent can shorten its loop of observation and action to keep its belief state up-to-date. This can be done because our RTBSS algorithm can find an action very quickly. Consequently, the agent makes frequent observations, thus it does not need a model for the less important parts of the world, because they do not have time to move a lot between observations.

Moreover, we have defined a dynamic reward function that gives a reward for clearing a road based on the positions of the fires and the other agents. This enables the agent to efficiently compute its estimated rewards based on its belief state without having to explicitly store all rewards for all possible states.

A policeman agent needs to assign a reward to each road in the city, which are represented as nodes in a graph (see Figure 3). The reward values change in time based on the positions of the agents and the fires, therefore the agent needs to recalculate them at each turn. To calculate the reward values, the agent propagates rewards over the graph, starting from the rewarding roads, which are the positions of the agents and the fires. For example, if a firefighter agent is on road r_1 then this road would receive a reward of 5, the roads adjacent to r_1 in

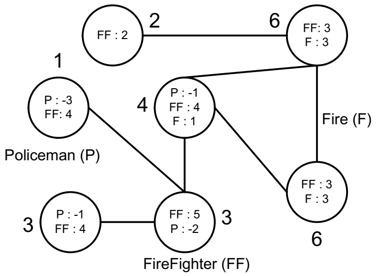


Fig. 3. Reward function's graph

the graph would receive a reward of 4, and so on. Also, we add rewards for all roads in a certain perimeter around a fire.

What is interesting with this reward function is that it can be used to coordinate the policeman agents. The coordination is necessary, because we do not want all agents to go to the same road. To do so, the agent propagates negative rewards around the other policeman agents, thus they all repulse each other. With this simple modification of the reward function, we were able to disperse efficiently, thus dynamically coordinate up to fifteen agents acting in a really dynamic environment.

Figure 3 shows an example of a reward graph. The nodes represent the roads and the reward source is identified in each node. The big number over a node is the total reward, which is the sum of all rewards identified in the node. As we can see, roads around the firefighter agent receive positive rewards, while roads around the policeman agent receive negative rewards. Therefore, the agent would want to go to roads near the fire and not necessarily go to help the firefighter because there is already a policeman agent near it. Consequently, agents are coordinating themselves simply by propagating negative rewards.

5.3 Results and Discussion

In such a huge problem as RoboCupRescue, it was impossible to compare our approach with other POMDP algorithms. Therefore, we compared our algorithm RTBSS with an heuristic method for the policeman agents. We have compared it with our last approach for the policemen in the RoboCupRescue simulation. In this approach agents were clearing roads according to some priorities. Each agent received a sector for which it was responsible at the beginning of the simulation. Afterwards, agents were clearing roads in this order: roads asked by the other agents, roads around refuges and fires and finally, all the roads in their sector.

The results that we have obtained on 7 different maps are presented in Figure 4. The approach presented in this paper improved the average score by 11 points. This difference is very important because in competitions, a few tenths of a point can make a big difference. On the graph, we show a 95% confidence interval that suggest that our algorithm allows more stable performances.

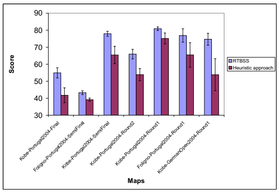


Fig. 4. Scores

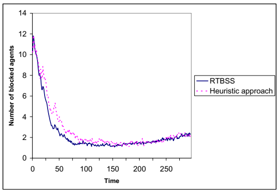


Fig. 5. Agents blocked

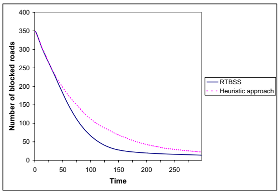


Fig. 6. Roads blocked

Figure 5 shows a comparison of the number of agents that are blocked at each cycle. The results show that our method allows prioritizing the most important roads since on average, there are one or two fewer blocked agents. Furthermore, Figure 6 shows the number of roads that are blocked at each cycle in the simulation. We see that RTBSS allows the policeman agents to clear the roads faster.

Briefly, with RTBSS, agents clear the most important roads faster than with the heuristic approach. Another result showing the efficiency of our approach is that it helped us to finish second at the 2004 international competition.

6 Related Work

The first works that come to mind are those approximation methods that aim to solve larger problems. Particularly, we compared our approach with 5 other algorithms [2, 3, 4, 5, 6] used to find an approximate solution. These approaches are very interesting because they achieve good solution quality. In addition, some of these algorithms can bound their distance from the optimal solution. However, they can only be applied to relatively small environments because they all proceed offline and require a lot of computation time on big problems.

For POMDPs, very few researchers have explored the possibilities of online algorithms. [12] used a real-time dynamic programming approach to learn a belief state estimation by successive trials in the environment. The main differences are that they do not search in the belief state tree and they need offline time to calculate their starting heuristic based on the Q_{MDP} approach.

7 Conclusion and Future Work

This paper presents RTBSS, an online POMDP algorithm useful for large, dynamic and uncertain environments. The main advantage of such a method is that it can be applied to problems with huge state spaces where other algorithms would take way too much time to find a solution. To reinforced our claims, we showed results on two problems considered to be large in the POMDP literature. Those results show that RTBSS becomes better as the environment becomes bigger, compared to state of the art POMDP approximation algorithms.

We have also applied our algorithm in a complex multiagent environment, the RoboCupRescue simulation. We showed how we have slightly modified our basic RTBSS algorithm to take the specificity of multiagent systems more into consideration, and we presented results showing its efficiency.

In our future work, we would like to improve our online algorithm by reusing the information computed previously in the simulation. We also want to combine offline and online strategies, because it might be possible to precompute some information offline that could be useful online. In short, RTBSS represents a good step towards making POMDP applicable in real life applications.

References

1. Papadimitriou, C., Tsisiklis, J.N.: The complexity of markov decision processes. Mathematics of Operations Research **12** (1987) 441–450
2. Braziunas, D., Boutilier, C.: Stochastic local search for POMDP controllers. In: The Nineteenth National Conference on Artificial Intelligence (AAAI-04). (2004)
3. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03), Acapulco, Mexico (2003) 1025–1032
4. Poupart, P.: Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes. PhD thesis, University of Toronto (2005) (to appear).
5. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence(UAI-04), Banff, Canada (2004)
6. Spaan, M.T.J., Vlassis, N.: A point-based POMDP algorithm for robot planning. In: In Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, Louisiana (2004) 2399–2404
7. Aberdeen, D.: A (revised) survey of approximate methods for solving partially observable markov decision processes. Technical report, National ICT Australia (2003)
8. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Technical Report CS-96-08, Brown University (1996)
9. Boutilier, C., Poole, D.: Computing optimal policies for partially observable decision processes using compact representations. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), Portland, Oregon, USA, AAAI Press / The MIT Press (1996) 1168–1175
10. Boyen, X., Koller, D.: Tractable inference for complex stochastic processes. In: In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. (1998) 33–42
11. Kitano, H.: Robocup rescue: A grand challenge for multi-agent systems. In: Proceedings ICMAS 2000, Boston (2000)
12. Geffner, H., Bonet, B.: Solving large POMDPs using real time dynamic programming (1998)

Cooperative Action Control Based on Evaluating Objective Achievements

Hikari Fujii[1], Masayuki Kato[1], and Kazuo Yoshida[2]

[1] School of Science for Open Environmental Systems, Keio University
http://www.yoshida.sd.keio.ac.jp/~robocup/
[2] Department of System Design Engineering, Keio University
fujii@yoshida.sd.keio.ac.jp

Abstract. This paper deals with a cooperative control method for a multi-agent system in dynamic environment. This method enables a robot to perform flexible cooperation based on the global evaluation of the achievement of objectives. Each robot communicates qualitative evaluation on the achievement level of each objective. Each robot calculates the global evaluation on the achievement of the team objective from the individual evaluation. The evaluation on the objective achievement is abstracted in order to reduce the influence of variation of the evaluation value and the communication load. As an example, the method is applied to the EIGEN team robots for the Middle Size League of RoboCup, since it is necessary for the soccer robots to cooperate each other in dynamic environment. Its effectiveness was demonstrated through the RoboCup 2004 competition.

1 Introduction

It is important for a multi-agent system to act in a cooperative manner. The cooperation in a multi-agent system is able to improve the efficiency in achieving a task and in executing some tasks that are difficult to be accomplished by one agent. Therefore, many researchers have studied about cooperative action of multi-agent systems [1] and [2]. Nowadays, it has been expected to realize robots that are symbiotic with human in open environment. These robots are required to act cooperatively with human, other robots and artifacts in complicated environment. In the dynamic environment, there are many unexpected happenings. In order to make robots act appropriately by considering various situations, many rules become necessary for the cooperative control. Further, if the configuration changes because of some unexpected events, robots can not adapt to the situation. Therefore, in [3] authors have studied a cooperative action control method based on the evaluation of the objective achievement of a robot system. To realize such action, it is necessary to develop a more flexible cooperative control method.

RoboCup Middle Size League is a soccer game executed by autonomous mobile robots. Robots are forbidden to use a global sensor. Since there are many robots in a field, it is difficult to construct a global model and the wireless LAN used in communication is not stable. In the RoboCup Middle Size League, a cooperation capability is

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 208 – 218, 2006.

required to flexibly adapt a robot team to work in a dynamic environment, which is a good test bed for a multi-agent system. Cooperative behavior is one of the important research topics in the RoboCup Middle Size League and many researchers have studied about it. There are representative examples of Dynamic Task Assignment [4-6], Dynamic Role Assignment [7-11] and so on. Dynamic Role Assignment is realized for the efficient cooperation among the robots. It scope is to assign a Role to avoid potential conflicts among robots.

In particular, for example, the method described in [9] needs accurate data about the agent position, the number of the robots for each role is fixed and the roles are exchange among robots. Instead, in our method, the number of the robot for each role is not fixed strictly, and the role is decided according to the objective achievement of the whole system. With this method, the organization of the team is changed dynamically according to the situation.

In our approach, a flexible selection of the objectives is realized by using the method of the qualitative information on robot own achievement level of the objective. The evaluation of achievement level of the objective among robots team is calculated based on the sum of the respective self-evaluations of each robot. This method enables robots to change appropriately the multi-agent system behavior keeping the high autonomy of each agent. Each robot is able to assume the same role before achieving the desired state to accomplish the global objective. Only one finally selects the offensive role, while supporting and defensive roles can be assumed by more than one robot. Robots are also able to cooperate without sharing accurate position data and the same action modules introduce in session 3.1. The abstraction introduce in session 2.1 of the evaluation on the objective achievement is effective to reduce the influence of variation of evaluation value and the communication load.

In this paper, the method which has first been proposed in [3] is further developed to apply it to a robots system with heterogeneous robots, in order to cope with the increase of robot number. In this case, the robots with omni directional drive are used as a field player and the robots with differential drive are used as a goalkeeper and as a defender to better exploit the different characteristics of robots hardware. The usefulness of the cooperative control method is investigated in the RoboCup Middle Size League and the effectiveness is examined thorough RoboCup 2004.

2 Cooperative Control Method

2.1 Concept

Our research aims at establishing a cooperative control for mobile robots in dynamic environments where autonomous robots need many rules to achieve a task because of a lot of different situations to be considerate. Further, robots often fail to achieve a task because of unexpected happenings. In order to solve this problem, this study takes into account the achievement level of the objective. The objective is defined according to the essential requirements for achieving a task. In this study, the cooperative behavior among agents is realized by performing an evaluation of the degree of achieving an objective and by sharing this evaluation result among all the agents.

According to this method, each agent possesses the information on the objective of the multi-agent system, that is, the global objective. They can estimate the current state from the environmental information. With this information, each agent calculates two kinds of evaluation information. The first is its own degree of achieving an objective, which is called *Self-evaluation*, and the second the contribution degree of an agent to the achievement of the global objective, which is called *Agent Satisfaction*. In this study, the function for calculating *Self-evaluation* is empirically defined. Each evaluation value takes a simple integer. Each agent knows the desired state for achieving the objective which is called *System Objective*. This value takes also an integer. These evaluation values can be considered as qualitative data abstraction of the degree of achieving an objective.

The quantitative information like sensory data and the physical quantities have been directly used in many studies when an agent needs the information about other agents in order to cooperate each other. However, some studies show that a method using qualitative information might be more useful than the method using quantitative information in open environment which varies from moment to moment. The abstraction reduces the influence of variation of the evaluation value. Further, the cooperation among heterogeneous agents is achieved through the individual evaluation functions if they are abstracted in the same scale.

To calculate the *Agent Satisfaction*, each agent compares the value of its *Self-evaluation* with the one of other robots, and selects those values which are higher value than its own value in order to sum up all of them. With this operation, each agent has a different value of *Agent Satisfaction* according to its state. In the case that an agent satisfies the global objective, the other agent is inhibited to achieve the objective according to the high value of *Agent Satisfaction*.

In some situation, a priority is given among agents. In order to realize the effective cooperation, the priority agent should be selected according to the agent role or hardware characteristics. Special consideration is paid to the evaluation of the priority agent. Since this kind of agent could have been weighted by either a high or a low evaluation, then it can greatly influence the evaluation of the whole system. The priority agent induces an action according to the own evaluation. The evaluation of the priority agent is always considered by the other agents when they calculate the *Agent Satisfaction*.

The outline of the proposed method is as follows.

STEP1: Each agent evaluates its state with regards to all its respective objectives, and this evaluated information is shared among all agents.
STEP2: Each agent calculates the *System Satisfaction* based on the sum of the evaluation done by agents which have an evaluation higher than its own and of the evaluation done by the priority agents.
STEP3: The agent selects the action according to the Agent Satisfaction and the *System Objective*.

The concept and the flow of this method are shown in Fig.1.

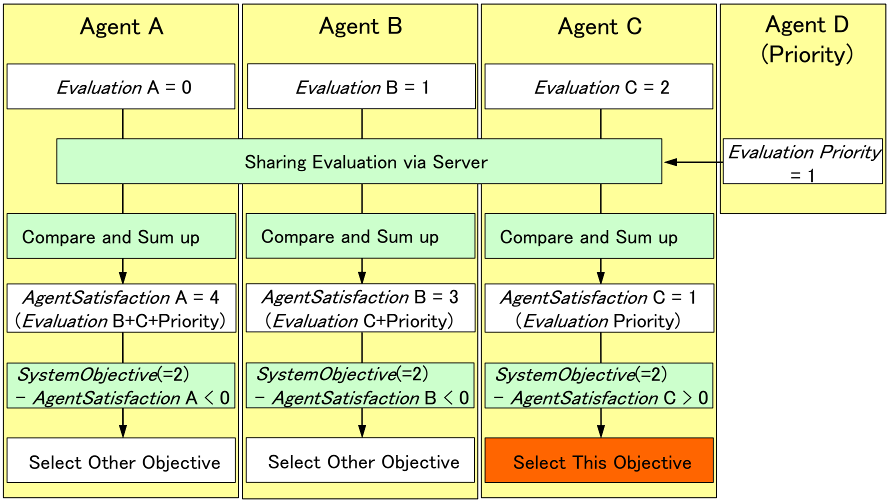

Agent A
Agent B
Agent C
Agent D (Priority)
Evaluation A = 0
Evaluation B = 1
Evaluation C = 2
Evaluation Priority = 1
Sharing Evaluation via Server
Compare and Sum up
Compare and Sum up
Compare and Sum up
AgentSatisfaction A = 4 (*Evaluation* B+C+Priority)
AgentSatisfaction B = 3 (*Evaluation* C+Priority)
AgentSatisfaction C = 1 (*Evaluation* Priority)
SystemObjective(=2) − *AgentSatisfaction* A < 0
SystemObjective(=2) − *AgentSatisfaction* B < 0
SystemObjective(=2) − *AgentSatisfaction* C > 0
Select Other Objective
Select Other Objective
Select This Objective

Fig. 1. The concept and the flow of the proposed method

2.2 Formulation

According to the above-mentioned concept, the formulas for the proposed method are defined.

These variables are defined as follows:

- *System Objective* $_i$ Desired state of the Objective (i). Each agent has the same value.
- *Agent Satisfaction* $_i$ Satisfaction and evaluation index of the Objective (i) through the position of *Agent* ^{own}. Each agent has a different value.
- $Agent^{k}{}_{priority}$ Priority agent.
- $Evaluate_{i}(Agent^{j})$ Value of *Self-evaluation* about Objective (i) from $Agent_j$.
- $E_i\,(Agent_j)$ Value of the evaluation about the Objective (i) from $Agent_j$ considered by $Agent_{own}$
- m Number of the priority agent
- n Number of the non-priority agent

Agents always take into account the evaluation of the priority agent. The evaluation of Objective (i) is the sum of these evaluations. The formulation of the evaluation of Objective (i) is given by the following equation:

$$AgentSatisfaction_i = \sum_{k=1}^{m} Evaluate_i\left(Agent^{k}{}_{priority}\right) + \sum_{j=1}^{n} E_i\left(Agent_j\right) \tag{1}$$

where

$$E_i\left(Agent_j\right)=\begin{cases} Evaluate_i\left(Agent_j\right) & (Evaluate_i\left(Agent_j\right)>Evaluate_i\left(Agent_{own}\right)) \\ 0 & (Evaluate_i\left(Agent_j\right)\le Evaluate_i\left(Agent_{own}\right)) \end{cases}$$

$$v_i = SystemObjective_i - AgentSatisfaction_i \tag{2}$$

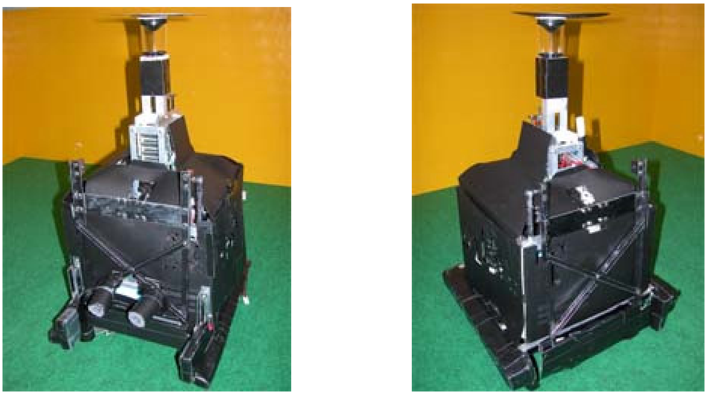
When the vi is negative or equal to 0, the agent recognizes that the Objective (i) has been achieved and inhibits the action for achieving Objective (i). When the v_i is positive, the agent recognizes that the Objective (i) has not been achieved and selects some suitable actions for achieving Objective (i). As a result, the agents behave cooperatively.

The above-mentioned method is for a single objective problem. This method can be applied to a multi-objective system by evaluating the multiple objectives of the system. In this paper, the objects have priority depending on the hardware. The objective which has highest priority is checked first. The details of the method applied to the real robots are described below.

3 Applications to the RoboCup Middle Size League

3.1 Overview

The cooperation method described in chapter 2 has been applied to RoboCup Middle Size robots. In 2004, the number of robots was increased and the team was built with robots which have different kinds of hardware as shown in Fig. 2.



(a)Field player robot with omni- directional drive

(b)Goalkeeper and Defender robot with omni-directional drive

Fig. 2. EIGEN team's robot

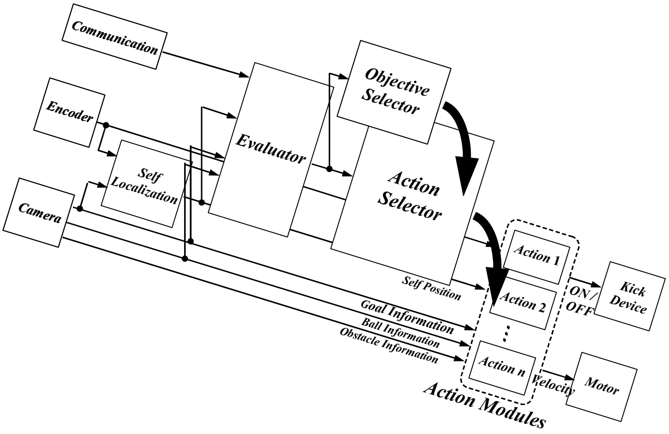


Fig. 3. The flow of the action control

In order to take into account the characteristic of the drives, the robots with omnidirectional drives are used as field players and the robots with differential drives are used as a goalkeeper and as a defender.

The flow of the action selection induced by the cooperation method is shown in Fig 3. An "objective selector", an "action selector" and their "action modules" are imple-mented in each robot. The objective selector selects the objective according to the cooperation method which is described in chapter 2. The action selector selects an ac-tion module according to the selected objective and the information on the environ-ment. An action module is designed in the form of an action element for achieving a midterm objective, such as "Go to ball", "Defense" and so on. The agent achieves the midterm objectives by selecting the action modules continuously. The agent recog-nizes the ball and the goals, and calculates its own position and orientation from the information about the landmarks.

The robots communicate each other and share the information on the ball, their position and the evaluation of the objective achievement via the Wireless LAN. The protocol of the communication is UDP. Therefore, the information is not always received. If the evaluation information of other agents is not acquired for a long time because of the troubles on the Wireless LAN, the agents select the objective with the higher priority which is decided according to their characteristic.

The evaluation function of the cooperation method is described below in detail.

3.2 Objective and Self-evaluation

Three kinds of the objectives for soccer playing robots are defined; "Offense", "Defense" and "Support". To keep a ball is one of basic actions of the soccer playing robots. However, it is not necessary for all robots to approach a ball. It is required for one member of the team to keep a ball. At the same time, defensive and support actions of the other robots are also important. Therefore, three kinds of objectives are defined in our method. The field player can select all of these objectives. The goalkeeper and the defender can select the "Defense" and "Support". The basic action for each objective is shown in Fig 4.

The basic action of the offense is carrying a ball to the goal. In order to achieve this, the robot has a high evaluation value when the robot is near to the ball and robot, ball and goal belong to a same line. The evaluation of support objective is calculated according to the distance between the actual robot position and the position needed to support other robots. The evaluation of defense objective is calculated according to the position of the robot. The goalkeeper and the defender have a high defense evaluation value when the robot is staying between the ball and the goal. The field player has a high defense evaluation value when the robot is near the position to a defensive position.

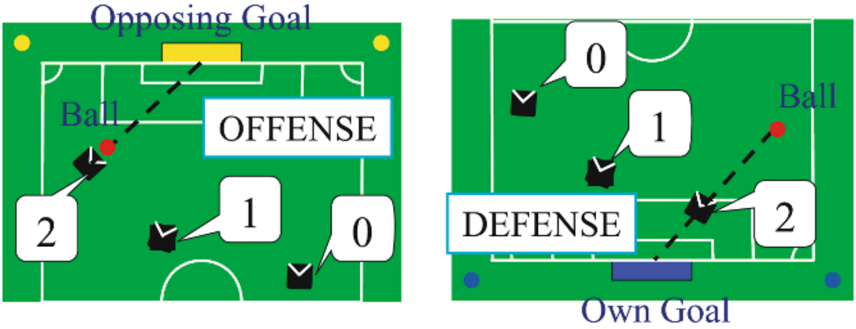


Fig. 4. The concept figures of the main action for each objective and evaluation value

3.3 Calculation of the Agent Satisfaction

The goalkeeper and the defender are defined as priority agents. The calculating method of *Agent Satisfaction* is described in this session.

The goalkeeper has higher priority than the defender. So, the *Agent Satisfaction* of the goalkeeper is formulated as follows:

$$AgentSatisfaction_i = E_i\left(Agent^{DF}\right) \tag{3}$$

where

$$E_i\left(Agent^{DF}\right) = \begin{cases} Evaluation_i\left(Agent^{DF}\right) & \left(Evaluation_i\left(Agent^{DF}\right) > Evaluation_i\left(Agent_{own}\right)\right) \\ 0 & \left(Evaluation_i\left(Agent^{DF}\right) \leq Evaluation_i\left(Agent_{own}\right)\right) \end{cases}$$

According to this formula, the goalkeeper takes into account the evaluation of the defender to determine the objectives if the defender has higher value on the evaluation. The *Agent Satisfaction* of the defender is formulated as follows:

$$AgentSatisfaction_i = \sum_{j=1}^{4} E_i\left(Agent_j\right) \tag{4}$$

where

$$E_i\left(Agent_j\right)=\begin{cases} Evaluation_i\left(Agent_j\right) & \left(Evaluation_i\left(Agent_j\right) > Evaluation_i\left(Agent_{own}\right)\right) \\ 0 & \left(Evaluation_i\left(Agent_j\right) \le Evaluation_i\left(Agent_{own}\right)\right) \end{cases}$$

According to this formula, the defender takes into account the evaluation of the other agents according if it is higher than the defender's evaluation. The *Agent Satisfaction* of the defender is formulated as follows:

$$AgentSatisfaction_i = \sum_{k=1}^{2} Evaluation_i\left(Agent_k^{GK/DF}\right) + \sum_{j=1}^{3} E_i\left(Agent_j\right) \tag{5}$$

where

$$E_i\left(Agent_j\right)=\begin{cases} Evaluation_i\left(Agent_j\right) & \left(Evaluation_i\left(Agent_j\right) > Evaluation_i\left(Agent_{own}\right)\right) \\ 0 & \left(Evaluation_i\left(Agent_j\right) \le Evaluation_i\left(Agent_{own}\right)\right) \end{cases}$$

The field players are not priority agents, so that the formulas for calculating the *Agent Satisfaction* are almost same as the basic formula shown in chapter 2. These relation ship is shown in Fig.5.

The cooperation method among robots with these evaluation functions are examined in chapter 4.

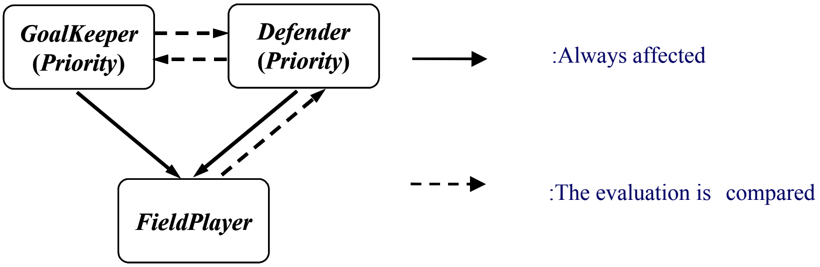


Fig. 5. The relationships among the agents

4 Experiments

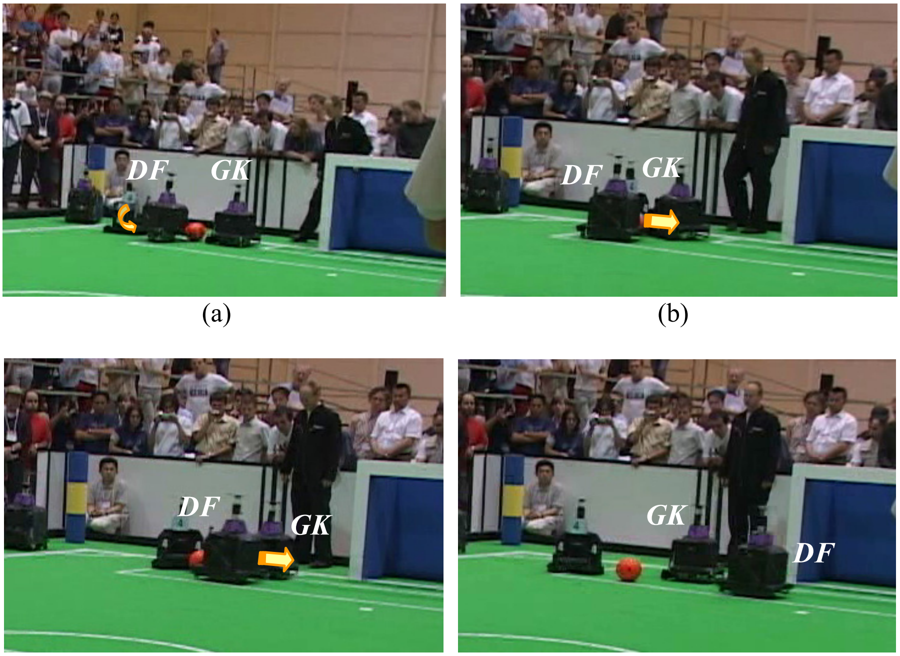
4.1 Experimental Situation

The proposed method has been applied to the real robots for RoboCup Middle Size League in RoboCup 2004. In that competition, EIGEN team had 5 robots; 3 field players, 1 defender and 1 goalkeeper. The field players have an omni-directional drive. The defender and the goalkeeper have a differential drive. These robots have an omni-directional camera and encoders as the sensor. They have a Celeron 2 GHz CPU, an image processing board and a kicking device with a solenoid. They have wireless LAN for communicating each other.

4.2 The Results of the Game with the Cooperation Method Between the Goalkeeper and the Defender

The scene from the finals in RoboCup 2004 is shown in Fig 6. In these figures, the FP means the field player, the GK means the goalkeeper and the DF means the defender. The arrows mean the direction toward the robots move.

At first, the goalkeeper and the defender saved the goal shown in (a). Next, the defender was moving to go away from the goalkeeper in (b), (c). After that, the defender stayed at the support position as shown in (d). Defender's actions prevented opponents to disturb goalkeeper's protection action. In this case, the goalkeeper estimated that it could protect the goal. So, the defender's evaluation about goalkeeper's defensive capability was high. Therefore, the defender went away from the ball and moved to the support position. According to these results, the effectiveness of the cooperative method applied to the goalkeeper and the defender has been proved.



(a) (b)

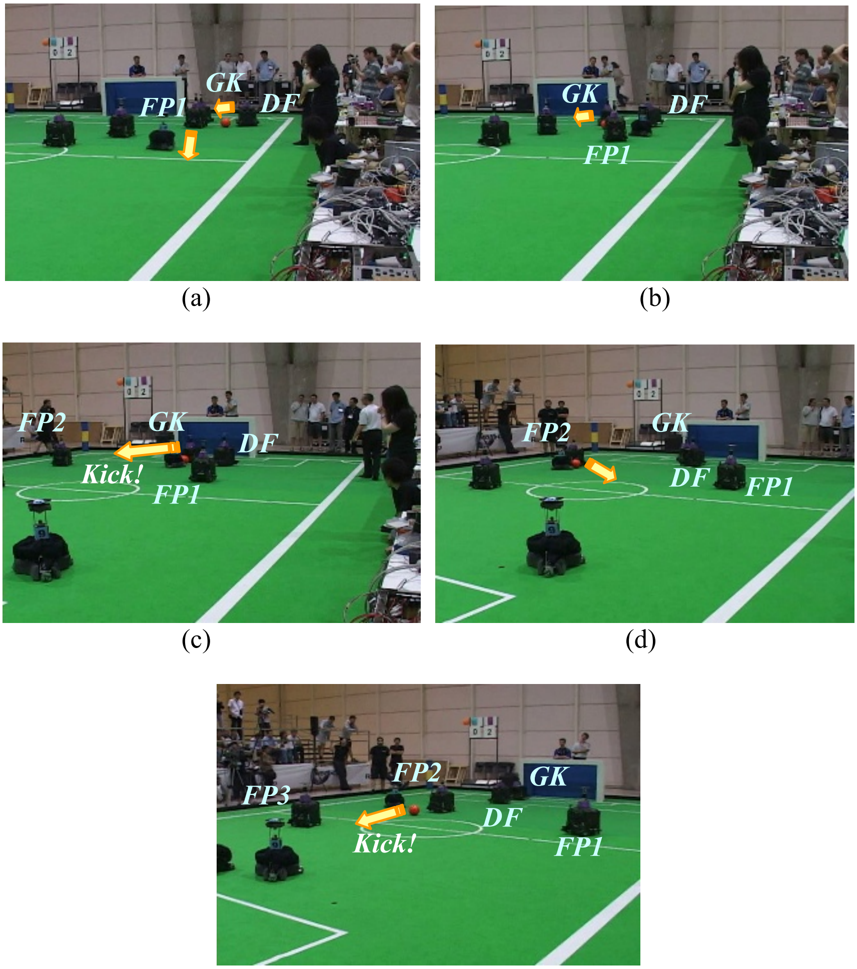
(c) (d)

Fig. 6. The scene of the cooperation between the goalkeeper and the defender

4.3 The Result of the Cooperation Among Robots

A scene from the semi-finals is show in Fig.7. In these figures, the counter attack was shown. First, the goalkeeper and the defender protected the goal and the field player was moving to go away form the ball as shown in (a). In this figure, the field player tried to prevent the other robots from disturbing the protecting action of both goalkeeper and defender. In this case, the goalkeeper and the defender estimated that they can protect the goal, therefore the approaching action of the field player was inhibited. In the next figure (b), the goalkeeper protected the goal. Then, the goalkeeper cleared the ball for the field players as shown in Fig. (c). At this time, the field player waited at the support position according to the selected objective. After that, the field player kept the ball and kicked it forward as shown in Figs. (d) and (e).

The cooperative actions illustrated in the sections 4.2 and 4.3 took place during the competition. As a result of using this approach, the EIGEN team won the first prize in RoboCup 2004 MSL competition. The effectiveness of our cooperation method was proved through this performance.



(a) (b)

(c) (d)

(e)

Fig. 7. The cooperation among robots

5 Conclusions

In our research, a cooperative control method using the evaluation information on objective achievement was applied to the real robots of RoboCup Middle Size League. According to our approach, a qualitative information is used and the achieving degree of the system objective is calculated from the information communicated from each agent. The effectiveness of the proposed method was demonstrated in the RoboCup 2004 competition.

Acknowledgments

This research work is supported in part by Grant in Aid for the 21st century center of excellence for "System Design: Paradigm Shift from Intelligence to Life" from Ministry of Education, Culture, Sport, and Technology in Japan and the foundation of Keio university.

References

1. L. Parker, ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation, IEEE Transaction on Robotics and Automation, vol.14, no.2, pp.220-240, 1998.
2. K. Ozaki, H. Asama, Y. Ishida, A. Matsumoto, K. Yokota, H. Kaetsu and I. Endo, Synchronized Motion by Multiple Mobile Robots using Communication, Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp.1164-1170, 1993
3. Hikari Fujii, Daiki Sakai, and Kazuo Yoshida, Cooperative Control Method Using Evaluation Information on Objective Achievement, Proceedings of the 7th Int. Symposium on Dis-tributed Autonomous Robotic Systems (DARS 04), 2004
4. E. Uchibe, T. Kato, M. Asada and K. Hosoda, Dynamic Task Assignment in a Multiagent/Multitask Environment based on Module Conflict Resolution, Proceedings of IEEE International Conference on Robotics & Automation , pp.3987-3992,2001
5. T.S. Dahl, M.J. Mataric and G.S. Sukhatme, Emergent Robot Differentiation for Distributed Multi-Robot Task Allocation, Proceedings of the 7th Int. Symposium on Distributed Autonomous Robotic Systems (DARS 04), pp. 191-200, 2004
6. B. P. Gerkey, M. J. Mataric, A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems, the International Journal of Robotics Research vol.23, No.9, pp. 939-954, 2004.
7. R. Emery, K. Sikorski and T. Balch, Protocols for Collaboration, Coordination and Dynamic Role Assignment in a Robot Team, Proceedings of IEEE International Conference on Ro-botics & Automation, pp3008-3015,2002
8. L. Chaimowicz, M.F.M. Campos and V. Kumar, Dynamic Role Assignment for Cooperative Robots Proceedings of IEEE International Conference on Robotics & Automation, pp3008-3015,2002.
9. Thilo Weigel, Jens-Steffen Gutmann, Markus Dietl, Alexander Kleiner, and Bernhard Nobel, CS Freiburg: Coordinating Robots for Successful Soccer Playing, IEEE Transactions on Robotics and Automation, Vol.18, No.5, 2002
10. L. Iocchi, D. Nardi, M. Piaggio, A. Sgorbissa, Distributed Coordination in Heterogeneous Multi-Robot Systems, Autonomous Robots 15(2), pp.155-168, 2003.
11. A. D'Angelo, E. Menegatti, E. Pagello, How a Cooperative Behavior can emerge from a Robot Team, Proceedings of the 7th Int. Symposium on Distributed Autonomous Robotic Systems (DARS 04), pp.71-80, 2004

Dynamic Positioning Based on Voronoi Cells (DPVC)

HesamAddin Torabi Dashti, Nima Aghaeepour, Sahar Asadi, Meysam Bastani, Zahra Delafkar, Fatemeh Miri Disfani, Serveh Mam Ghaderi, Shahin Kamali, Sepideh Pashami, and Alireza Fotuhi Siahpirani

Math and Computer Science Department,
Faculty of Science, University of Tehran, Iran
{dashti, nimaa, asadi, bastani, delafkar, fmiri, ghaderi, skamali, pashami, fotuhi}@khayam.ut.ac.ir
http://www.fos.ut.ac.ir/robocup

Abstract. In this paper we are proposing an approach for flexible positioning of players in Soccer Simulation in a Multi-Agent environment. We introduce *Dynamic Positioning based on Voronoi Cells* (DPVC) as a new method for players' positioning which uses *Voronoi Diagram* for distributing agents in the field. This method also uses *Attraction Vectors* that indicate agents' tendency to specific objects in the field with regard to the game situation and players' roles. Finally DPVC is compared with SBSP as the conventional method of positioning.

1 Introduction

In Multi-Agent Systems collaboration between individual agents has a critical role, since each agent decides and acts independently in order to achieve the common goal. In RoboCup Soccer Simulation each agent receives information from environment and builds a world model. Using this world model, agents are able to recognize their position in the field toward other agents. Each agent should cooperate with its team by selecting an appropriate action. Thus there should be a strategy which leads agents to take the best action. The most important aspects of Multi-Agent soccer strategy are as follow:

- The strategy must specify the formation of the team and the position of the agents inside this formation.
- The strategy must define different roles inside a formation and assign these roles to various player positions.
- For each role, the strategy must specify the behavior which is associated with the player's current role.
- The strategy must incorporate information about how an agent should adapt its behavior to the current situation.
- The strategy must indicate how each player should manage its battery during the game [1].

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 219–229, 2006.

To achieve these goals, we must distinct different players' states. There is always one player who is associated with ball and obeys a different decision method based on the strategy; other agents should move toward their best position in the field. So positioning is vital and should be carefully implemented in the strategy. The most popular method for dynamic positioning in RoboCup Soccer Simulation is SBSP (Situation Based Strategic Positioning)[2] which is presented by FC Portugal team. This method defines specific target positions for agents who do not possess the ball; these target positions are calculated with regard to the current formation of the team and roles of agents in this formation.

Although SBSP is a very powerful method for positioning it has some restrictions, one of them is the obligation to define *Home Positions*. In this paper we present a simple and flexible positioning method which we call *Dynamic Positioning based on Voronoi Cells* (DPVC). DPVC uses Voronoi Diagram to calculate target positions of agents.

Section 2 describes basics of positioning also shows restrictions of SBSP. In Sec. 3 Voronoi Diagram is introduced and a new algorithm for computing Voronoi Cells is presented. In Sec. 4 we show how to use a simple version of DPVC for positioning. In Sec. 5 we apply attraction vectors and study the behavior of DPVC. Section 6 includes some experimental results and comparisons between SBSP and DPVC.

2 Basics of Positioning

As mentioned before, in SBSP players calculate their target position by getting the game *situation*, *formation* of team, players' *roles* and their current position as input.

Situation of the game indicates rate of offence or defense considering ball position, position of teammates and opponents. Formation is a set of information each player needs to do positioning. It can be defined as an arrangement of players in the field which assigns a role to each agent. Role of an agent specifies characteristics of that agent regarding to the team formation and current position of the agent. Generally formations are used to keep players distributed in the field, to do so SBSP defines a home position for each role in the field. An agent determines its target position by computing a weighted sum of its home position and current position of the ball as an attraction point [1]. It is important to note that each role has specific attraction to ball. Also each player has a boundary area that can not exceed outside it.

For the player who is associated with the ball, no home position is defined and its movement is not bounded in a closed area. If such a player looses the ball, it has to move into its teammates' regions to return to its own region. This extra movement is a waste of battery.

A solution for this problem is to switch positions within a formation. However, switching positions can also cause increased player movement if a player has to move across the field to occupy its new position [5].

In most conventional positioning methods there is no significant change in players relative positions; because in each formation number of players getting the same role is fixed. This number changes by switching to another formation. Considering this fact, number of players in roles is predictable for the opponent team to adapt its strategy accordingly if it is provided with a coach.

The above problems can be solved by omitting home positions and formations as it is done in DPVC. DPVC computes attractions, roles, and target positions without using formation. It causes an increase in the importance of attractions for controlling agents to implement strategy.

In DPVC each agent has specific attractions to special objects like ball, opponent goal and etc. These attractions are presented by force vectors, applied to the agents. In our implementation of DPVC, vector sum of different attraction vectors determines an agent's movement; this vector is a function of different parameters in the field including ball position, opponent and teammate goal positions in the field. This means that attraction vectors show tendency of players to specific objects in the field. Attraction vectors are not the same for different roles; they are also influenced by game situation.

By removing formations, each agent's movements wouldn't be restricted in a bounded region and agents may gather in specific locations. This is due to the public attractions of some objects in the field. For example by applying ball attraction, players may aggregate around the ball. Avoiding this problem, players must repulse each other. To distribute players, we introduce a method which is based on Voronoi Diagram.

3 Voronoi Diagram

As mentioned before the method introduced in this paper for distributing agents is based on Voronoi Diagram. It helps us to divide the field into 10 cells (a cell for each player of the team except goalie). Since goalie has a different behavior, it obeys a different positioning method. By computing center of each cell and tending agents toward the center of their cells, an acceptable method for distributing agents in the field will be achieved. We now define Voronoi Diagram formally and then present an adaptive algorithm to soccer situation for computing each agent's Voronoi Cell.

3.1 Basics of Voronoi Diagram

If $P = \{p_1, p_2, \ldots, p_n\}$ be a set of n distinct objects (sites) in the plane; Voronoi Diagram of P will be the subdivision of the plane into n cells, one for each site in P, with the property that a point q lies in the cell corresponding to a site p_i iff $\mathrm{dist}(q, p_i) < \mathrm{dist}(q, p_j)$ for each p_j when $j \neq i$ [3].

Voronoi Diagram is usually used in situations where a plane should be partitioned into cells of influence, as it will be discussed in the field of soccer simulation. It can be shown that the Voronoi Diagram is a set of subdivisions whose edges are straight line segments. It means that the Voronoi Diagram is a set of convex polygons.

We can use Voronoi Diagram to have a dynamic positioning. Voronoi Cell of an agent is defined as the partition that contains the agent. In the first step we need an algorithm for computing each agent's Voronoi Cell.

3.2 Computing Voronoi Diagram

The most popular algorithm for computing Voronoi Diagram is introduced by Steve Fortune [3]. Although Fortune Algorithm computes Voronoi Diagram of n object in $O(n \log n)$ [3] and this makes it an ideal algorithm for most applications but it has some complexities due to use of data structures like priority queues and binary trees. On the other hand in a Multi-Agent System for each agent we need only its own Voronoi Cell whereas Fortune algorithm makes all the cells. Because of these problems in DPVC instead of Fortune Algorithm a new method is used which is more compatible with a Multi-Agent System. Time complexity of this method is $O(n^2)$. Although this is worse than Fortune Algorithm, we compute Voronoi Cells for only 10 agents so optimal time of Fortune Algorithm is not an important positive point.

We developed an algorithm (CalculateVC) that computes Voronoi Cell of agents separately every sense. Thus instead of calculating Voronoi Diagram for all agents, each player calculates its related cell.

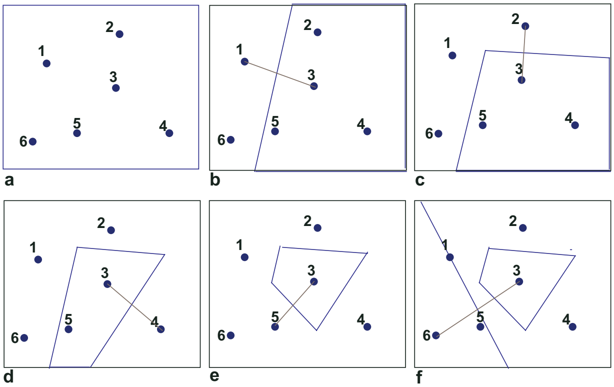


Fig. 1. Steps for computing the Voronoi Cell of agent 3

To compute the cell of agent i ($1 \leq i \leq 10$), at the beginning of the algorithm we consider the entire field as cell of this agent (Fig. 1) then in a greedy method this *initial Cell* will be abounded 9 times to get the desired result in a way that the perpendicular bisectors of agent i and other 9 agents are constructed. Each of these lines intersects the cell of agent i at most in two points, resulting in two polygons that are both convex. From these polygons the one that contains agent i will be considered as the cell of agent i. Figure 1 shows the consecutive steps of this algorithm.

Alg.1. Computing Voronoi Cell of agent i

```
Algorithm CalculateVC (agent i):

  // Input: an agent indexed $i$
  // Output: a polygon (cell) as the Voronoi Cell of input agent
  // Consider the entire field as the initial Voronoi Cell of $i$
  $cell_i \leftarrow$ all the field
  Point $P_i \leftarrow$ position of $i$
  for all agents $j \neq i$ do
    Point $P_j \leftarrow$ position of agent $j$
    Line $L \leftarrow$ perpendicular bisector of $P_i$ and $P_j$
    Intersect $L$ with $cell_i$
    if L intersects $cell_i$ in 0 or 1 points then
      Continue
    end if
    // L cuts $cell_i$ into two polygons (left and right)
    Polygon $cell_0 \leftarrow$ right polygon
    Polygon $cell_1 \leftarrow$ left polygon
    if $cell_0$ contains $P_i$ then
      $cell_i \leftarrow cell_0$
    else
      $cell_i \leftarrow cell_1$
    end if
  end for
```

CalculateVC makes 9 intersection lines. Calculating intersection points of a line and a convex polygon with k edges needs $O(k)$ time using Sutherland-Hodgeman Algorithm[4]. Since there are 9 intersection lines, k is always less than 9. As a result, time complexity of CalculateVC is $O(9 * 9)$.

4 Dynamic Positioning Based on Voronoi Cells (DPVC)

In DPVC for each agent we use its Voronoi Cell to present a vector to tend that agent in a direction for repulsing its teammates, this vector is called *Voronoi Vector*. Voronoi Vector of each agent will be combined by all attraction vectors of that agent and the result is the force vector that should be applied to the agent in each sense. It means that Voronoi Vectors' duty is to distribute agents in the field. We now explain how to use Voronoi Cells to achieve a method for distributing agents. In the first step for agent i $(1 \leq i \leq 10)$ the Voronoi Cell is computed (see Alg.1). This cell is a convex polygon with k_i edges. In the second step center of this cell is computed using algorithms based on computational geometry. We can define the center of a simple polygon as the center of mass of a two dimensional object having a shape like that polygon , assuming that the density is the same in all parts of the object. The vector from an agent's position toward the center of its Voronoi Cell is defined as the Voronoi Vector of that agent.

According to the definition of Voronoi Cell we can approximate compression (density) of other agents around an agent by the space of that agent's Voronoi Cell. If this area is increased, the compression of the other agents near that cell (agent) will be decreased. The relative positioning (direction and distance) of an agent toward the center of its cell shows the state of that agent in regards to the other agents' compression zone. If an agent is near to the center of its cell, its distance from other agents is near optimal. For example in Fig. 2 (a) cell of agent 1 is smaller than other cells, this means that the compression is high around this cell. We can also see that the center of cell 1 is near to agent 1 so this agent is in the same distance from other agents (on the average) and its relocation (in this sense) does not help to remove compression. Notice that cell 5 has a wider area and as a consequence the compression is low around it. We can also see that the agent 5 is far from center of its cell and it is near to the compression zone so by moving agent 5 to the center of cell 5, compression will be decreased. It is possible to verify this point for other agents. As it is shown in Fig. 2 (b) and Fig. 2 (c) by moving the agents towards the center of their cells, areas of the wider cells (like cell 5) decreases and areas of smaller cells(like cell 1) increases. It is wise to consider the equality of Voronoi Cell areas as a factor (or definition) for distribution, so in a sequence of senses the agents get more distributed and this process continues until all the agents reach a balance state. This means that the agents' positions and the center of their cells will be in the same point. In other word the Voronoi Vector will be zero vector. To show that the agents reach a balance state after some senses, it is necessary to define center of a polygon in such a way that by moving an agent toward the center of its cell, summation of its distances from the other agents increases. Defining center of mass as the center of a polygon (as it is suggested in DPVC) satisfies this condition. With this definition, to show that agents reach equivalency we use these notations: $d_{ij} =$ distance between agent i and agent j, $P_s^i = \sum_{j=1}^{10} d_{ij}$ for agent i in sense s, $sumD_s = \sum_{k=1}^{10} p_s^k$ in sense s, Considering the definition for center of a polygon it is obvious that $sumD$ has an ascendant proceed. This means that: $sense_a < sense_b \rightarrow sumD_{sense_a} < sumD_{sense_b}$. In other word when agents move toward the center of their cells summation of

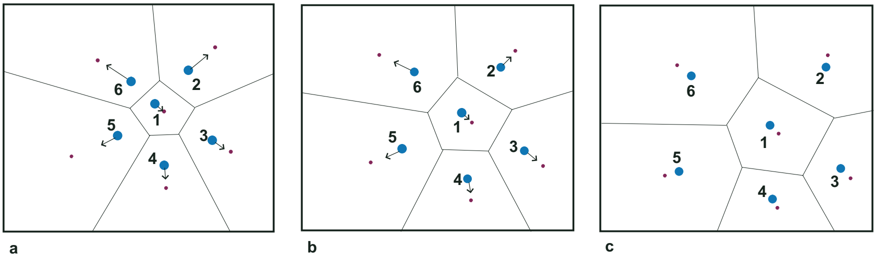


Fig. 2. Agent's movement toward their cell's center divides the field into equal areas

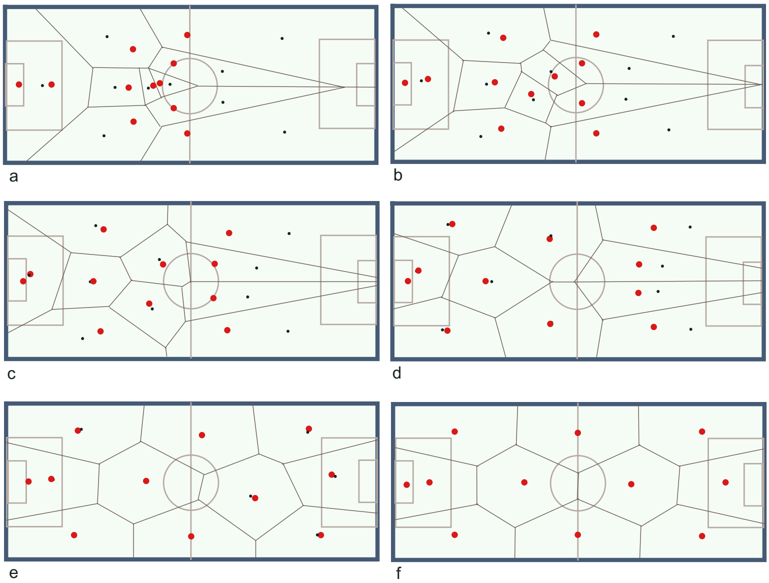


Fig. 3. Agents' movement using DPVC without attraction in rcssserver3D

distances between all players ($sumD$) rises. Also the field of match (generally the rectangle on which we make Voronoi Diagram) is bounded , so this ascendant proceeding should be stopped because $sumD$ can not be more than a limit and after t senses it will stop. In this sense agents get equivalency. This means that if we force agents toward the center of their cells (by applying Voronoi Vectors) they move in a path that results in an equivalence state.

The final shape of Voronoi Diagram after getting equivalency depends on the initial formation of agents in the field, it means that different initial formations result in different final shapes and there is no unique final shape. Figure 3 shows agents movement using DPVC without attraction in RoboCup Soccer Simulation Server 3D (rcssserver3D).

Observation shows that if Voronoi Diagram in final equivalence state has some sort of symmetry toward X axis and Y axis the result will be a stable equivalence. This means that if we apply a little change in the position of an agent (after reaching the equivalency), in the next sense Voronoi Diagram will be formed in a way to rectify this change. Thus the agent moves toward its last position to get the equivalence again. Notice that it differs from defining a static point like home position for agents, because if the agent's displacement increases more than a certain limit, DPVC probably tends agents to a new equilibrium state. On the other hand in real conditions where attractions are applied, the system is so dynamic that agents never get an equilibrium state, they are just forced in an equivalency path that is updated in every sense.

If in the final equilibrium state the Voronoi Diagram becomes asymmetric, the result will be an unstable equivalency, in other word a little change in the position of an agent makes the other agents move continuously until they get a new stable or unstable equilibrium. In this new equivalency, shape of Voronoi Cells differs from last equilibrium state.

In our earlier discussion of distributing players, Voronoi Cells are constructed from only teammates (except goalie) as Voronoi constructing points (sites); but considering game situation and role of players, it is a good idea to distribute some of agents in open spaces between opponents to make passes more successful. If we add opponents' positions to the set of points (sites) that construct Voronoi Cells, the field will be partitioned into 21 cells i.e. a cell for each player in the field except goalie of our team. By applying this method on the agents specified by situation and role, their Voronoi Vectors will be constructed in a way that make them to move toward opponent team's openings. In other word these agents have a kind of repulsion to both opponents and teammates.

5 DPVC with Attraction

When we apply both attraction vector and Voronoi Vector, the agent has acceptable behavior. In this condition the agents repulse each other (due to Voronoi Vectors) and also attract certain objects like ball and opponent or teammate goal with respect to the match situation (due to attraction vectors).

Attraction vectors prevent agents from getting equivalency, so some of them are always moving but as it was indicated before a force (Voronoi Vector) is applied to agents in a path to achieve equivalency and it distributes agents in the field. For example when an agent recede center of its cell (because of its attraction vectors), the Voronoi Vectors of other agents move them in a way to fill that agent's opening.

Voronoi Vectors should be applied in an optimal direction. It is not desirable that direction of Voronoi Vectors change in two sequential senses; because this probably makes agents to do unnecessary extra movements and this is a waste of battery. This undesirable condition appears when the Voronoi Vector is applied to agent in direction of getting an unstable equilibrium. However the probability that this undesirable condition occurs is very low because agents should be in very rare and scarce positions to be forced toward an unstable equivalency. On the other hand matches are so dynamic and in each sense attraction vectors change, so in real conditions agents' movement is practically optimal. This fact is adapted with physics too. the probability that a moving sphere stops on the top of a hill (unstable equivalency) is so lower than stopping in the bottom of hill (stable equivalency), because the equilibrium zone is so limited on the top.

Considering DPVC we have an efficient method that can distribute agents in the field and also has some tools for implementation of a strategy. As explained in Sec. 2 attraction vectors determine agents' behavior. So it is important to assign appropriate attraction vectors for each agent with respect to its role and game situation. We must define *attraction functions* to set attraction vectors in a way to coordinate the players. These functions get game situation and player's role as input. So in different situations, attraction vectors change; for example in offensive situation there is more attraction towards opponent's goal. As a conclusion team strategy can be applied by attraction functions.

Also in DPVC number of players in each role is not fixed and changes according to the game situation. As a result agents' behavior is more flexible and it reduces the team's predictability, and this is a positive point for DPVC in comparison to SBSP.

As mentioned in Sec. 2 in conventional methods of positioning when formation changes, agents have to move across the field to get their new positions. But in DPVC, this extra movement is omitted due to the fact that destination of agents is updated in each sense, also every sense Voronoi Cells of each agent is updated and its shape changes continuously. These changes are concrete i.e. there is no sudden change in the shape of Voronoi Cell of an agent in two consecutive senses. As a result Voronoi Cells' center of masses and consequently Voronoi Vectors (destination of agents) change continuously.

Considering these facts it is obvious that DPVC method has properties of a strategic positioning as are indicated in Sec. 1.

6 Experimental Results

Performance of a team not only depends on its positioning method but also depends on other decision modules and efficiency in implementation of agents skills. Accordingly it is difficult to define an appropriate criterion to evaluate the positioning method. In order to survey the applied positioning, we compared two similar teams using different positioning methods in rcssserver3D. One of these teams uses SBSP as positioning method and the other uses DPVC. Since there

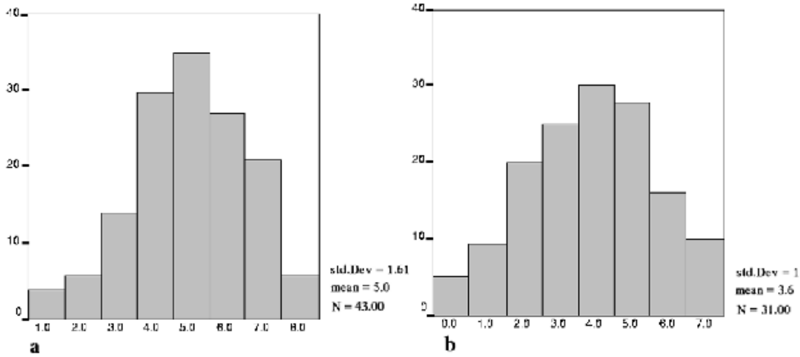


Fig. 4. The figure shows statistical distribution for average number of passable players of the team against Aria, using DPVC and SBSP methods for positioning. N is number of the team opportunities to pass the ball. Numbers in x-axis show number of passable players while numbers in y-axis are times that these numbers occur. As it is shown while using DPVC in average there are five passable players whereas by using SBSP there are only 3.6 passable players.

Table 1. Results when two similar teams using different positioning methods (DPVC and SBSP) play against Aria team

	DPVC	SBSP
Average number of passable players	4.09	3.36
Ball in control of the team	58.31%	56.92%
Ball in own half	18.80%	22.15%
Ball in own penalty area	1.47%	1.26%
Ball in opponent penalty area	23.02%	19.98%

is usually little density of players near corners of the field, we improved DPVC by restricting the initial Voronoi Cells of agents to a hexagonal surrounded by the entire field rectangle.

To compare these two positioning methods we prepared two experiments. In these experiments both teams play against Aria team the champion of RoboCup 2004 3D competitions. In the first experiment the number of passable players around the ball when the ball is in possession of the testing team is measured. Passable player is a player who has the opportunity to get the ball if it is passed. So being a passable player is a matter of player's distance from the ball. In our experiment a player is defined to be passable if its distance from ball is less than 20 meters. Fig. 4 is a statistical diagram of passable players of the team. In Fig. 4 (a) DPVC is used as the positioning method, whereas in Fig. 4 (b) the positioning is based on SBSP.

In the second experiment both the team using DPVC and the team using SBSP are ran against Aria 10 times. Table .1 reports results of these two series of tests. Records of this table show parameters defined to compare the influence of each method of positioning on success of the team.

7 Conclusion

Although we have set the parameters of attraction functions experimentally, but the experimental results show that DPVC behavior is acceptable as a method of positioning. We hope to improve DPVC by applying a learning method like Reinforcement Learning. Mathematical analysis of Voronoi Diagram when agents (sites) are moving can be an open field of study.

We are also working on implementing attractions (e.g. attraction to goals) indirectly with restricting initial Voronoi Cells of agents moving their vertical bounds toward opponent goal or teammate goal.

In this paper we used a geometrical model (Voronoi Diagram) for distributing agents; there are other models that can be used. For example we are surveying some physical models in which agents can be assumed electrical charges in a 2D space. In this way attractions can be assigned by defining positive or negative charges for different objects; in this way potential field on the ground direct agents (electrons) towards less potential points and cause agents' movement in the correct direction.

References

1. de Boer, R., Kok, J.: The Incremental Development of a Synthetic Multi-Agent System: The UvA Trilearn 2001 Robotic Soccer Simulation Team. M.S. Thesis. Faculty of Science, University of Amsterdam (2002) Section 9.5.3
2. Reis, L.P., Lau, N., Oliveira, E.C., Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents, Springer's Lecture Notes in Artificial Intelligence, Vol.2103, Berlin (2001) 175–197
3. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O., Computational Geometry, Springer–Verlag, New York (2000) Chapter 7
4. Sutherland, I.E., Hodgman, G.W., Reentrant polygon clipping, Communications of the ACM, Vol.17 No.1 (1974) 32–42
5. Stone, P., Layered Learning in Multi-Agent Systems. PhD Thesis. School of Computer Science, Carnegie Mellon University (1998) Chapter 3

Gaze Direction Determination of Opponents and Teammates in Robot Soccer*

Patricio Loncomilla[1] and Javier Ruiz-del-Solar[1,2]

[1] Department of Electrical Engineering, Universidad de Chile
[2] Center for Web Research, Department of Computer Science, Universidad de Chile
{ploncomi, jruizd}@ing.uchile.cl

Abstract. Gaze direction determination of opponents and teammates is a very important ability for any soccer player, human or robot. However, this ability is still not developed in any of the RoboCup soccer leagues. We aim at reverting this situation by proposing a gaze direction determination system for robot soccer; the system is designed primarily for the four-legged league, but it could be extended to other leagues. This system is based on a robot-head pose detection system, consisting on two main processing stages: (i) computation of scale-invariant local descriptors of the observed scene, and (ii) matching of these descriptors against descriptors of robot-head prototypes already stored in a model database. After the robot-head pose is detected, the robot gaze direction is determined using a head model of the observed robot, and the current 3D position of the observing robot camera. Experimental results of the proposed approach are presented.

1 Introduction

Among many other capabilities, good soccer players should have the ability for anticipating the actions of opponents, and sometimes of teammates, by just observing the other players attitude and pose. As in other similar situations, the human most employed mechanism for solving this task is gaze direction determination, or the determination of the place where the opponent or teammate player under analysis is looking. For instance, by using this mechanism an attacker player can know if an opponent is observing him, and then planning his next actions for avoiding that the opponent attack him or obstruct his trajectory. In another typical situation a soccer player can know where the ball is, by looking at the same position where an opponent is looking (in case the opponent knows the ball position). In a third situation a soccer player can send the ball, i.e. perform a pass, to a position where a teammate is looking at. Furthermore, when kicking the ball, first-class soccer players can mislead opponents by looking at a different place than the place where they are sending the ball. Some examples of these typical situations are shown in figure 1.

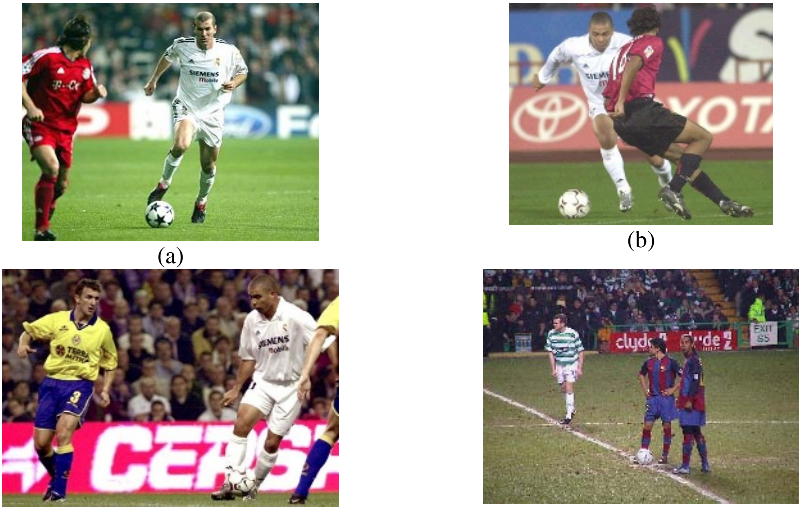
* This research was funded by Millenium Nucleus Center for Web Research, Grant P04-067-F, Chile.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 230 – 242, 2006.

On hand of the described situations, it can be affirmed that gaze direction determination of opponents and teammates is a very important ability for any soccer player, robot or human. However, this ability is still not developed in any of the RoboCup soccer leagues. We aim at reverting this situation by proposing a gaze direction determination system for robot soccer. This system is designed primarily for the four-legged league, but it could be extended to other leagues. Moreover, the same gaze determination methodology can be used for enhancing cooperative and competitive skills in situations where the robots interacting abilities are important.

In the here-proposed approach, gaze direction determination is based on a robot-head pose detection system. This detection system employs two main processing stages. In the first stage, scale-invariant local descriptors of the observed scene are computed. Then, in the second stage these descriptors are matched against descriptors of robot-head prototypes already stored in a model database. After the robot-head pose is recognized, the robot gaze direction is determined using a head model of the observed robot, and the current 3D position of the observing robot camera. In the here-employed robots (Sony AIBO) the relation between head and camera pose is fixed, therefore additional camera pose determination is not required.

The local descriptors computation and matching are based on [1], but many important parts of the method have been improved for fitting it to the robot-head detection problem and for achieving high detection accuracy.



(a)

(b)

(c)

(d)

Fig. 1. Some examples of real soccer situations where the gaze direction determination plays an important role. (a) and (b) An attacker player knows if an opponent is observing him and at which distance. (c) A defender knows where the ball is, by looking at the same place where the attacker is looking. (d) Soccer players can mislead opponents by looking at a different place than the place where they are sending the ball.

2 Related Work

Human gaze direction (i.e. line of gaze) determination has been the subject of a great number of studies (for example [3][4][8][9]), with applications in very different fields such as medical research for oculography determination, car drivers behavior characterization, human-robot and human-computer interaction, including computer interfaces for handicapped people. However, to our knowledge there are no studies on determining the gaze direction in robots. We believe this is a problem that needs to be solved for enhancing and enriching cooperative and competitive tasks in which the robots interacting capabilities are important (i.e. robot soccer). Already developed methodologies employed for human gaze direction determination are not applicable for robots. They are based on anthropometric models of the human head and eyes, or they employ face or iris detection algorithms, or even special lighting (infrared lights). Therefore, new methodologies need to be employed for the robot case. Some alternatives could be the construction of explicit 3D robot-head models, the development of robot-face detection algorithms or the use of scale-invariant local descriptors for performing the detection. Taking into account the impressive development of object recognition algorithms based on scale-invariant descriptors in the last years ([1][6][7]), and the fact that head and face variability in robots is much smaller than in humans, we believe that for the moment, they are the best methodology for solving this problem. Most successful proposed systems employ either the Harris detector [5] or SIFT (Scale Invariant Feature Transform) features [1] as building blocks. In this work we employ SIFT features because of their higher robustness and stability. However, due to the physical characteristics of some robots models as the AIBO ERS7 (rounded head shape and head surface producing a high amount of highlights), it is very difficult to obtain reliable SIFTs on them. For this reason, we improve the traditional SIFTs computation and matching algorithms.

3 Proposed Robot Gaze Direction Determination System

3.1 Scale-Invariant Local Descriptors Computation

Detection of scale-space extrema. A difference-of-Gaussian (DoG) function is employed for identifying potential interest points that are invariant to scale and orientation. These keypoints are searched over all scales and image locations using a scale-space transformation. It can be proved that by using the DoG over the scale-space, image locations that are invariant to scales can be found, and that these features are more stable than other computed using the gradient, Hessian or Harris corner function [1]. The scale-space of an image is defined as a function, $L(x,y,\sigma)$, which corresponds to the convolution of the image with a Gaussian of scale σ. The DoG function between two nearby scales separated by a constant multiplicative factor k can be computed as:

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$$

The local extrema (maxima and minima) of $L(x, y, \sigma)$ are detected by comparing each sample with its 26 neighbors in the scale-space (8 in the same scale, 9 in the scale above and 9 in the scale below).

Accurate keypoint localization. First, local extrema to sub-pixel / sub-scale accuracy are found by fitting a 3D quadratic to the scale-space local sample point. The quadratic function is computed using a second order Taylor expansion having the origin at the sample point [2]:

$$D(\mathbf{x}) = D(\mathbf{0}) + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \qquad (1)$$

where $\mathbf{x}$ is the offset from the sample point. Then, by taking the derivate with respect to $\mathbf{x}$ and setting it to zero, the location of the extrema of this function is given by:

$$\hat{\mathbf{x}} = -H^{-1}\nabla D(\mathbf{0}) \qquad (2)$$

In [1][2] the Hessian and gradient are approximated by using differences of neighbor samples points. The problem with this coarse approximation is that just 3 samples are available in each dimension for computing the Hessian and gradient using pixel differences, which produces a non-accurate result. We improve this computation by using a real 3D quadratic approximation of the scale-space, instead of discrete pixel differences. Our 3D quadratic approximation function is given by:

$$\tilde{D}(x, y, \sigma) = a_1x^2 + a_2y^2 + a_3\sigma^2 + a_4xy + a_5x\sigma + a_6y\sigma + a_7x + a_8y + a_9\sigma + a_{10}$$

Using the 27 samples contained in the 3x3x3 cube under analysis, the unknowns (a_i) can be found. Using vector notation, this linear system will be given by:

$$\begin{bmatrix} x_1^2 & y_1^2 & \sigma_1^2 & x_1y_1 & x_1\sigma_1 & y_1\sigma_1 & x_1 & y_1 & \sigma_1 & 1 \\ x_2^2 & y_2^2 & \sigma_2^2 & x_2y_2 & x_2\sigma_2 & y_2\sigma_2 & x_2 & y_2 & \sigma_2 & 1 \\ & & & & \ldots & & & & & \\ x_{27}^2 & y_{27}^2 & \sigma_{27}^2 & x_{27}y_{27} & x_{27}\sigma_{27} & y_{27}\sigma_{27} & x_{27} & y_{27} & \sigma_{27} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \ldots \\ a_{10} \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ \ldots \\ D_{27} \end{bmatrix}$$

where D_i corresponds to the sample point value (intensity) i. We can write this linear system as $\mathbf{Ba} = \mathbf{d}$. The least-squares solution for the parameters $\mathbf{a}$ is given by:

$$\mathbf{a} = \left(\mathbf{B}^\mathbf{T}\mathbf{B}\right)^{-1}\mathbf{B}^\mathbf{T}\mathbf{d}$$

It should be stressed that the matrix $\left(\mathbf{B}^\mathbf{T}\mathbf{B}\right)^{-1}\mathbf{B}^\mathbf{T}$ needs to be computed once for the whole image, and that it can be eventually pre-computed. Now, the accurate location of the extrema can be computed using (2), with the following Hessian and gradient expression:

$$\mathbf{H} = \begin{bmatrix} 2a_1 & a_4 & a_5 \\ a_4 & 2a_2 & a_6 \\ a_5 & a_6 & 2a_3 \end{bmatrix}; \ \nabla\tilde{D}(0) = \begin{bmatrix} a_7 \\ a_8 \\ a_9 \end{bmatrix} \qquad (3)$$

Second, local extrema with a contrast lower (noise) than a given threshold Th_{contr}, are discarded ($|\tilde{D}(\hat{\mathbf{x}})| < Th_{contr}$). Third, extrema corresponding to edges are discarded using curvature analysis. A peak that corresponds to an edge will have a large principal curvature across the edge but a small one in the perpendicular direction. The curvature can be computed from the 2x2 submatrix $\mathbf{H}_{xy}$ that considers only the x and y components of the Hessian. Taking into account that we are interested on the ratio between the eigenvalues, we will discard extrema in which the ratio of principal curves is above a threshold *r*, or equivalently local extrema that fulfill the following condition (see [5] for a deeper explanation):

$$\frac{\mathrm{Tr}(\mathbf{H}_{xy})^2}{\mathrm{Det}(\mathbf{H}_{xy})} > \frac{(r+1)^2}{r}$$

In [1] $\mathbf{H}_{xy}$ is computed be taking differences of neighbor sample points. As already mentioned, this approximation produces a non-accurate result. We improved this situation by computing $\mathbf{H}_{xy}$ from (3).

Orientation assignment. By assigning a coherent orientation to each keypoint, the keypoint descriptor can be represented relative to this orientation and hence achieve invariance against rotations. The scale of the keypoint is employed for selecting the smoothed image *L*(x,y) with the closest scale, and then the gradient magnitude and orientation are computed as:

$$\mathrm{m}(x,y) = \sqrt{(L(x+1,y)-L(x-1,y))^2 + (L(x,y+1)-L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1)-L(x,y-1))/(L(x+1,y)-L(x-1,y)))$$

As in [1], an orientation histogram is computed from the gradient orientations at sample points around the keypoint (*b1* bins are employed). A circular Gaussian window whose size depends of the scale of the keypoints is employed for weighting the samples. Samples are also weighted by its gradient magnitude. Then, peaks in the orientation histogram are detected: the highest peak and peaks with amplitudes within 80% of the highest peak. Orientations corresponding to each detected peak are employed for creating a keypoint with this orientation. Hence, multiple keypoints with the same location and scale but different orientation can be created (empirically, about 85% of keypoints have just one orientation).

Keypoint descriptor computation. For each obtained keypoint, a descriptor or feature vector that considers the gradient values around the keypoint is computed. The obtained descriptors are invariant against some levels of change in 3D viewpoint and illumination. The keypoints and their associated descriptors are knows as SIFT (Scale Invariant Feature Transform) features or just SIFTs.

First, in the keypoint scale the gradient magnitude and orientation are computed around the keypoint position (usually a neighborhood of 8x8 or 16x16 pixels is considered). Then, a Gaussian window weights the gradient magnitudes, and the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. Second, the obtained gradient values are accumulated in orientation histograms summarizing the contents of 4x4 subregions (*b2* bins are employed). Thus, a descriptor vector is built, where each vector component is given by an orientation histogram. Depending on the neighborhood size, 2x2 or 4x4 vectors are obtained. Third, illumination effects are reduced by normalizing the descriptor's vector to unit length. Abrupt brightness changes are controlled by limiting the intensity value of each component of the normalized vector. Finally, descriptors vectors are re-normalized to unit length.

3.2 Matching of Local Descriptors and Robot-Head Prototypes Descriptors

Basically, the robot-head pose is determined by matching the image descriptors with descriptors corresponding to robot-head prototype images already stored in a database. The employed prototypes correspond to different views of a robot head, in our case the head of an AIBO ERS7 robot. Due to we are interested in recognized the robot identity (number), prototypes for each of the four players are stored in the database. In figure 2 are displayed the 16 prototype heads corresponding to one of the robots. The pictures were taken every 22.5°. The whole matching process here-proposed considers nine processing stages. In the first stage, the image keypoint descriptors are individually matched against prototype descriptors. In the second stage this matching information is employed for obtaining a coarse prediction of the object (robot-head) pose. In the third stage possible affine transformations between a prototype and the located object are determined. In the later six stages these affine transformations are verified, and some of them discarded or merged. Finally, if the object is present in the image just one affine transformation should remain. This transformation determines the object pose. It is worth to mention than in the original work of Lowe [1], only the first four stages here-described were considered. We included five additional verification stages that improve the detection of robot heads. This is very important because due to the physical characteristics of the AIBO ERS7 heads (rounded head shape, head surface producing a high amount of highlights, etc.), it is very difficult to obtain reliable SIFTs on them.

Individual Keypoint Descriptors Matching. The best candidate match for each image keypoint is found by computing its Euclidian distance with all keypoints stored in the database. It should be remembered that each prototype includes several keypoint descriptors. Considering that not all keypoints are always detected (changes in illumination, pose, noise, etc.) and that some keypoints arise from the image background and from other objects, false matches should be eliminated. A first alternative is to impose a minimal value to a match to be considered correct. This approach has proved to be not robust enough. A second alternative consists on comparing the distance to the closest neighbor to that of the second-closest neighbor. If this ratio is greater than a given threshold, it means than this image keypoint descriptor is not discriminative enough, and therefore discarded. In [1] the closest neighbor and second-closest neighbor should come from a different object model (prototype). In the current case this is not a good idea, since we have multiple views of the same object (the robot head). Therefore, we impose the conditions than the second-closest neighbor should come from the same prototype than the closest neighbor. The image under analysis and the prototype images generate a lot of keypoints, hence having an efficient algorithm for computing the keypoint descriptors distance is a key issue. This nearest neighbor indexing is implemented using the Best-Bin-First algorithm [10], which employs a k-d tree data structure.

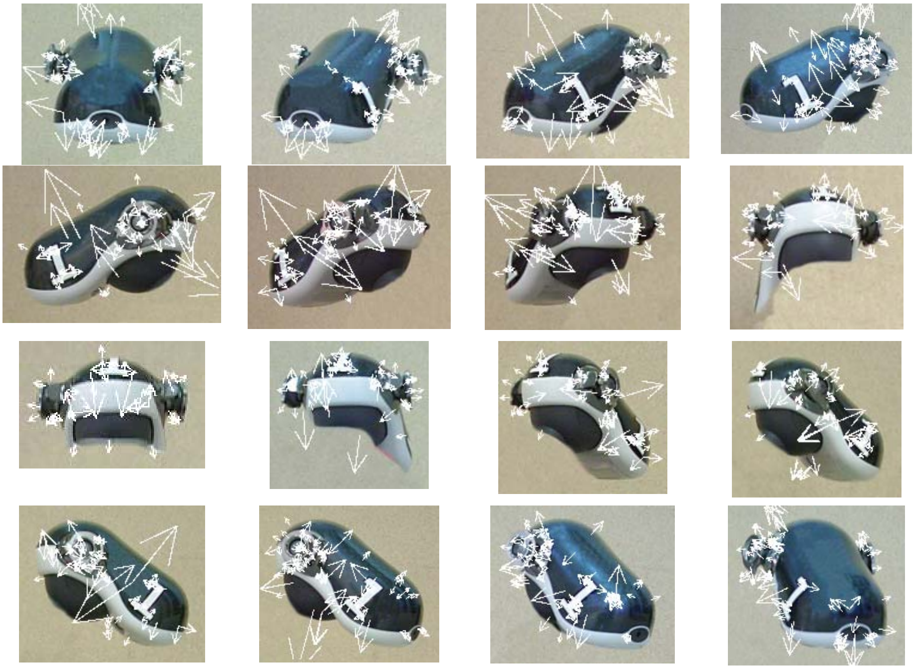


Fig. 2. AIBO ERS7 robot-head prototypes with their SIFTs. Pictures taken every 22.5°.

Object Pose Prediction. In the pose space a Hough transform is employed for obtaining a coarse prediction of the object (robot-head) pose, by using each matched keypoint for voting for all object pose that are consistent with the keypoint. A candidate object pose is obtained if at least 3 entries are found in a Hough bin. Usually, several possible object pose are found. The prediction is coarse because the similarity function implied by the four parameters (2D location, orientation and scale) is only an approximation of the 6 degree-of-freedom of a 3D object. Moreover, the similarity function cannot account for non-rigid deformations.

Finding Affine Transformations. In this stage already obtained object pose are subject to geometric verification. A least-squares procedure is employed for finding an affine transformation that correctly account for each obtained pose. An affine transformation of a prototype keypoint (x,y) to an image keypoint (u,v) is given by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

where the m_i represent the rotation, scale and stretch parameters, and t_x and t_y the translation parameters. The parameters can be found if three or more matched keypoints are available. Using vector notation, this linear system will be given by:

$$\begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} u \\ v \\ \cdots \\ \cdots \end{pmatrix}$$

We can write this linear system as $\mathbf{Cp} = \mathbf{u}$. Finally, the least-squares solution for the parameters **p** is given by:

$$\mathbf{p} = \left(\mathbf{C}^{\mathrm{T}}\mathbf{C}\right)^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{u}\cdot$$

Affine Transformations Verification using a Probabilistic Model. The obtained model hypothesis, i.e. affine transformations, is subject to verification using a probabilistic model (see detailed description in [11]).

Affine Transformations Verification based on Geometrical Distortion. A certain affine transformation shouldn't deform very much an object when mapping it. Given that we have just a hypothesis of the object pose, it is not easy to determine the object distortion. However, we do have the mapping function, i.e. the affine transformation. Therefore, we can verify if the mapping function produce distortion or not using a known, regular and simple object, such as a square. The affine transformation of a square should produce a rotated parallelogram. If the affine transformation does not produce a large distortion, the conditions that the transformed object should fulfill are (see notation in fig. 3):

$$\max\left\{\frac{d(AB)/d(A'B')}{d(BC)/d(B'C')},\frac{d(BC)/d(B'C')}{d(AB)/d(A'B')}\right\} < th_{prop} \;\; ; \;\; \alpha = \sin^{-1}\left|\frac{\det(\overrightarrow{A'B'} \quad \overrightarrow{B'C'})}{d(\mathrm{A'B'})\times d(\mathrm{B'C'})}\right| > th_{\alpha}$$

$\overrightarrow{A'B'}$ is a vector from A' to B', $\det\left(\overrightarrow{A'B'} \quad \overrightarrow{B'C'}\right)$ computes the parallelogram area.

Affine Transformations Verification based on Spatial Correlation. Affine transformations producing low lineal correlation, r_s, between the spatial coordinates of the matched SIFTs in the image and in the prototype are discarded:

$$r_s = \min\left(\max(r_{xx}, r_{xy}), \max(r_{yx}, r_{yy})\right) < th_{rs}$$

r_{xx} and r_{yy} correspond to the correlation in the x and y directions of the N matched SIFTs, while r_{xy}=r_{yx} corresponds to the cross correlation between both directions. r_{xx} and r_{xy} are calculated as (r_{yy} and r_{yx} are computed in a similar way):

$$r_{xx} = \left|\frac{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)\left(x'_i - \overline{x'}\right)}{\sqrt{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2 \sum_{i=1}^{N}\left(x'_i - \overline{x'}\right)^2}}\right| \; ; \; r_{xy} = \left|\frac{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)\left(y'_i - \overline{y'}\right)}{\sqrt{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2 \sum_{i=1}^{N}\left(y'_i - \overline{y'}\right)^2}}\right|$$

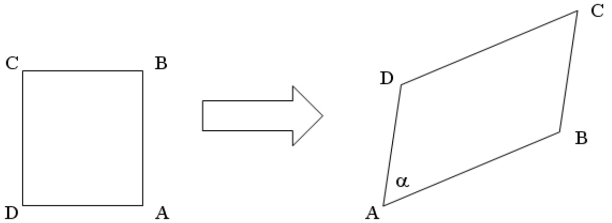


Fig. 3. Affine mapping of a square

Affine Transformations Verification based on Graphical Correlation. Affine transformations producing low graphical correlation, r_g, between the robot-head prototype image and the candidate robot-head subimage can be discarded:

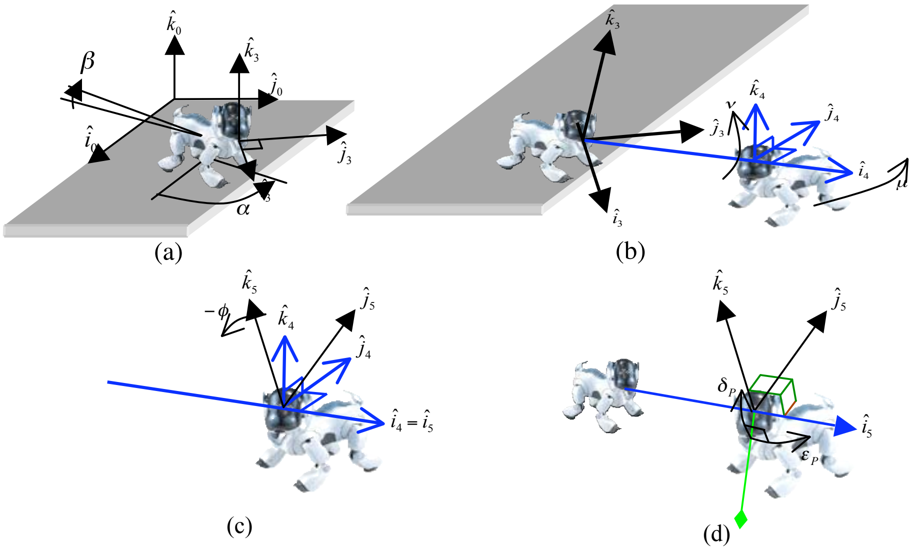
$$r_g = \frac{\sum_{u=0}^{U}\sum_{v=0}^{V}\left(I(u,v)-\overline{I}\right)\left(I'\left(x_{TR}(u,v),y_{TR}(u,v)\right)-\overline{I'}\right)}{\sqrt{\sum_{u=0}^{U}\sum_{v=0}^{V}\left(I(u,v)-\overline{I}\right)^2\sum_{u=0}^{U}\sum_{v=0}^{V}\left(I'\left(x_{TR}(u,v),y_{TR}(u,v)\right)-\overline{I'}\right)^2}} < th_{rg}$$

The affine transformation is given by {x=x$_{TR}$(u,v), y=y$_{TR}$(u,v)}. *I*(u,v) and *I'*(x,y) correspond to the head prototype image and the candidate robot-head subimage, respectively. $\overline{I}$ and $\overline{I'}$ are the corresponding pixel mean values.

Affine Transformations Verification based on the Object Rotation. In some real-world situations, real objects can have restrictions in the rotation (respect to the body plane) they can suffer. For example the probability that a real soccer robot is rotated in 180° (inverted) is very low. For a certain affine transformation, the rotation of a detected object (candidate robot-head) with respect to a certain prototype can be determined using the SIFTs keypoint orientation information. Thus, the object rotation, *rot*, is computed as the mean value of the differences between the orientation of each matched SIFTs keypoint in the prototype and the corresponding matched SIFTs keypoint in the image. Transformations producing large *rot* values can be discarded ($rot > th_{rot}$).

Affine Transformations Merging based on Geometrical Overlapping. Sometimes more than one correct affine transformation corresponding to the same object can be obtained. There are many reasons for that, small changes in the object view respect to the prototypes views, transformations obtained when matching parts of the object as well as the whole object, etc. When these multiple, overlapping transformations are detected, they should be merged. As in the case when we verify the geometrical distortion produce by a transformation, we perform a test consisting in the mapping of a square by the two candidate affine transformations to be joined. The criterion for joining them is the overlap, *over*, of the two obtained parallelograms (see notation in fig. 3):

$$over = 1 - \frac{dist(A'_1 A'_2) + dist(B'_1 B'_2) + dist(C'_1 C'_2) + dist(D'_1 D'_2)}{perimeter(A'_1 B'_1 C'_1 D'_1) + perimeter(A'_2 B'_2 C'_2 D'_2)} > th_{over}$$



(a) (b)

(c) (d)

Fig. 4. Defined reference systems (RFs) (see explanation in main text). (a) RFs ”0” and “3”. (b) RFs “3” and “4”, rotation angles μ and ν. (c) RFs “4” and “5”, and affine rotation angle ϕ. (d) Observed robot in RF “5”, line of gaze in green.

It should be also verified if the difference between the rotations produced for each transform is not very large. Therefore, two transforms will be joined if:

$$|rot_1 - rot_2| < th_{diff\_rot}.$$

3.3 Gaze Determination

The line of gaze of the observed robot, in global coordinates, can be computed using the following information: (i) observing robot pose in global coordinates, (ii) prototype view angle, and (iii) distance and rotation angle of the observed robot. The observing robot pose can be estimated using the self-localization system (any mobile robot control software has a similar system). The prototype view angle is fixed and known, it was defined when the model database was built. Finally, the distance and rotation angle of the observed robot can be determined from the affine transformation.

For performing the computations we define the following coordinate systems: $\{\hat{i}_0\ \hat{j}_0\ \hat{k}_0\}$, the global reference system (RF), $\{\hat{i}_3\ \hat{j}_3\ \hat{k}_3\}$, a RF fixed to the observing robot’s camera (between system “0” and “3” there are 3 coordinate transformations), and $\{\hat{i}_5\ \hat{j}_5\ \hat{k}_5\}$, a RF located at the observed robot’s head (between system “3” and “5” there are 2 coordinate transformations). The considered angles and distances are defined in table1.

In the RF 5, two points will define the line of gaze: the camera’s position of the observed robot, and the intersection of the gaze straight line (of parameter λ) with the floor. This straight line will be given in system “5” coordinates by:

$$x_5(\lambda) = -\lambda \cos\delta_P \cos\varepsilon_P \ ; \ y_5(\lambda) = -\lambda \cos\delta_P \sin\varepsilon_P \ ; \ z_5(\lambda) = \lambda \sin\delta_P$$

Table 1. Angles and distances definitions

	Definition	**Source**
α	Rotation angle of robot's body with respect to axis $\hat{i}_0$ with rotation axis $\hat{k}_0$	Self-localization
β	Elevation angle of robot's body with respect to plane $\{\hat{i}_0 \;\; \hat{j}_0\}$	Accelerometer
γ	Tilt 1 angle (body–neck angle)	Robot joints
δ	Tilt 2 angle (neck–head angle)	Robot joints
ε	Head's pan angle (neck–head angle)	Robot joints
δ_P	Prototype head tilt angle in the matched image	Prototype angle
ε_P	Prototype head pan angle in the matched image	Prototype angle
l_1	Neck's longitude (distance between the two rotation centers of the neck)	Robot geometry
l_2	Head's longitude (distance between neck-head rotation center and the camera)	Robot geometry
P	3D position of the observing robot body–neck rotation center measured from the global reference system	Self-localization
C	3D position of the observing robot's camera measured from the global reference system. This point corresponds to the origin of reference system 3.	Self-localization
R	3D position of the observed robot head measured from the observing robot camera. Measured in reference system 3.	To be computed
μ	Horizontal angle of the straight line that joints the two robot-heads, measured from the "3" reference system (see figure 4 (b)).	To be computed
ν	Vertical angle of the straight line that joints the two robot-heads, measured from the "3" reference system (see figure 4 (b)).	To be computed
ϕ	Affine transformation associated rotation. Computed using the mean of the SIFT angle differences in all the keypoints matches used to compute the transformation (named previously *rot*).	Robot head-pose determination system

These equations can be translated to global coordinates (x_0, y_0, z_0) using coordinate transformations. The intersection with the body will correspond, in global coordinates, to: $z_0(\lambda) = 0$. Then, going from RF "5" to RF "0" is given by:

$$(x_0 \quad y_0 \quad z_0 \quad 1)^T = M_{10} M_{21} M_{32} M_{43} M_{54} (x_5 \quad y_5 \quad z_5 \quad 1)^T$$

with:

$$M_{10} = \begin{pmatrix} \cos\alpha\cos\beta & -\sin\alpha & \cos\alpha\sin\beta & P_x \\ \sin\alpha\cos\beta & \cos\alpha & \sin\alpha\sin\beta & P_y \\ -\sin\beta & 0 & \cos\beta & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix}; \quad M_{21} = \begin{pmatrix} -\sin\gamma & 0 & -\cos\gamma & -l_1\sin\gamma \\ 0 & 1 & 0 & 0 \\ \cos\gamma & 0 & -\sin\gamma & l_1\cos\gamma \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{32} = \begin{pmatrix} \sin\delta & 0 & \cos\delta & l_2\sin\delta \\ \cos\delta\sin\varepsilon & \cos\varepsilon & -\sin\delta\sin\varepsilon & l_2\cos\delta\sin\varepsilon \\ -\cos\delta\cos\varepsilon & \sin\varepsilon & \sin\delta\cos\varepsilon & -l_2\cos\delta\sin\varepsilon \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{43} = \begin{pmatrix} \cos\mu\cos\nu & -\sin\mu & -\cos\mu\sin\nu & Rx_3 \\ \sin\mu\cos\nu & \cos\mu & -\sin\mu\sin\nu & Ry_3 \\ \sin\nu & 0 & \cos\nu & Rz_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad M_{54} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A robot-head in the image is characterized by an affine transformation that maps the prototype image onto a certain portion of this image. The real distance between the two robot heads is calculated as follows: the prototype head's image has four vertex: A, B, C, D and was taken with a distance ρ (when the picture was originally taken). The affine transformation maps this image into a rhomboid with vertex A', B', C' and D'. As the visual area decreases in a quadratic way with the distance, if the camera has no distortion, the Rx_3 coordinate of the observed robot's head in the $\{\hat{i}_3\ \hat{j}_3\ \hat{k}_3\}$ axis system can be calculated as:

$$Rx_3 = \rho\sqrt{\frac{prototype\ image\ area}{mapped\ area}} = \rho\sqrt{\frac{d(AB)\times d(BC)}{\det(\overrightarrow{A'B'}\quad \overrightarrow{B'C'})}}$$

Where ρ is the distance between the camera and the prototype head (at the acquisition time). If the horizontal angle of view of the camera is *Wu* and the vertical one is *Wv*, the camera resolution is *Mu* (horizontal) x *Mv* (vertical), and the head image's center is in the image position (u,v), then $Ry_3$and Rz_3 can be calculated as:

$$\mu = Wu\frac{Mu/2-u}{Mu},\ \nu = Wv\frac{Mv/2-v}{Mv};\ Ry_3 = Rx_3 tg^{-1}(\mu),\ Rz_3 = Rx_3 tg^{-1}(\nu).$$

4 Experimental Methodology and Results

The critical part of the here-proposed gaze determination system is the detection of the robot-heads. Therefore in this article results of this sub system are reported. In a future work we are going to report experimental results of the whole system.

The robot-head detection system was implemented in the AIBO ERS-7. The subsampled scale-space is built from the original AIBO images (208x160 pixels). Using these small images speeds up the calculations, but the use of a small initial Gaussian (σ=0.7) for building the scale-space makes the computation of interest points very noisy. This is the reason why the here-proposed parabolic interpolation and additional verification stages must be used. The SIFT points and descriptors calculation takes between 1.05 seconds and 1.25 seconds, depending on the number of objects under observation. The matching voting and transformation calculation takes around 30 milliseconds for each prototype head analyzed.

Robot-head detection experiments using real-world images were performed. In all of these experiments the 16 prototypes of robot player 1 were employed (see figure 2). These prototypes (around 100x100 pixels) are stored in the flash memory as BMP files. A database consisting on 39 test images taken on a four-legged soccer field was built. In these images robot "1" appears 25 times, and other robots appear 9 times. 10 images contained no robot. Currently this database is been expanded to be made public, together with the robot prototypes database. In table 2 are summarized the obtained results. If we consider full detections, in which both, the robot-head pose as well as the robot identity is detected, a detection rate of 68% is obtained. When we considered partial detections, i.e. only the robot identity is determined, a detection rate of 12% is obtained. The combined detection rate is 80%. At the same the number of false positives is very low, just 6 in 39 images. These figures are very good, because when processing video sequences, the opponent or teammates robots are seen in several consecutive frames. Therefore, a detection rate of 80% in single images should be high enough for detecting the robot-head in few frames.

Although more intensive experiments should be performed for characterizing our system, we believe that these preliminary experiments show the high potential of the proposed methodology, as a way of achieving player recognition and gaze estimation. The SIFT descriptors are not based in color information; therefore they are complementary to existing vision systems employed in the RoboCup leagues. A mixed SIFT and color-based vision system could be employed in the four-legged league if the SIFT computation time could shortened.

Table 2. Robot-head detection of robot #1 (only robot #1 prototype were employed)

Full detections (head + identifier number)	17/25	68%
Partial detections (only the identifier number)	3/25	12%
Full + partial detections	20/25	80%
Number of false detections in 39 images		6

References

1. D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *Int. Journal of Computer Vision*, 60 (2): 91-110, Nov. 2004.
2. M. Brown and D. G. Lowe, Invariant Features from Interest Point Groups, *British Machine Vision Conference* - BMVC 2002, 656 – 665, Cardiff, Wales, Sept. 2002.
3. V. Bakic, G. Stockman, Real-time Tracking of Face Features and Gaze Direction Determination, *4th IEEE Workshop on Applications of Computer Vision* – WACV'98, 256 – 257, Princeton, USA, Oct. 19 - 21, 1998.
4. A. Perez, M.L. Cordoba, A. Garcia, R. Mendez, M.L. Munoz, J.L. Pedraza, F. Sanchez, A Precise Eye-Gaze Detection and Tracking System, *11th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision* - WSCG'2003, Plzen - Bory, Czech Republic 3-7 February 2003.
5. C. Harris and M. Stephens, A combined corner and edge detector, *4th Alvey Vision Conf.*, 147-151, Manchester, UK, 1988.
6. F. Schaffalitzky and A. Zisserman, Automated location matching in movies, *Computer Vision and Image Understanding* Vol. 92, Issue 2-3, 236 – 264, Nov./Dec. 2003.
7. K. Mikolajczyk and C. Schmid, Scale & Affine Invariant Interest Point Detectors, *Int. Journal of Computer Vision*, 60 (1): 63 - 96, Oct. 2004.
8. Q. Ji and X. Yang, Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance, *Real-Time Imaging*, 8, 357-377 (2002).
9. T. Ohno, N. Mukawa and A. Yoshikawa, FreeGaze: A Gaze Tracking System for Everyday Gaze Interaction, *Symposium on Eye Tracking Research and Applications*, 125-132, 2002.
10. J. Beis and D.G. Lowe, Shape Indexing Using Approximate Nearest-Neighbor Search in High-Dimensional Spaces, *IEEE Conf. Comp. Vision Patt. Recog,* 1000-1006, 1997.
11. D.G. Lowe, Local Features View Clustering for 3D Object Recognition, *Proc. of the IEEE Conf. on Comp. Vision and Patt. Recog.*, 682 – 688, Hawai, Dic. 2001.

Simultaneous Learning to Acquire Competitive Behaviors in Multi-agent System Based on Modular Learning System

Yasutake Takahashi[1,2], Kazuhiro Edazawa[1], and Kentarou Noma[1],
and Minoru Asada[1,2]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering,
[2] Handai Frontier Research Center,
Osaka University
Yamadagaoka 2-1, Suita, Osaka, 565-0871, Japan
{yasutake, eda, noma, asada}@er.ams.eng.osaka-u.ac.jp
http://www.er.ams.eng.osaka-u.ac.jp

Abstract. The existing reinforcement learning approaches have been suffering from the policy alternation of others in multiagent dynamic environments. A typical example is a case of RoboCup competitions since other agent behaviors may cause sudden changes in state transition probabilities of which constancy is needed for the learning to converge. The keys for simultaneous learning to acquire competitive behaviors in such an environment are

- a modular learning system for adaptation to the policy alternation of others, and
- an introduction of macro actions for simultaneous learning to reduce the search space.

This paper presents a method of modular learning in a multiagent environment, by which the learning agents can simultaneously learn their behaviors and adapt themselves to the situations as consequences of the others' behaviors.

1 Introduction

There have been an increasing number of approaches to robot behavior acquisition based on reinforcement learning methods [1, 4]. The conventional approaches need an assumption that the state transition is caused by an action of a learning agent so that the learning agent can regard the state transition probabilities as constant during its learning. Therefore, it seems difficult to directly apply the reinforcement learning method to a multiagent system because a policy alteration of other agents may occur, which dynamically changes the state transition probabilities from the viewpoint of the learning agent. RoboCup provides such a typical situation, that is, a highly dynamic, hostile environment, in which agents must obtain purposive behaviors.

There are a number of studies on reinforcement learning systems in a multiagent environment. Kuhlmann and Stone [8] have applied a reinforcement learning

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 243–254, 2006.

system with function approximater to the keep-away problem in the situation of RoboCup simulation league. In their work, only the passer learns his policy to keep the ball away from the opponents. The other agents, receivers and opponents, follow fixed policies given by the designer beforehand.

Asada et al. [2] proposed a method that sets a global learning schedule in which only one agent is specified as a learner and the rest of agents have fixed policies. Therefore, the method cannot handle the alternation of the opponent's policies. Ikenoue et al. [3] showed simultaneous cooperative behavior acquisition by fixing learners' policies for a certain period during the learning. These studies suggest that it is enable to acquire a reasonable behavior in a multi-agent environment if the learner can see the environment including the other agents almost fixed because the others keep their policies for a certain time. In the case of cooperative behavior acquisition, both agents do not have any reason to change their policies while they successfully acquire positive rewards with the result of the cooperative behavior for each other. The agents tend to update their policies gradually so that the state transition probabilities seem almost fixed from the view point of the other learning agents.

In case of competitive behavior acquisition in a multiagent environment, however, it is unlikely that the agent tends to select the action that causes positive rewards for the opponents and a negative reward for itself. The punished agent tends to change drastically its policy so that it can acquire a positive reward by which it gives a negative reward to the opponents. This policy alternation causes dynamic changes in the state transition probabilities from the viewpoint of the learning agent therefore it seems difficult to directly apply the reinforcement learning method to a multiagent system.

A modular learning approach would provide one solution to this problem. If we can assign multiple learning modules to different situations, respectively, in each of which the state transition probabilities can be regarded as constant, then the system could show a reasonable performance. Jacobs and Jordan [7] proposed the mixture of experts, in which a set of the expert modules learn and the gating system weights the output of the each expert module for the final system output. This idea is very general and has a wide range of applications (ex. [11, 9, 14, 13, 5]).

We adopt the basic idea of the mixture of experts into architecture of behavior acquisition in the multi-agent environment. Takahashi et al. [12] have shown preliminary experimental results of behavior acquisition in the multi-agent environment, however, the learning modules were assigned by the human designer. In this paper, first, we show how it is difficult to directly introduce a multi-module learning system for even single agent learning in a multi-agent environment because of its complexity and introduce a simple learning scheduling which makes it relatively easy to assign modules automatically. Second, we introduce macro actions to realize simultaneous learning in multiagent environment by which the each agent does not need to fix its policy according to some learning schedule. Elfwing et al. [6] introduced macro actions to acquire a cooperative behavior with two real rodent robots. The exploration space with macro actions becomes much smaller than the one with primitive actions, then, the macro action increases the possibility to meet cooperative experiences and leads the two agents to find a reasonable solution in realistic learning time. We show the introduction of macro actions enable the agents to learn competitive behaviors simultaneously. We have applied the proposed multi-module learning system to soccer robots which participate in RoboCup competition and show experimental results on computer simulation and real robot implementation.

2 Tasks and Assumption

Fig.1 shows a situation which the learning agents are supposed to encounter. The game is like a three on one; there are one opponent and other three players. The player nearest to a ball becomes to a passer and passes the ball to one of the teammates (receivers) while the opponent tries to intercept it.

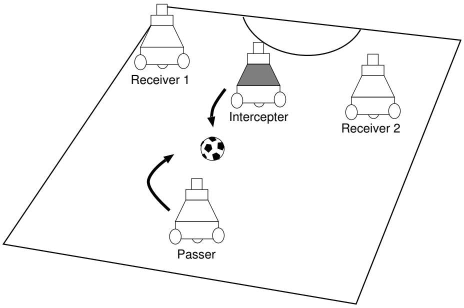


Fig. 1. Task : 3 on 1

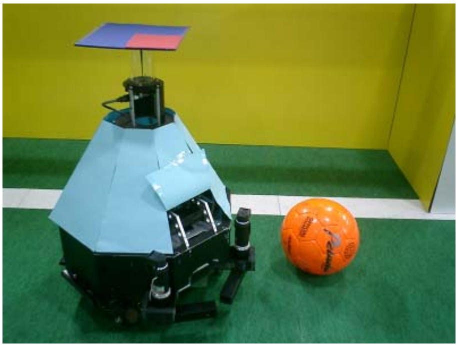


Fig. 2. A real robot

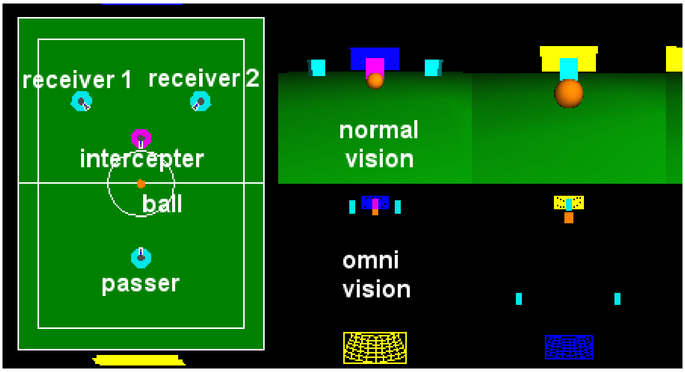


Fig. 3. Viewer of simulator

Fig. 2 shows a mobile robot we have designed and built. Fig. 3 shows the viewer of our simulator for our robots and the environment. The robot has an omni-directional camera system. A simple color image processing is applied to detect the ball, the interceptor, and the receivers on the image in real-time (every 33ms). The left of Fig. 3 shows a situation in which the agent can encounter and the right images show the simulated ones of the normal and omni vision systems. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane).

3 Multi-module Learning System

3.1 Basic Idea

The basic idea is that the learning agent could assign one behavior learning module to one situation which reflects other agent's behavior and the learning module would acquire a purposive behavior under the situation if the agent can distinguish a number of situations in each of which the state transition probabilities are almost constant. We introduce a modular learning approach to realize this idea (Fig. 4). A module consists of a learning component that models the world and an action planner. The whole system follows these procedures:

- select a model which represents the best estimation among the modules,
- update the model, and
- calculate action values to accomplish a given task based on dynamic programming.

As an experimental task, we suppose ball passing with possibility of being intercepted by the opponent (Figs. 1 and 3). The problem for the passer (interceptor) here is to select one model which can most accurately describe the interceptor's (passer's) behavior from the viewpoint of the agent and to take an action based on the policy which is planned with the estimated model.

3.2 Architecture

Fig. 5 shows a basic architecture of the proposed system, that is, a multi-module reinforcement learning system. Each module has a forward model (predictor) which represents the state transition model, and a behavior learner (action planner) which estimates the state-action value function based on the forward model

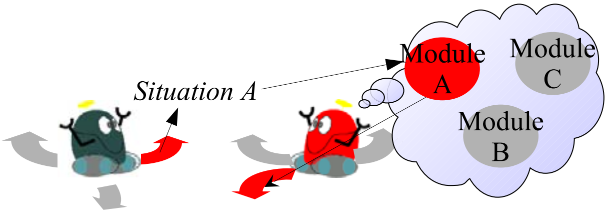


Fig. 4. Adaptive behavior selection based on Multi-module learning system

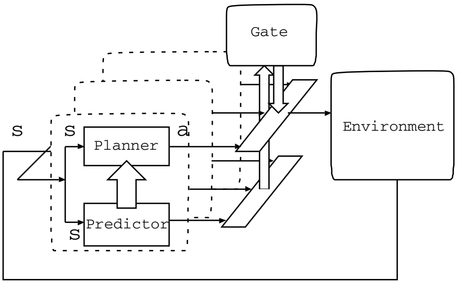


Fig. 5. A multi-module learning system

in a reinforcement learning manner. This idea of combination of a forward model and a reinforcement learning system is similar to the H-DYNA architecture [10] or MOSAIC [5]. The system selects one module which has the best estimation of a state transition sequence by activating a gate signal corresponding to a module while deactivating the gate signals of other modules, and the selected module sends action commands based on its policy.

Predictor. Each learning module has its own state transition model. This model estimates the state transition probability $\hat{\mathcal{P}}^a_{ss'}$ for the triplet of state s, action a, and next state s':

$$\hat{\mathcal{P}}^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{1}$$

Each module has a reward model $\hat{\mathcal{R}}^a_{ss'}$, too:

$$\hat{\mathcal{R}}^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \tag{2}$$

We simply store all experiences (sequences of state-action-next state and reward) to estimate these models.

Planner. Now we have the estimated state transition probabilities $\hat{\mathcal{P}}^a_{ss'}$ and the expected rewards $\hat{\mathcal{R}}^a_{ss'}$, then, an approximated state-action value function $Q(s,a)$ for a state action pair s and a is given by

$$Q(s,a) = \sum_{s'} \hat{\mathcal{P}}^a_{ss'} \left[\hat{\mathcal{R}}^a_{ss'} + \gamma \max_{a'} Q(s',a') \right] , \tag{3}$$

where γ is a discount rate.

Module Selection for Action Selection. The reliability of the module becomes larger if the module does better state transition prediction during a certain period, else it becomes smaller. We assume that the module which does the best state transition prediction has the best policy against the current situation because the planner of the module is based on the model which describes the situation best. In the proposed architecture, the reliability is used for gating the action outputs from modules. We calculate an execution-time reliability ${}^{exec}g_i$ of the module i as follows:

$${}^{exec}g_i = \prod_{t=-T+1}^{0} e^{\lambda p_i^t}$$

where p_i is an occurrence probability of the state transition from the previous $(t-1)$ state to the current (t) one according to the model i, and λ is a scaling factor. The T indicate a period (step) to evaluate the reliability of the module and we set T as 5 in the following experiments. The agent continues to use the module for a certain period, for example 5 step or 1 second, after it changes the module in order to avoid oscillation of the policies.

Module Selection for Updating Models. We use an update-time reliability ${}^{update}g_i$ of the module for updating modules. The calculation of this reliability contains the future state transition probabilities:

$${}^{update}g_i = \prod_{t=t-T}^{t+T} e^{\lambda p_i^t}.$$

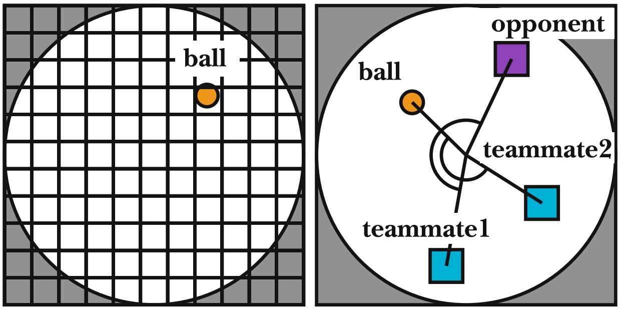
4 Behaviors Acquisition Under Scheduling

As we mentioned in 1, first, we show how it is difficult to directly introduce the proposed multi-module learning system in the multi-agent system. We introduce a simple learning scheduling in order to make it relatively easy to assign modules automatically.

4.1 Configuration

The state space is constructed in terms of the centroid of the ball on the image, the angle between the ball and the interceptor, and the angles between the ball and the receivers (see Figs. 10 (a) and (b)). We quantized the ball position space 11 by 11 as shown in Fig. 10 (a) and the each angle into 8. As a result, the number of states becomes $11^2 \times 8 \times 8 \times 8 = 61952$. The action space is constructed in terms of desired three velocity values (x_d, y_d, w_d) to be sent to the motor controller (Fig. 7). Each value is quantized into three, then the number of action is $3^3 = 27$. The robot has a pinball like kick device, and it automatically kicks the ball whenever the ball comes to the region to be kicked. It tries to estimate the mapping from sensory information to appropriate motor commands by the proposed method.

The initial positions of the ball, the passer, the interceptor, and receivers are shown in Fig. 1. The opponent has two kinds of behaviors; it defends the left side, or right side. The passer agent has to estimate which direction the interceptor will defend and go to the position in order to kick the ball to the direction the interceptor does not defend. From a viewpoint of the multi-module learning system, the passer will estimate which situation of the module is going on, select the most appropriate module to behave. The passer acquires a positive reward when it approach to the ball and kicks it to one of the receivers.



(a) state variables (position) (b) state variables (angle)

Fig. 6. State variables

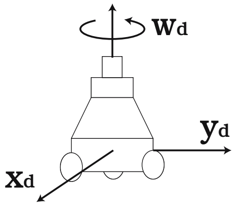


Fig. 7. Action variables

4.2 Learning Scheduling

We prepare a learning schedule composed of three stages to show its validity. The opponent fixes its defending policy as right side block at the first stage. After 250 trials, the opponent changes the policy to block the left side at the second stage and continues this for another 250 trials. Then, the opponent changes the defending policy randomly after one trial.

4.3 Simulation Result

We have applied the method to a learning agent and compared it with only one learning module. We have also compared the performances between the methods with and without the learning scheduling. Fig. 8 shows the success rates of those during the learning. The success indicates that the learning agent successfully kick the ball without interception by the opponent. The success rate indicates the number of successes in the last 50 trials. The "mono. module" in the figure indicates "monolithic module" system and it tries to acquire a behavior for both policies of the opponent. The multi-module system with scheduling shows a better performance than the one-module system. The monolithic module with scheduling means that we applied learning scheduling mentioned in **4.2** even though the system has only one learning module. The performance of this system is similar with multi-module system until the end of first stage (250 trials), however, it goes down at the second stage because the obtained policy is biased against the experiences at the fist stage and cannot follow the policy change of the opponent. Since the opponent takes one of the policies at random at the third stage, the learning agent obtains about 50% of success rate. "without scheduling" means that we do not applied learning scheduling and the opponent changes its policy at random from the start. Somehow the performance of the monolithic module system without learning scheduling is getting worse after the 200 trials. The multi-module system without learning schedule shows the worst performance in our experiments. This result indicates that it is very difficult to recognize the situation at the early stage of the learning because the modules

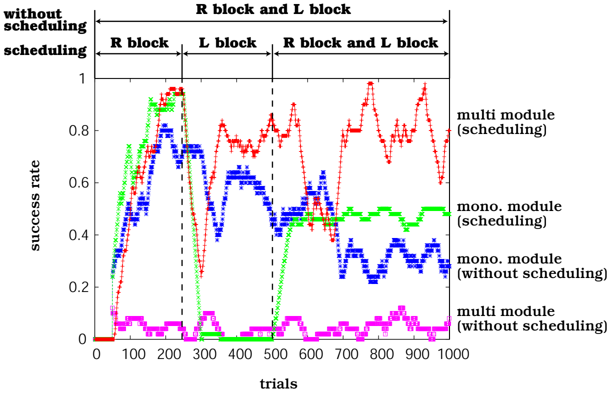


Fig. 8. Success rate during the learning

has too few experiences to evaluate their fitness, then the system tends to select the module without any consistency. As a result, the system cannot acquire any valid policies at all.

5 Simultaneous Learning with Macro Actions

We introduce macro actions to realize simultaneous learning in multiagent environment by which the each agent does not need to fix its policy according to some learning schedule. In this experiment, the passer and the interceptor learn their behaviors simultaneously. The passer learns behaviors for different situations which are caused by the alternation of the interceptor's policies, that is, blocking left side or right one. The interceptor also learns behaviors for different situations which are caused by the alternation of the passer's policies, that is, passing a ball to left receiver or right one.

5.1 Macro Actions and State Spaces

Fig. 9 shows the macro actions of the passer and the interceptor. The macro actions for the interceptor are blocking the pass way to the left receiver and right one. On the other hand, the macro action for the passer are turning left, turning right around the ball, and approaching to the ball to kick it. A ball gazing control is embedded to the both learners. The number of the actions reduced from 27 (see 4.1) primitives to 2 or 3 macro actions. The state space for the passer is constructed in terms of the y position of the ball on the normal image, and the angle between the ball and the centers of interceptor, the ones between the balls and the two receivers on the image of omni-directional vision. The number of the states reduced from 61952 (see 4.1) to 3773 because the set of macro actions enable us to select smaller number of state variables and coarser quantization. The state space for the interceptor is constructed in terms of the y position of the passer on the image of normal vision system, and the

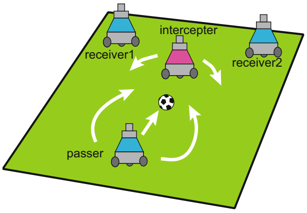
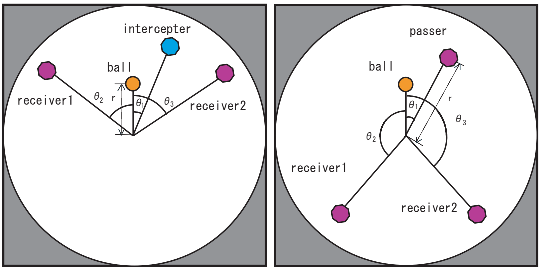


Fig. 9. Macro actions



(a) passer (b) interceptor

Fig. 10. State variables

angle between the ball and the passer and the ones between the ball and the two receivers on the image of omni-directional vision. The number of the states is 2695.

5.2 Experimental Results

We have checked how the simultaneous learning of the passer and interceptor works on our computer simulation. Both agents start to learn their behaviors from scratch and have 1500 trials without any scheduling. Fig. 11 show the success rates during the simultaneous learning of the passer and the interceptor. This figure shows that the interceptor has higher success rate at the beginning of learning, the passer is getting to acquire the appropriate behaviors corresponding to the interceptor's behaviors, and the both agents have almost equal success rate at the end of learning stage. The sum of the both success rates is not 1 because the both player sometimes failed to pass or intercept simultaneously.

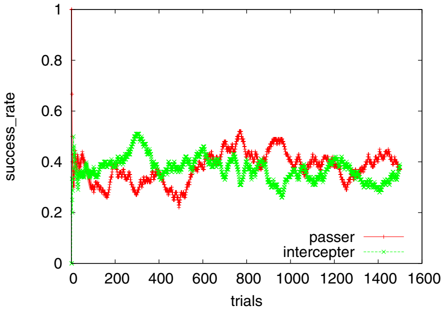


Fig. 11. Sequence of success rates during simultaneous learning

In order to check if the both learners acquire appropriate behaviors against the opponent's behaviors, we fixed one agent's policy and check that the other can select an appropriate behavior and its success rate. Table 1 shows these results. The both players have two modules and assign them to appropriate

Table 1. Success rates for passer and receiver in different cases

passer	interceptor	passer's success rate[%]	interceptor's success rate [%]	draw rate[%]
LM0,LM1	LM0	59.0	23.0	18.0
LM0,LM1	LM1	52.7	34.3	13.0
LM0	LM0,LM1	25.6	55.0	19.4
LM1	LM0,LM1	26.0	59.3	14.7
LM0,LM1	LM,LM1	37.6	37.3	25.1

situations by themselves. LM and the digit number right after the LM indicate Learning Module and index number of the module, respectively. For example, the passer uses both LM0 and LM1 and the interceptor use only LM0, then the passer's success rate, interceptor's success rate, and draw rate are 59.0 %, 23.0%, and 18.0%, respectively. Apparently, the player switching multi-modules achieves higher success rate than the opponent using only one module. These results show that the multi-module learning system works well for both.

We have applied the same architecture to the real robots. Fig. 12 shows the one example behaviors by real robots. First, the interceptor tried to block the left side, then the passer approached the ball with intention to pass it to the right receiver. The interceptor found that it tried to block the wrong side and change to block the other side (right side), but, it is too late to intercept the ball and the passer successfully pass the ball to the right receiver.

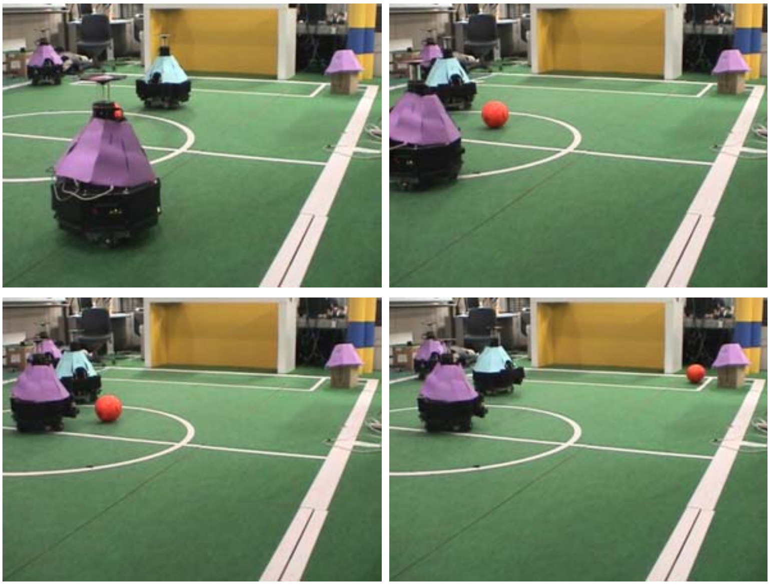


Fig. 12. A sequence of a behavior of passing a ball to the right receiver while the interceptor blocks the left side

6 Conclusion

In this paper, we proposed a method by which multiple modules are assigned to different situations which are caused by the alternation of the other agent policy and learn purposive behaviors for the specified situations as consequences of the other agent's behaviors.

We introduced macro actions to realize simultaneous learning of competitive behaviors in a multi-agent system. We have shown results of a soccer situation and the importance of the learning scheduling in case of none-simultaneous learning without macro actions and the validity of the macro actions in case of simultaneous learning in the multi-agent system.

References

1. M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.
2. M. Asada, E. Uchibe, and K. Hosoda. Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development. *Artificial Intelligence*, 110:275–292, 1999.
3. Shoichi Ikenoue Minoru Asada and Koh Hosoda. Cooperative behavior acquisition by asynchronous policy renewal that enables simultaneous learning in multiagent environment. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pages 2728–2734, 2002.
4. Jonalthan H. Connell and Sridhar Mahadevan. *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.
5. Kenji Doya, Kazuyuki Samejima, Ken ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. Technical report, Kawato Dynamic Brain Project Technical Report, KDB-TR-08, Japan Science and Technology Corporatio, June 2000.
6. Stefan Elfwing, Eiji Uchibe, Kenji Doya, and Henrik I. Christensen1. Multi-agent reinforcement learning: Using macro actions to learn a mating task. In *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages CD–ROM, Sep 2004.
7. R. Jacobs, M. Jordan, Nowlan S, and G. Hinton. Adaptive mixture of local experts. *Neural Computation*, (3):79–87, 1991.
8. Gregory Kuhlmann and Peter Stone. Progress in learning 3 vs. 2 keepaway. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.
9. Satinder P. Singh. The effeicient learnig of multiple task sequences. In *Neural Information Processing Systems 4*, pages 251–258, 1992.
10. Satinder P. Singh. Reinforcement learning with a hierarchy of abstract models. In *National Conference on Artificial Intelligence*, pages 202–207, 1992.
11. Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8:323–339, 1992.
12. Yasutake Takahashi, Kazuhiro Edazawa, and Minoru Asada. Multi-module learning system for behavior acquisition in multi-agent environment. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages CD–ROM 927–931, October 2002.

13. J. Tani and S. Nolfi. Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach. Technical report, Sony CSL Technical Report, SCSL-TR-97-008, 1997.
14. Jun Tani and Stefano Nolfi. Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach. Technical report, Technical Report: SCSL-TR-97-008, 1997.

A HMI Supporting Adjustable Autonomy of Rescue Robots

Andreas Birk and Max Pfingsthorn

School of Engineering and Science
International University Bremen
Campus Ring 1, D-28759 Bremen, Germany
a.birk@iu-bremen.de

Abstract. Human rescue workers are a scarce resource at disaster sites. But it is still a long way to go before fully autonomous rescue robots will be fieldable. The usefulness of rescue robots will hence strongly depend on the availability of user interfaces that enable a single first responder to operate a whole set of robots. For this challenge, it is important to preprocess and streamline the immense data flow from the robots and to assist the operator as much as possible in the processes of controlling the robots. This paper introduces an adaptive graphical interface supporting adjustable autonomy of rescue robots. The design is based on insights from the literature in this field where intensive surveys of the actual needs in this domain were compiled.

1 Introduction

Rescue robots have a large application potential as demonstrated for the first time on a larger scale in the efforts at the World Trade Center after the 9/11 event [Sny01]. For an overview of potential tasks of rescue robots and the related research in general see for example [RMH01]. One of the main challenges in using robots in search and rescue missions is to find a good trade-off between completely remotely operated devices and full autonomy. The complexity of search and rescue operations makes it difficult if not impossible to use fully autonomous devices. On the other hand, the amount of data and the drawbacks of limited communication possibilities make it undesirable if not unfeasible to put the full control of the robot into the hands of a human operator.

The goal of the IUB rescue robots team is to develop fieldable systems within the next years. Since the beginning of its research activities in this field in 2001, the team has participated in several RoboCup competitions to test its approaches [Bir05, BCK04, BKR^{+}02]. In addition to work on mapping [CB05] and adhoc-networking [RB05], the development of the robots themselves based on the so-called CubeSystem [Bir04] is an area of research in the team [BKP03, BK03].

As mentioned, one of the the main challenges for using rescue robots is to find a good tradeoff between completely remotely operated devices and full autonomy [BK03]. Ideally, a single rescue worker supervises a multitude of semi-autonomous robots that provide only the most crucial data to an operators

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 255–266, 2006.

station. Here, we present an approach to this challenge in form of an adaptive graphical human machine interface (HMI) supporting adjustable autonomy.

Adjustable Autonomy (AA) in general addresses the issue that, while autonomous agents get more and more powerful, it is still impossible for a real application to exclude the human operator from the loop of operations and retain the same effectiveness. Using Adjustable Autonomy, an agent can behave autonomously and dynamically change its level of independence, intelligence and controlfreely placeable. The concept of Adjustable Autonomy is very broad and does not only encompass robotics. It has been applied to many fields, such as office automation [SPT03], desktop software [Mim98], military [FCGB00], and space exploration [SBA$^+$03].

A way to achieve Adjustable Autonomy is to support multiple levels of autonomy of the agent, e.g. fully autonomous, guided by the operator, and fully controlled by the operator [BDM02, GJCP01]. It is also conceivable to support a continuous range of autonomy levels, e.g. by employing a continuous measure of neglect and problem priorities [OG03].

For example, an ideal situation in Adjustable Autonomy would be the following: A rescuer at a disaster site employs the help of a team of robots to survey a part of a collapsed building. To be more effective, the rescuer commands the robots to spread out and to explore, detecting and locating victims, but also fire, etc. While the robots search the building, they may run into problems, like getting stuck on stairs, not knowing how to avoid an obstacle best, etc. All those problems are forwarded to the rescuer. That is the only time when the rescuer really has to take care of individual robots. Once a robot has found an interesting set of features indicating an object or event, the notice is also sent to the rescuer and marked in a collaborative map. All sensor data of the respective robot is made available to the rescuer to verify the findings of the robot. Once victims are found and a safe path to them has been identified, rescue crews are deployed to extract the victims from the building.

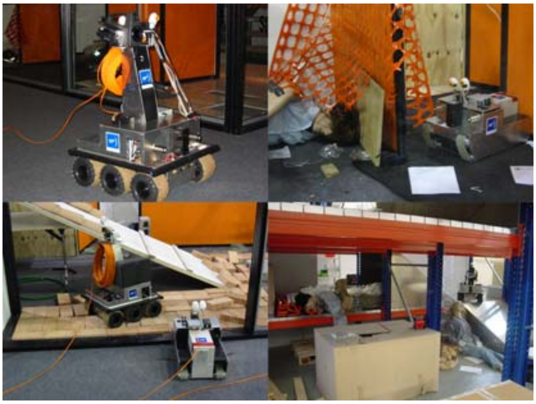


Fig. 1. The IUB rescue robots at RoboCup

Previous approaches to a Graphical User Interface (GUI) for Adjustable Autonomy encompass a wide variety of settings and intentions. Most interesting for our interests is the investigation of designs by Goodrich and Olsen [GJCP01], Fong et al [FCTB01, FTB01, FCGB00, FTB02, FT01], Backes et al [BNP$^+$] and Bruemmer et al [BDM02]. In addition, evaluations in terms of usability are highly important as for example done by Olsen and Goodrich [OG03] and Scholtz et al [SYDY04].

The approaches taken in the aforementioned papers roughly fall into two classes. In the first one, a focus is made on direct control and portability of hardware. In the second, task planning and data visualization is emphasized. A general trend in this line of research is visible: Most rely on one main mode of direct visualization of the robot's state, either through a map generated by the robot or a direct video stream. This coarse form of data visualization is especially useful for gaining a quick understanding of the robot's situation, referred to as *Context Acquisition* [OG03]. In addition, the presented interfaces offer multiple ways of commanding the robot: through haptic and gesture manipulation [FCGB00], direct vector input and choosing a location on a map [FCTB01], and pure high-level task planning [BNP$^+$]. This process is referred to as *Expression* [OG03].

In an attempt to combine the best of both, i.e., direct interaction with the robot and data visualization, we attempt to merge both paradigms: to employ intuitive data visualization and direct intuitive control methods in order to optimize *Context Acquisition* and *Expression* time, exactly those deemed most costly by Olsen and Goodrich [OG03].

2 The Design Goals and Core Components

The main application target for the system are real-time, multi-robot tasks. While the developed interface framework will allow for various uses, main design arguments will be geared to the application in real-time environments. This includes a bigger data visualization area and rather small control areas, since most control is done via peripheral input devices.

Murphy and Rogers [MR96] pointed out inefficiencies in traditional teleoperation interfaces that are supposed to be corrected with this new interface framework. Those deficiencies include constant high-bandwidth demands, cognitive fatigue due to too many tasks done at once, and poor displays. It is mentioned that conservative teleoperation interfaces are inefficient because the operator usually only handles one robot and that reduces work efficiency by a factor of five to eight. Further important design guidelines were derived from the work of Olsen and Goodrich [OG03] and of Scholtz et al [SYDY04], which is based on an evaluation of existing human-robot interfaces.

Olsen and Goodrich hypothesize that the process of human-robot interaction can be broken down into four parts: *Subtask Selection* (picking a robot to service from a list), *Context Acquisition* (perceiving the robot's current state and problem), *Planning* (thinking of a way to solve the robot's problem), and *Expression* (stating the plan in robot oriented commands). Scholtz et al state from observations made during the World Championship Competition in

RoboCup Robot Rescue 2004 in Padova, Italy, that successful interfaces should include a representation for the global environment, display of the robot's current state, integrated display of sensor data, ability for self-diagnosis of the robot, and contex-sensitive information display.

Based on these guidelines, we developed an interface that is divided into three blocks, the Canvas, the Sidebar, and the Visibility Controller. Each core component is described in detail in the following.

2.1 The Canvas

The Canvas is a drawing space for 3D representations of the world, using OpenGL. Here, usually a map would be drawn and populated with robots. Due to the extreme flexibility of the design, almost any kind of visualization is possible here. Even 2D displays, like status bars for all connected robots or video displays are possible.

In the context of the above mentioned criteria, the Canvas represents a common ground for sensor data visualization. Therefore, a facility has been established to support easy fusing of sensor data by the interface itself. For example, both the map and the display of the robot and its state can be separate modules both drawing themselves to this canvas. As a result, the data is merged automatically, making it easier for the user to perceive the robot's current situation.

In addition, a camera implemented by the Canvas can be moved around in an intuitive fashion (very much like a First Person game interface) to allow viewing the depicted 3D environment from all angles. In addition, the camera can be "snapped" to the current robot's position and orientation (pose) such that it follows the movements of the robot. This gives rise to a separate viewing method, known from racing games which should especially come in handy when purely teleoperating the robot.

The Canvas enhances *Context Acquisition* as well as *Subtask Selection*, as all robots are shown and their state can be evaluated at the same time. In Scholtz's terms, the Canvas addresses the first three points.

2.2 The Sidebar

The Sidebar constitutes a place to display control elements. As space is limited on the screen, this part is kept to a minimum of size. This is because in general, control is exerted via peripheral input devices, such as mouse, joystick or joypad, and steering wheels. Only displaying their state is useful for even more direct user feedback, as mentioned above.

The Sidebar also allows for the grouping of control elements. This gives a clearer structure to the controlling side of the interface and thus decreases the cognitive load necessary for the *Expression* process, as mentioned above.

In addition, the Sidebar has a built-in collapse and expand function. Each group can be collapsed (hidden) or expanded (shown) by the request of the user. This way, the user can choose which kind of control is supposed to be used. This feature covers Scholtz's last point.

2.3 The Visibility Controller

The last and least visible part is the Visibility Controller. Almost completely hidden to the user, other than a simple drop-down list to choose a visibility setting, the Visibility Controller takes care of scaling, hiding and positioning single elements that are displayed. It can handle both control and display elements.

The Visibility Controller is designed to store a vector of importances of interface parts. When a certain mode of operation is assumed (compare [GJCP01, BDM02]), the Visibility Manager will set the importances of all elements. The elements then know how to adjust their own behavior, form, and shape to reflect that importance value.

This concept of a Visibility Controller clearly addresses Scholtz's last point and Olsen and Goodrich's concepts of *Context Acquisition* and *Expression*, since it requires less time to choose an appropriate control from an already smaller list.

2.4 The Dynamics Aspects

As briefly described in the previous section, the interface presented also consists of a so called Visibility Controller. This module manages importances of single elements shown on the screen. According to these importances, the single modules representing drawable items perform different actions.

For example, a control element can decide to hide itself and to stop producing output if its importance falls beneath a certain threshold level, while a display element might implement a form of scaling (e.g. for video displays) or fading (e.g. for less importance of the map). Also rearrangement is possible, for example in the case of video displays. As the importance of a video display increases, it can change the arrangement of multiple video streams it is displaying, e.g. shifting from a stack structure to a grid in order to distribute the available space in a better way.

As Goodrich and Olsen [GJCP01] and Fong et al [FCTB01] point out in their treatises, choosing different modes of autonomy is a highly desirable feature. Especially, not only does the robot behavior change, but the interface should reflect this change as well. This gives further motivation for the Visibility Controller as it further addresses the problem of *Context Acquisition*, i.e., the mode of operation can be inferred through the arrangement of the display [OG03] and specifying context-sensitive information [SYDY04].

While it is reasonable to assume that such importances are changed dynamically and automatically by choosing a mode of operation, this fact is quite limiting for the user. During the *Planning* phase, optimal use has to be made of the displayed information in order to create a plan how to circumvent the robot's current problem. This might require reading displays with a lower importance. Hence, it is important to leave a chance of changing importances of single displays and controls to the user both during missions planning as well as on the fly.

For the interface and design presented, it does not make a difference how the changing of importances is implemented. However, the tools used to implement the whole interface already provide adequate ways of doing so. For example, the

OpenGL standard [SB] defines four component colors, including an Alpha component, which would make it easy to "fade" a display on the Canvas according to some function of the importance. Also, OpenGL defines scaling functions in all three directions. The Qt Toolkit [Tro], used to develop the standard GUI components of the interface, already allows for dynamic hiding and resizing of components.

3 The Framework Structure and Its Implementation

3.1 Qt Toolkit

The Qt Toolkit by Trolltech [Tro] is a very easy-to-use, cross-platform GUI toolkit with some more support for cross-platform development, like threading, file handling and the like. Qt employs their own model of event-driven programming, namely "signals" and "slots". Those are dynamically callable interfaces to events such as "clicked" for buttons and "activated" for combo boxes.

Qt abstracts windowing operations very nicely and it is as easy to construct an application out of ready made components as to implement a very special-purpose "widget", i.e. a custom-made component of the GUI. Also, Qt offers a straight-forward interface to mouse events and even to OpenGL, so the Qt Toolkit was the first choice as the underlying windowing library for this project.

3.2 FAST Robots

As the interface has to rely on an infrastructure to be able to communicate with controlled robots, the HMI is based on the FAST Robots framework (Framework Architecture for Self-Controlled and Teleoperated Robots) [KCP$^+$03]. This middleware used for high-level sensor abstraction and communications employs a strong generalization for sending any type of serializable data over a network. Initially, the framework was designed for remote data collection for virtual experiments, but progressed to being a very extensible architecture for rapid prototyping in robotics.

In Figure 2 on the left, the basic structure of the architecture can be seen. Communications flow starts at the level of the sensor as it is queried in regular time intervals by the owning "experiment". The experiment reads the data and sends it on via its network link. On the other side of the network link, the corresponding "monitor" reads the information, and due to a stored ID number, forwards the data to the right "display". In the case of control data flow, the situation is reversed: The "control" is given data by the user and, due to the asynchronous nature of Graphical User Interfaces, queries the monitor to forward its data via its network link. The corresponding experiment receives the data and, employing a similar distribution scheme like the monitor did for sensor data, forwards the control data to the right "actuator".

The right side of figure 2 shows a depiction of the relationship between a sensor and a display implementation. Such a pairing shares another class: A NetworkData subclass implementing the specific way data from this sensor is

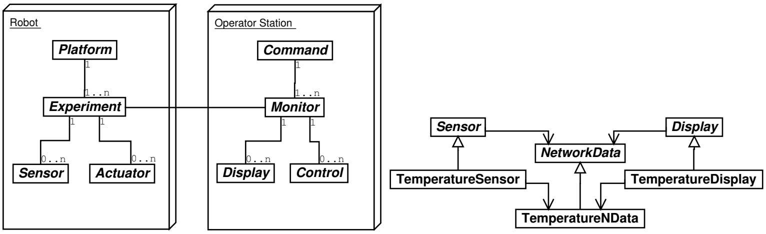


Fig. 2. The structure of FAST Robots (left) and the Sensor/Display/NetworkData triple (right)

serialized and deserialized to a form suitable for transportation over the network. It is conceivable to implement different NetworkData subclasses for a single type of sensor/display pair as different compression algorithms might be used, e.g. JPG, PNG or even Wavelet compression for pictures.

While the design of FAST Robots is very general and platform-independent, the current implementation heavily uses the Qt Toolkit mentioned above for various subtasks, such as threading, concurrency, data serialization and the GUI of the "Command". Lately, efforts have been made to port FAST Robots to other architectures and libraries.

3.3 Interface Framework

The interface presented focuses on the Controller side of the schematic shown in Figure 2 and will thus be called "FAST Commander". In simple words, a new front end for FAST Robots applications was to be developed. Hence, the new interface should be as flexible and as extensible as possible in order to be as versatile as FAST Robots.

The UML diagram shown in Figure 3 shows the structure of the design. There are four main components:

- The RobotList, which manages all Robots connected and ensures forwarding of data. Also, the current robot is chosen here and data coming from the current robot is forwarded through a special channel. If a certain type of data from a robot is important it is also forwarded while the robot is not the current robot. In addition, specific types of data can be muted, i.e. they do not get forwarded at all. Most importantly, the Robot maps robot-specific sensor IDs to application wide keys, e.g. from '1' to "video0" meaning that sensor 1 on this robot is the first video sensor.
- The GLViewCanvas, which presents all GLCanvasItems to the user. Those items are the main means of displaying sensor and other (e.g. mapping) data. The GLViewCanvas implements general necessities, such as a freely movable camera to change perspective of the display and processing mouse events, which are forwarded to the associated GLCanvasItems. In addition, the canvas also implements a sort of z-index for each GLCanvasItem, which

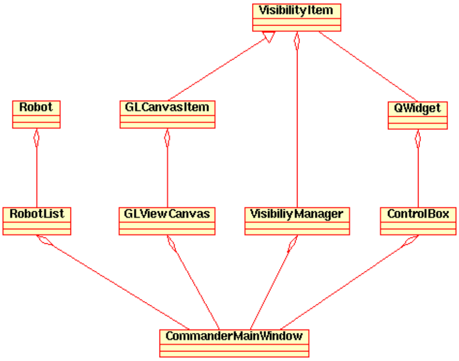


Fig. 3. The UML diagram of the framework provided

is used to sort the associated items. While mouse events are forwarded in reverse z-order (top-most item first), the drawing is done in z-order (bottom-most item first). Since sorting is only done during insertion, there is no additional cost and a more natural feel is achieved for the mouse interaction.

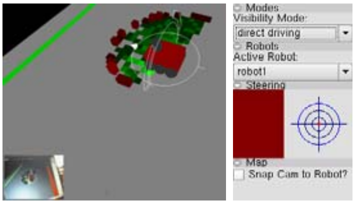
- The ControlBox (previously called Sidebar), which holds QWidgets (the most general GUI component class from the Qt Toolkit), is a container for control components. Control components are grouped and may be hidden (collapsed) or shown (expanded) as a group.
- The VisibilityManager, which implements dynamic aspects of the interface, as described in section 2.4. Implementation wise, the Visibility Manager does not only know what to do with VisibilityItem subclasses (i.e. how to set their importance) but also what to do with QWidgets (e.g. hide them if the importance drops beneath a certain threshold) and group names from the ControlBox (same as for QWidgets).

All these components are always present in any instantiation of the FAST Commander interface. Any additional components, e.g. a map canvas item, a joystick control, a connection to a specific robot and its mappings, are added at mission specification. This ensures high customizability and task independence.

3.4 The Basic Displays

Figure 4 shows a screenshot of the core displays:

- A Map Display, which draws a 3D map according to the robot's sensor input. Cells believed to be empty are drawn flat and in Green color, whereas cells believed to be occupied are drawn in red with a height proportional to the strength of the belief. Importance is implemented as a proportional fading.
- A Robot Display, which draws the current state and pose of a single robot in a sensible position relative to the map. The robot may assume a 3D pose and the display can also indicate if the robot is active. Additionally, two gauges exist that can indicate an overall health status.



Visibility Mode: direct driving

Active Robot: robot1

□ Snap Cam to Robot?

Fig. 4. Core displays shown in the GLViewCanvas (left) and controls grouped on an example sidebar (right)

- A Video Display, which draws video streams as a stack of video pictures starting from the lower left corner. Their size can vary dynamically and scaling is applied for both resizing of the GLViewCanvas and reflecting the importance level.

3.5 Core Controls

The first control is the Joystick control. Apart from standard controls such as a component allowing to choose a robot for activation, changing the visibility arrangement, and controlling the GLViewCanvas, specific controls are implemented that allow further access to the display's features, e.g., controlling the map. The following controls are present in the interface (see Figure 4 for a picture of an example sidebar):

- The VisibilityControl, which lets the user choose one of some preset arrangement of importances geared toward a specific mode of operation. In this example case, choices included "map-based driving" and "direct driving".
- The RobotListControl, which offers a choice of all connected robots to change focus to. Once a different robot is selected, all control messages are sent to that robot.
- The JoystickControl, which gives direct feedback of the read joystick position and button states. The red line on the bullseye on the right indicates the current displacement of the joystick. The red bar on the left is divided into segments representing all buttons on the joystick. When a button is pressed, the corresponding segment turns green.

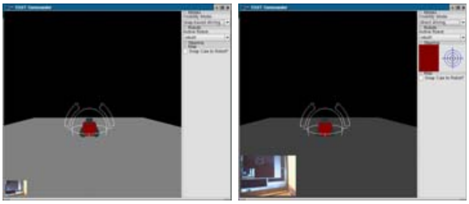


Fig. 5. An example of the Visibility Manager, where attention is directed by scaling a video stream up

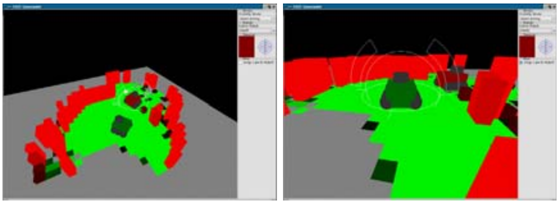


Fig. 6. An example of the GLViewCanvas and the RobotList, where it is possible to either have a free moving camera (left) to get an overview or to snap the camera view to a robot (right)

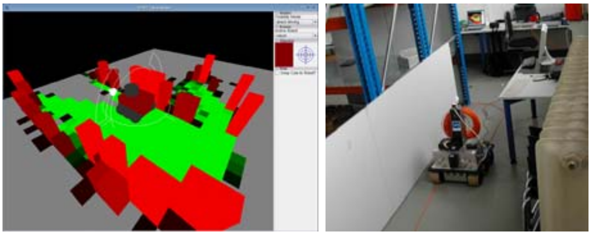


Fig. 7. A test run with Papa Goose (Top: The control interface, Bottom: The robot in the lab)

- The CanvasControl, which gives the single option of snapping the camera of the canvas to the poses of the current robot. This option gives rise to the earlier mentioned "race car"-like driving experience.

4 Conclusion

An adaptive human machine interface (HMI) for rescue robots was presented. The HMI supports adjustable autonomy by automatically changing its display and control functions based on relevance measures, the current situation the robots encounter, and user preferences. The according parameters and rules can be specified during mission planning before the actual run as well as on the fly. The design follows general guidelines from the literature, based on intensive surveys of existing similar systems as well as evaluations of approaches in the particular domain of rescue robots.

References

[BCK04] Andreas Birk, Stefano Carpin, and Holger Kenn. The IUB 2003 rescue robot team. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer, 2004.

[BDM02] David J. Bruemmer, Donald D. Dudenhoeffer, and Julie L. Marble. Dynamic-autonomy for urban search and rescue. In *Proceedings of the 2002 AAAI Mobile Robot Workshop*, Edmonton, Canada, 2002.

[Bir04] Andreas Birk. Fast robot prototyping with the CubeSystem. In *Proceedings of the International Conference on Robotics and Automation, ICRA'2004*. IEEE Press, 2004.

[Bir05] Andreas Birk. The IUB 2004 rescue robot team. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer, 2005.

[BK03] Andreas Birk and Holger Kenn. A control architecture for a rescue robot ensuring safe semi-autonomous operation. In Gal Kaminka, Pedro U. Lima, and Raul Rojas, editors, *RoboCup-02: Robot Soccer World Cup VI*, volume 2752 of *LNAI*, pages 254–262. Springer, 2003.

[BKP03] Andreas Birk, Holger Kenn, and Max Pfingsthorn. The iub rescue robots: From webcams to lifesavers. In *1st International Workshop on Advances in Service Robotics (ASER'03)*. 2003.

[BKR$^+$02] Andreas Birk, Holger Kenn, Martijn Rooker, Agrawal Akhil, Balan Horia Vlad, Burger Nina, Burger-Scheidlin Christoph, Devanathan Vinod, Erhan Dumitru, Hepes Ioan, Jain Aakash, Jain Premvir, Liebald Benjamin, Luksys Gediminas, Marisano James, Pfeil Andreas, Pfingsthorn Max, Sojakova Kristina, Suwanketnikom Jormquan, and Wucherpfennig Julian. The IUB 2002 rescue robot team. In Gal Kaminka, Pedro U. Lima, and Raul Rojas, editors, *RoboCup-02: Robot Soccer World Cup VI*, LNAI. Springer, 2002.

[BNP$^+$] Paul G. Backes, Jeffrey S. Norris, Mark W. Powell, Marsette A. Vona, Robert Steinke, and Justin Wick. The science activity planner for the mars exploration rover mission: Fido field test results. In *Proceedings of the 2003 IEEE Aerospace Conference*, Big Sky, MT, USA.

[CB05] Stefano Carpin and Andreas Birk. Stochastic map merging in rescue environments. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*, page p.483ff. Springer, 2005.

[FCGB00] Terrence Fong, Francois Conti, Sebastien Grange, and Charles Baur. Novel interfaces for remote driving: gesture, haptic and pda. In *SPIE Telemanipulator and Telepresence Technologies VII*, Boston, MA, November 2000.

[FCTB01] Terrence Fong, Nathalie Cabrol, Charles Thorpe, and Charles Baur. A personal user interface for collaborative human-robot exploration. In *6th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS)*, Montreal, Canada, June 2001.

[FT01] T. Fong and C. Thorpe. Vehicle teleoperation interfaces. *Autonomous Robots*, 11(1), July 2001.

[FTB01] Terrence Fong, Charles Thorpe, and Charles Baur. Advanced interfaces for vehicle teleoperation: Collaborative control, sensor fusion displays, and remote driving tools. *Autonomous Robots*, 11(1), July 2001.

[FTB02] Terrence Fong, Charles Thorpe, and Charles Baur. Robot as partner: Vehicle teleoperation with collaborative control. In *Multi-Robot Systems: From Swarms to Intelligent Automata*. Kluwer, 2002.

[GJCP01] Michael A. Goodrich, Dan R. Olsen Jr., Jacob W. Crandall, and Thomas J. Palmer. Experiments in adjustable autonomy. In *Autonomy, Delegation, and Control: Interacting with Autonomous Agents*. IJCAI workshop, 2001.

[KCP$^+$03] Holger Kenn, Stefano Carpin, Max Pfingsthorn, Benjamin Liebald, Ioan Hepes, Catalin Ciocov, and Andreas Birk. Fast-robotics: a rapid-prototyping framework for intelligent mobile robotics. In *Proceedings of the 2003 IASTED International Conference on Artificial Intelligence and Applications*, pages 76–81, Benalmadena, Malaga, Spain, 2003.

[Mim98] Yoshiaki Mima. Bali: A live desktop for mobile agents. In *Proceedings of the 1998 IEEE Third Asian Pacific Computer and Human Interaction*, Kangawa, Japan, July 1998.

[MR96] R. Murphy and E. Rogers. Cooperative assistance for remote robot supervision, 1996.

[OG03] Dan R. Olsen and Michael A. Goodrich. Metrics for evaluating human-robot interactions. In *Proceedings of PERMIS 2003*, September 2003.

[RB05] Martijn Rooker and Andreas Birk. Combining exploration and ad-hoc networking in robocup rescue. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages pp.236–246. Springer, 2005.

[RMH01] M. Micire R. Murphy, J. Casper and J. Hyams. Potential tasks and research issues for mobile robots in robocup rescue. In Tucker Balch Peter Stone and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Lecture keywordss in Artificial Intelligence 2019. Springer Verlag, 2001.

[SB] SGI and Architectural Review Board. The OpenGL Standard. Available at: http://www.opengl.org/.

[SBA$^+$03] Maarten Sierhuis, Jeffrey M. Bradshaw, Alessandro Acquisti, Ron van Hoof, Renia Jeffers, and Andrzej Uszok. Human-agent teamwork and adjustable autonomy in practice. In *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA*, Japan, May 2003.

[Sny01] Rosalyn Graham Snyder. Robots assist in search and rescue efforts at wtc. *IEEE Robotics and Automation Magazine*, 8(4):26–28, December 2001.

[SPT03] Paul Scerri, David V. Pynadath, and Milind Tambe. Towards adjustable autonomy for the real world. *Journal of AArtificialIntelligence Research*, 17:171–228, 2003.

[SYDY04] Jean Scholtz, Jeff Young, Jill L. Drury, and Holly A. Yanco. Evaluation of human-robot interaction awareness in search and rescue. In *Proceedings of the International Conference on Robotics and Automation, ICRA'2004*. IEEE Press, 2004.

[Tro] Trolltech. The Qt Graphical User Interface Toolkit. Available at: http://www.trolltech.com/.

Communicative Exploration with Robot Packs

Martijn N. Rooker and Andreas Birk

School of Engineering and Science
International University Bremen
Campus Ring 1, D-28759 Bremen, Germany
a.birk@iu-bremen.de

Abstract. Exploration is a core challenge for RoboCup Rescue. So-called communicative exploration is a novel strategy for multi-robot exploration that unlike other approaches takes the limits of wireless communication systems into account. Here, previous results that where achieved for a team of robots linked to a basestation are significantly extended to also cover robot packs, i.e., multi-robot teams that are not permanently tied to an operator's station. Unlike teams that are constrained by the immobility of a basestation, packs can explore arbitrarily large regions. Results from experiments with packs of 4, 5 and 6 robots are presented. The new strategy constantly maintains the communication between the robots while exploring, whereas the commonly used frontier-based exploration strategy, which is used in the experiments as comparison to our approach, leads to a rapid loss of communication.

1 Introduction

Exploration is a core issue for many robotics applications [Zel92, MWBDW02] including especially RoboCup Rescue. Obviously, the usage of multi-robot systems is a very interesting option for exploration as it can lead to a significant speed-up and increased robustness, which both are very important for rescue missions. A popular basis for multi-robot exploration is the frontier-based Exploration algorithm introduced by Yamauchi [Yam97], which was extended by himself [Yam98] as well as by Burgard et.al.[BFM$^+$00] to deal with multiple robots. These extensions suffer the drawback that perfect communication between the robots is assumed. When it comes to real multi-robot systems, communication is based on wireless networks typically based on the IEEE 802.11 family of standards, which is also known as WLAN technology [OP99]. WLAN links suffer from various limitations [PPK$^+$03]. Especially, they have a limited range posing a severe limit on the usefulness of the aforementioned exploration algorithms. In [RB05] we presented a new exploration strategy that takes the range limits into account and that is therefore more suited for real application scenarios. This previous work was limited to robots tied to a basestation, i.e., there was a hard limit to the maximum area that could be explored. Here, we extend the previous work to robot packs, i.e., the constraint of a basestation is dropped. This is especially of interested when working with autonomous robots, which is one of the next big challenges within RoboCup Rescue.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 267–278, 2006.

Communicative exploration builds in general on the Frontier-Based Exploration algorithm [Yam97], where a frontier is defined as regions on the boundary between open space and unexplored space. A robot moves to the nearest frontier, which is the nearest unknown area. By moving to the frontier, the robot explores new parts of the environment. This new explored region is added to the map that is created during the exploration. In the multi-robot approach different robots are moving stochastically over to the frontier [Yam98], respectively in a coordinate manner such that multiple robots will not move to the same position [BFM$^+$00]. When we assume a realistic communication model for a multi-robot system, there is a limit to the communication range of each robot. This is not taken into account in previous approaches where nothing prevents the robots from moving further and further away from each other.

We extend the frontier-based exploration such that exploration takes place while the robots maintain a distributed network structure which keeps them in contact with each other through ad-hoc networking [Per00], assuming some underlying dynamic routing [JMH04, RT99, JW96]. This *communicative exploration* algorithm is based on a utility function, which weights the benefits of exploring unknown territory versus the goal of keeping communication intact.

The rest of this paper is structured as follows. Our line of research is motivated in more detail in section 2. The concrete communicative exploration algorithm is introduced in section 3. The experiments and results are presented in section 4. Section 5 concludes the paper.

2 Exploration by Robots Packs

The approach presented in this paper is based upon the frontier-based exploration, which is described in [Yam97, Yam98]. As mentioned in the introduction, the basic idea of this algorithm is simply to move to the boundary between explored and open space. As illustrated[1] in figure 1, robots tend to drift apart and communication is lost.

At the International University Bremen (IUB), a team is working since 2001 in the domain of rescue robots [BKP03, BCK04, BKR$^+$02](Figure2). Rescue robots shall assist first responders in urban disasters scenarios ranging from earthquakes to gas or bomb explosions [RMH01, Sny01]. A typical mission tasks is the detection and localization of victims. Exploration combined with the constraints of real-world communication systems is an obvious topic of interest in this application scenario. We were hence interested in overcoming the limitations of the frontier-based approach.

We describe in [RB05] a first step where the issue of keeping in constant contact with a base-station while exploring is addressed (figure 3). The transmission of data to an operators station is crucial in rescue missions as it can

[1] Cells with explored space are colored dark green, unexplored ones are bright gray, the frontier cells are yellow, obstacles are dark gray. Robots are red spots, their communication ranges red circles. Active links are indicated by red lines.

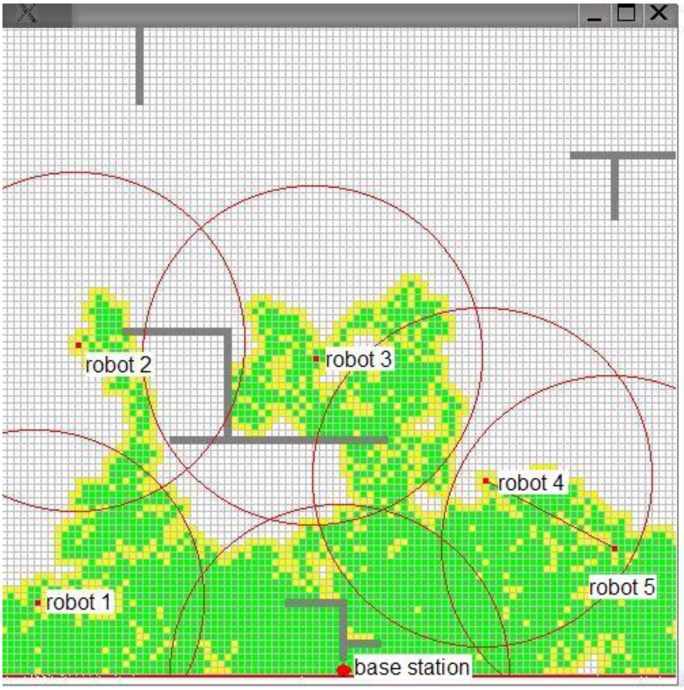


Fig. 1. The frontier-based exploration algorithm does not put any constraints on the spread of the robot pack. The robots soon drift far apart and communication is easily lost. In the above screenshot from a typical simulated run, only robots 4 and 5 are in each others cell and hence capable of communicating with each other.

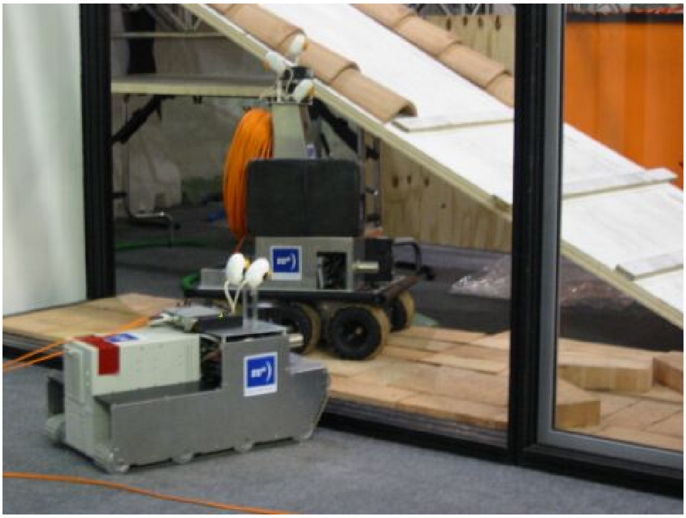


Fig. 2. Two of the IUB rescue robots at the RoboCup Rescue competition

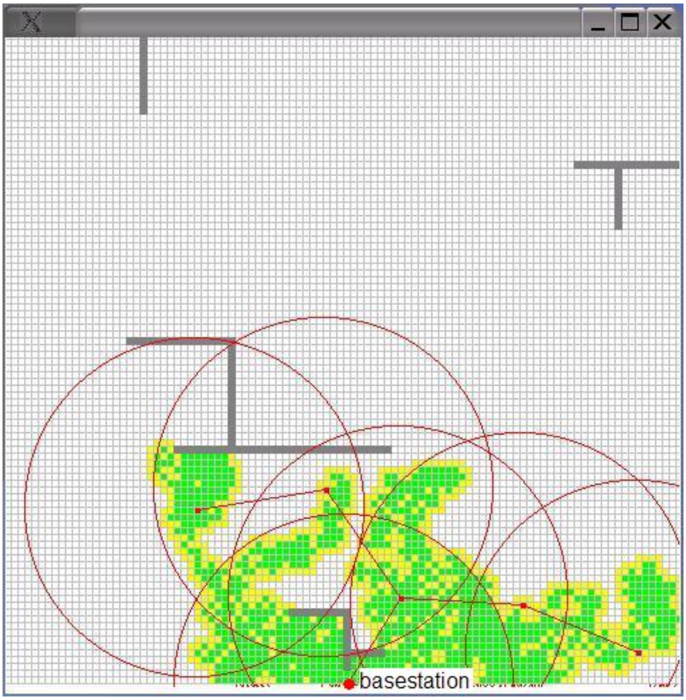


Fig. 3. A screen-shot from a typical simulated run of the basic communicative exploration algorithm where the robots constantly keep in contact with an immobile base-station. For this purpose, an utility function penalizes the motion that lead to a loss of a link in the ad-hoc network formed by the robots. Though being important for some application scenarios, this approach limits the maximum operation area of the robots.

not be assumed that the robots return to the spot where they are deployed, in contrary, their total loss during a mission is a likely risk. Therefore, they have to deliver all crucial information, like victims and hazards found or map-data, ideally on-line to an operators station, which is at a secured position. For this purpose, the robots either have to be in direct contact with the base-station or to use other robots as relays.

In this paper, an extension of our previous results to robot packs is presented. This means that the constraint of keeping contact with an immobile base-station is dropped. Instead, the *communicative exploration* algorithm keeps the network structure in a robot pack intact, which can move freely to do the exploration. This allows to apply the results to arbitrary robot teams. Note that the algorithm can be highly beneficial independent of exploration tasks. Though there is some work dealing with cooperative robots without communication [Ark92], typical architectures for coordinating multi-robot systems like ALLIENCE [Par02] require proper communication structures.

3 The Communicative Exploration Algorithm

The following section describes the *communicative exploration* algorithm in detail. Note that to make a guaranteed "optimal" exploration, i.e., to cover the largest possible area without communication loss, a proper motion-planning for the aggregate of all n robots would have to be done in *every* step, which is for complexity reasons infeasible. The basic idea is therefore to use an utility function that penalizes moves that lead to a loss of communication links. The utility is then used to select a best possible candidate from a random population.

At time t, every robot i has a position $P_i(t) = (x_i, y_i)$, which represents the position in the world W. W is represented by a grid, where every cell can contain four different values, namely *unknown, frontier, visited, obstacle.*

Configuration of n robots at time t is defined as:

$$cfg(t) = \{P_1(t), P_2(t), \cdots, P_n(t)\}$$

For moving to a new configuration at time $t + 1$, a configuration change cfg_c is calculated. A configuration change is defined as follows:

$$cfg\_c(t) = \{m_1(t), m_2(t), \cdots, m_n(t)\}$$

with $m_i(t)$ being the movement of robot i at time t. For every robot, the following movements are defined:

$$m_i(t) \in M = \{N, NE, E, SE, S, SW, W, NW, R\}$$

with R representing no movement and the other values a movement in one to the surrounding grid cells, if possible.

With this definition for a configuration change, there are in total 9^n configuration changes possible for n robots. Unfortunately, all possible configurations are not always possible. For example, it could happen that in a specific configuration a robot moves into an obstacle or that multiple robots in a configuration move to the same position, which will result in a collision and maybe damage of the robots. Furthermore, the exponential number of possible new configurations makes it impossible to calculate all of them. Hence, the decision has been made to generate a limited amount of random calculated configurations per time-step. Instead of considering all the 9^n possible configurations, k configurations are calculated with $k \ll 9^n$.

For every new calculated configuration, a *utility value* is calculated. This utility value represents the usefulness of the new calculated configuration. For the calculation of the utility of a configuration, the different possible locations where the robots can move to are taken into consideration. Every robot in the configuration adds a certain value to the total configuration, depending on its position in the calculated configuration. The following possibilities for a new position can occur:

- **Impossible position:** This position can happen in two different situations:
  - Two or more robots want to move to the same position.
  - A robot wants to move to a position that is occupied by an obstacle.

  Configurations with these position should be avoided, therefor a negative value is assigned to these positions.
- **Loss of communication:** As mentioned before, the idea behind the approach is to maintain communication between all robots during the exploration process. The maintenance of communication can be direct or indirect. To check if communication is maintained in a configuration, the whole configuration at this point and not only a specific robot. As soon as one robot in the configuration is out of communication range, the configuration should be avoided. In this case a negative value is assigned to the total utility value of this configuration.
- **Frontier cell:** A frontier cell represents the boundary between explored and unexplored areas of the world. So for exploring new areas, the robots should move to frontier cells. Configurations where robots move to frontier cells are the ones that are favored above all, therefor a positive value is assigned to every robot that moves to a frontier cell.
- **Other:** The last possibility where a robot can move to is a cell that has already been explored. This could also mean that a robot maintains its position in the configuration. Although this position is not optimal, it is not a position that has to be avoided. This position will not add anything to the utility of a configuration, there a "neutral" value is assigned to this position.

The utility value of a single robot position is calculated with the following values:

$$U(P_i(t+1)) = \begin{cases} -100 & \text{if impossible position} \\ 1 & \text{if frontier cell} \\ 0 & \text{otherwise} \end{cases}$$

With these values the total utility value of a configuration change can be calculated with the following formula:

$$U(cfg\_c_i) = \delta + \sum_{i=1}^{n} U(P_i(t+1))$$

with δ being the assigned value for the maintenance or loss of communication. In the experiments performed the value for losing communication is set to -10.

The decision if communication is maintained in a configuration can be done in a rather simple manner. Every robot has a neighbor list. This list contains robots that are directly connected, which means that two robots i and j, $i \neq j$, are within each others communication range. In the approach presented here, it is assumed that if two robots are within communication range of each other, there is a communication link between these two robots.

So every robot within communication range is stored in the neighbor list. These neighbor lists can be seen as adjacent list, which can be used to represent a graph. In this graph the robots are the vertices and if two robots are within each others communication range (and thus in each others neighbor list) an edge consists between these two vertices. Now that a graph is available, the communication maintainability can be calculated. To check if two robots are connected with each other (direct or indirect) there should be a path on the graph between these two robots.

To check if all the robots are connected with each other, it is enough to check if every robot is connected to one specific robot. For this, one robot i is taken as temporary base-station and for every robot j, $i \neq j$ is checked if a connection exist with i. If every robot j is connected with robot i, it also means that every robot in the graph is connected with the other robots. Checking if a path exist between two robots can be done by using well-known graphs algorithms, like DFS or BFS.

As mentioned, there are 9^n different configurations for n robots. The exponential number of possible new configurations makes it impossible to check all of them. Hence, we generate a limited number of random new configurations per time-step and choose the one with the best utility. So, instead of considering all the 9^n new configuration, only k configurations are considered, with $k << 9^n$. In the experiments presented here k is set to 50, leading to an extremely fast evaluation of the possible configurations.

4 Experiments and Results

For the experiments performed, we define a world with some obstacles placed in it, as can be seen in figure 4. The world is represented as a classic evidence grid [ME85], whereby every grid cell can have one of four different values (*unknown, frontier, visited, obstacle*). At the begin of the experiments, all the cells are initialized to unknown. Only the cells that contain obstacles are initialized different.

The robots that are used in the experiments are homogeneous. They have the possibility to explore a certain region. At a certain position, the current position will be marked as *visited* and the surrounding cells will be marked as *frontier*. Every robot has the ability to communicate. If two robots are within a certain range of each other, a communication link is created between them. In the experiments, it is assumed that a communication link is always created when two robots are within communication range. Furthermore, every robot has the ability to move around in the environment. The robots are moving from one grid cell to the other, thereby having the possibility to move vertical, horizontal, diagonal or to remain at their current position.

The frontier-based exploration is used for comparison to the communicative exploration. For testing how well the different exploration approaches perform, the following measurements are taken. For both approaches it is calculated how many grid cells are explored during a run. Each run consists out of 2500 time steps, whereby a time step is defined as every robot making one movement. A movement can in this case also imply that a robot remains at its current position. Furthermore, for the frontier-based approach, the *communication level* is calculated. The communication level indicates how many robots are connected to each other. If there is a connection level of 100% all the robots are connected with each other.

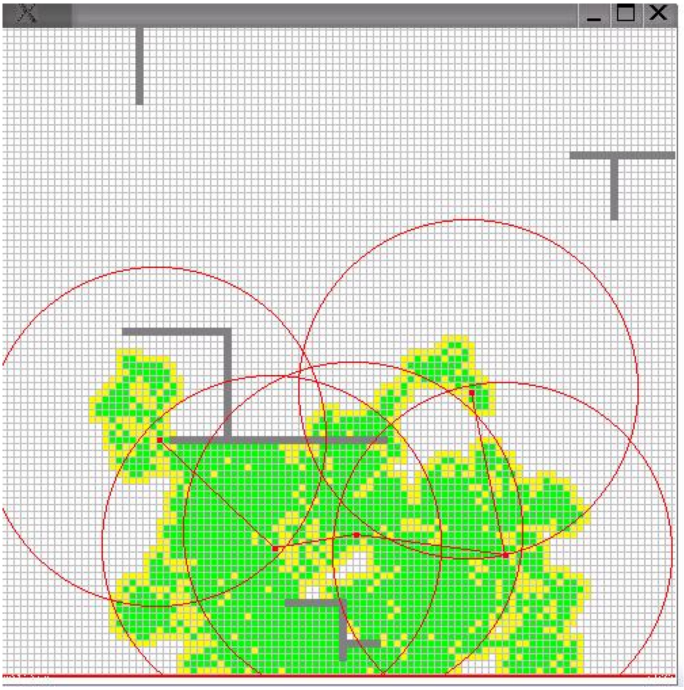
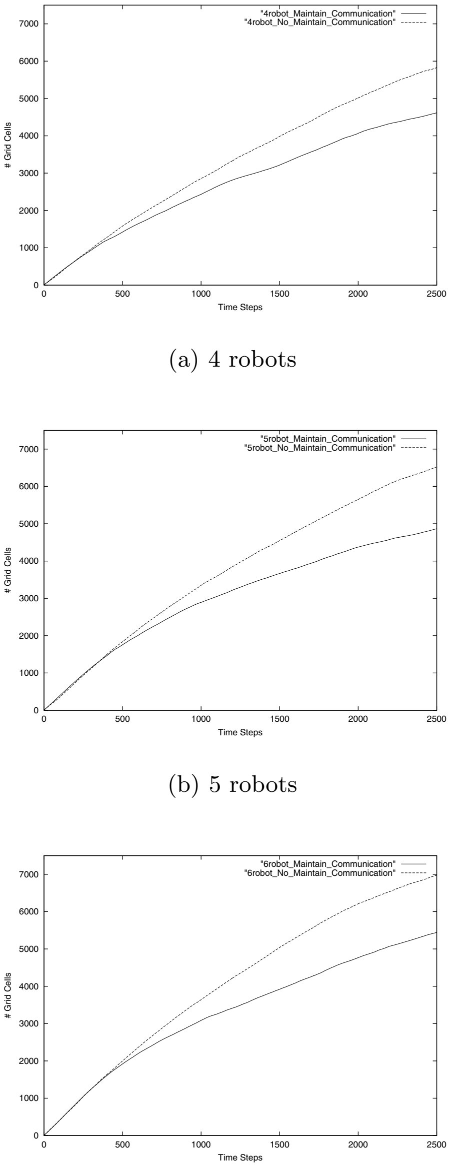


Fig. 4. A pack of 5 robots exploring while maintaining communication with each other

Experiments with different amounts of robots are performed. One experiment uses 4 robots, the second 5 robots and the last experiment 6 robots. Every experiment is performed 10 consecutive times.

The results in terms of exploration speed are shown in figure 5. It can clearly be seen that frontier-based exploration is performing better than communicative exploration in this respect. This can be explained in an easy way. The advantage of the first approach is that the robots do not have a "limitation" on their movement, as do the robots in the communication-based approach have. Therefore they can spread out very fast.

Although communicative exploration moves on slower, its most important task is accomplished. During the whole process of exploration, communication between all the robots is maintained, i.e., the communication level is constantly 100%. This can not be said from the other exploration approach. As can be seen in figure 6, full communication is only established in the beginning of the exploration process, but deteriorates after a few time steps amazingly rapid and never reaches a level of full communication again. The reason for this is exactly the same as the reason why this approach is exploring faster. As the movements of the robots are not bounded by the communication threshold, robots travel rapidly out of each other communication range and are not able to restore the communication link between each other again.

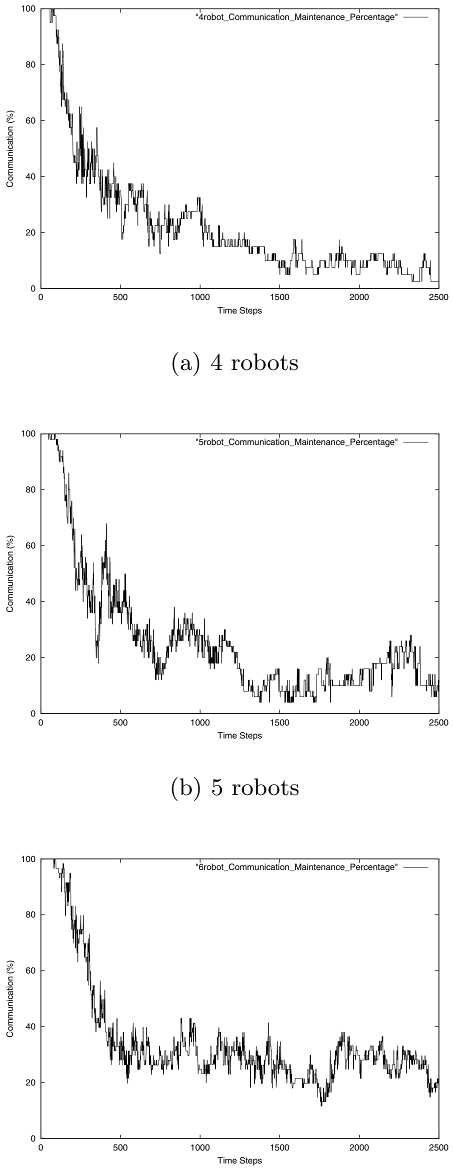


(a) 4 robots

(b) 5 robots

(c) 6 robots

Fig. 5. The amount of grid cells explored with frontier-based and communicative exploration for packs of 4, 5 and 6 robots. The graphs are based on averages of 10 runs. The frontier-based exploration (upper, dashed line) outperforms communicative exploration (lower, solid line). But the communicative exploration maintains communication links between all robots.



(a) 4 robots

(b) 5 robots

(c) 6 robots

Fig. 6. The percentage of communication between the robots during frontier-based exploration for packs of 4, 5 and 6 robots. 100% communication means that every robot is in contact with each other. Again, an average of 10 runs is used. Note that the communication between the robots is lost very fast. Communicative exploration maintains in contrast a constant level of 100%.

5 Conclusions

An extension of the frontier-based approach for exploration [Yam97, Yam98] was presented. In the original algorithm, all robots operate at the borderline to the unexplored space. They hence move further and further away from each other. This leads to communication loss when a realistic network model based on cells with limited ranges is applied. The novel approach of communicative exploration manages to maintain sufficient links between the robots such that a proper network structure is kept. This is achieved by a simple utility function that penalizes the threat of communication losses. In previous work of ours, the approach was limited to robot teams that are constraint by a basestation. Here, we extend the approach to freely moving robot packs. Experiments are presented with packs of 4, 5 and 6 robots where frontier-based exploration leads to a rapid loss of communication. Communicative exploration manages to constantly keep the communication between the robots in the pack intact while exploring. This feature is bought at the expense of somewhat slower progress in the exploration process.

References

[Ark92] R. C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[BCK04] Andreas Birk, Stefano Carpin, and Holger Kenn. The IUB 2003 rescue robot team. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer, 2004.

[BFM$^+$00] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[BKP03] Andreas Birk, Holger Kenn, and Max Pfingsthorn. The iub rescue robots: From webcams to lifesavers. In *1st International Workshop on Advances in Service Robotics (ASER'03)*. 2003.

[BKR$^+$02] Andreas Birk, Holger Kenn, Martijn Rooker, Agrawal Akhil, Balan Horia Vlad, Burger Nina, Burger-Scheidlin Christoph, Devanathan Vinod, Erhan Dumitru, Hepes Ioan, Jain Aakash, Jain Premvir, Liebald Benjamin, Luksys Gediminas, Marisano James, Pfeil Andreas, Pfingsthorn Max, Sojakova Kristina, Suwanketnikom Jormquan, and Wucherpfennig Julian. The IUB 2002 rescue robot team. In Gal Kaminka, Pedro U. Lima, and Raul Rojas, editors, *RoboCup-02: Robot Soccer World Cup VI*, LNAI. Springer, 2002.

[JMH04] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr), July 2004. IETF Internet Draft, draft-ietf-manet-dsr-10.txt.

[JW96] D. B. Johnson and D. A. Waltz. Dynamic source routing in ad-hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[ME85] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 116–121, 1985.

[MWBDW02] Alexei A. Makarenko, Stefan B. Williams, Frederic Bourgault, and Hugh F. Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ Intl. Workshop on Intelligent Robots and Systems*, 2002.

[OP99] Bob O'Hara and Al Petrick. *The IEEE 802.11 Handbook: A Designer's Companion.* Standards Information Network IEEE Press, 1999.

[Par02] Lynne E. Parker. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220 – 240, 2002.

[Per00] C. E. Perkins. *Ad Hoc Networking.* Addison Wesley Professional, 2000.

[PPK$^+$03] Jin-A Park, Seung-Keun Park, Dong-Ho Kim, Pyung-Dong Cho, and Kyoung-Rok Cho. Experiments on radio interference between wireless lan and other radio devices on a 2.4 ghz ism band. In *The 57th IEEE Semiannual Vehicular Technology Conference*, volume 3, pages 1798 – 1801, 2003.

[RB05] Martijn Rooker and Andreas Birk. Combining exploration and ad-hoc networking in robocup rescue. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages pp.236–246. Springer, 2005.

[RMH01] M. Micire R. Murphy, J. Casper and J. Hyams. Potential tasks and research issues for mobile robots in robocup rescue. In Tucker Balch Peter Stone and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Lecture keywordss in Artificial Intelligence 2019. Springer Verlag, 2001.

[RT99] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[Sny01] Rosalyn Graham Snyder. Robots assist in search and rescue efforts at wtc. *IEEE Robotics and Automation Magazine*, 8(4):26–28, December 2001.

[Yam97] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automoiaton*, pages 146–151, July 1997.

[Yam98] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents(Agents '98)*, pages 47–53, May 1998.

[Zel92] Alexander Zelinsky. A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, 8(6), 1992.

Consistency Management Framework for Database Used in Integrated Simulations

Itsuki Noda

Information Technology Research Institute
National Institute of Advanced Industrial Science and Technology
2-41-6 Aomi, Koto-ku, Tokyo 135-0064, Japan
i.noda@aist.go.jp

Abstract. In this article, I propose a mathematical framework of consistency management that guarantees validity among data that are used in integrated simulation systems. When we apply integrated simulations to real-time prediction/evaluation of complex social phenomena like disaster and rescue, checking and keeping consistency of data is an important issue to validate simulation results, because multiple and delayed information are reported to the database continuously in real-time applications. The proposed formalization gives fundamental background of consistency and validity of database and simulation. I also investigate about cost of the management in two major implementation styles.

1 Introduction

In an integrated social simulation like RoboCupRescue Simulation, data management is one of important issues, because various heterogeneous sub-simulators interact with each other. In a Japanese national special project for earthquake disaster mitigation in urban areas (DDT project), we are developing an integrated disaster-and-rescue simulation system [1, 2], in which a number of simulators for various phenomena, for example, damages of ground and road, TSUNAMI, fire-ignition, liquefaction, and so on, are used to provide initial data for the multi-agent rescue simulation. In the case where we apply such an integrated simulation to real-time prediction/evaluation of real phenomena, we must pay attention to management of consistency of dataset used in sub-simulators in order to guarantee the validity of total simulations.

The consistency of data-set can be broken by the following features of the application:

- Several data can be reported for a single phenomena. Some of the data may include noise or may be wrong.
- Some data reach to rescue headquarters with large delay. Such data may conflict with assumed setting of simulations that run before the arrival of the data.

In such case, each session of the simulation is grounded to the different dataset. This means that we need to handle carefully the simulation result.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 279–289, 2006.

Even in the case where we use simulations off-line, we face similar situation when we run many sessions using various initial datasets. In an integrated simulation, some sub-simulations use results of other simulations. Therefore, we need to bring together results of sub-simulations for another sub-simulation carefully not to use inconsistent datasets.

In this article, I will give a formalization of *version control* of datasets to keep consistency among datasets. In the rest of this paper, requirements for the version control are figured out by listing several use-cases of integrated simulations in section 2. Section 3 shows definitions and theories about *version* of datasets and its control. Costs of several operation of version control are investigated in section 4, and discussed from various point of view in section 5.

2 Requirements

In order to figure out the concept and requirement of *version control*, I will point out some use-cases of datasets in a database that is used with a integrated simulation.

2.1 Use-Case 1: Noise in Sensing

In real-time applications, data are not always true. In general, different sensors report different data for the same phenomenon. Sometimes, the difference among these data is significantly large, so that we can not handle these data as a distribution of the same phenomenon. In such a case, we must consider some data are true and others are false. Generally, it is difficult to distinguish true/false of each data.

For example, 5 and 7 were reported alternately as a value of a seismic intensity of a certain aftershock at a certain point in Chuetsu Earthquake in Nov. 2004. In such case, results of simulations based on intensity 5 and 7 are quite different. Generally, we must choose one of 5 or 7 as a true value of the intensity because it is generally meaningless to use an average of 5 and 7.

Choice of values causes another issue on the integrated simulation. Suppose that there are phenomena X, Y, Z, W. In an integrated simulation, Y and Z can be estimated from X value by simulations independently, and W can be calculated from Y and Z. Suppose X is sensed by two sensors a and b, which report different values x_a and x_b, respectively. Here, a simulation calculates Y using x_a and outputs y_a as its value. In the same time, another simulation calculates Z as z_b based on x_b. In this case, we should avoid calculating value W by y_a and z_b, because these values are grounded on different values of phenomenon X. For example, we should not integrate results of two simulations, estimated damage of road based on intensity 5 and estimated number of victims under intensity 7.

2.2 Use-Case 2: Delay of Report

Because sensors are located far from the database, the sensed values are reported with delay. Especially in applications to disaster and rescue, the delay may be very large. Actually, damages of Yamakoshi village in Niigata were reported to head quarters of rescue two days after Chuetsu Earthquake.

In such cases, not all data are available for a certain simulation, so that we must estimate lacked data by interpolating or other methods to run the simulation. When the estimated data turns out to be different from the real data that are reported with delay, we must ignore results of the simulation and re-calculate using the new real data.

2.3 Use-Case 3: Complex Dependency

In an integrated simulation, we may have deep dependencies among sub-simulations. For example, in DDT project[1, 2], the simulation system consists of more than 10 sub-simulations, which are connected with each other via a database.

In such case, relation of dependency described in section 2.1 becomes more complex. In addition, large-scaled simulations like social ones handle huge dataset. Therefore, checking mechanism of consistency should be scalable and light-weight.

2.4 Summary of Requirements

Based on the above discussions about the use-cases, we can summarize that management systems of dataset should have:

- a facility to realize a kind of version control to manage choices of dataset,
- a way to check consistency among simulations and dataset,
- a facility to extract dataset that is consistent with an existing dataset, and
- a mechanism to keep consistency of existing dataset when a delayed sensor data is inserted into the database.

3 Formalization

In this section, I give a formalization of version control of dataset, and derive theorems about consistency, both of which enable to handle requirements examined in the previous section.

3.1 Database and Snapshot

We suppose that a *database* is a collection of *data*, each of which indicates a sensed or simulated value about a *thing* at a certain time. We also suppose that the *database* is dynamic. In other words, sensors and simulators put new *data* into the database continuously [1].

Formally, these words are defined as follows:

[1] We assume that data in a database are never updated or deleted. When various values for a certain event are reported, then all values are stored independently in the database.

Thing: A thing θ is an identifier of an object, feature, or phenomena. For example, "building X", "speed of car Y" and "seismic intensity at a point Z" are *things.* A thing can consist of other things according to a structure of a corresponding object or phenomena.

Event: An event e is a snapshot of a thing θ at a time t. An event e can be notated as a tuple $\langle \theta, t \rangle$. For example, "damage of human X at time t" "speed of car Y at time t", and "the degree of blockade of road Z at time t" are *events.*

Data: A data $d_{\theta,t,s}$ is a value of event $\langle \theta, t \rangle$ acquired by a sensor or simulation s. Because a sensed or simulated value is affected by several conditions and noises, multiple *data* for a certain *event* can exist. When the data comes from a sensor, s indicates an identifier of the sensor. When the data is estimated by a simulation, s is a *version* defined in the next section.

When we indicate a time to store the data into a database, we use the following notation:

$$d_{\theta,t,s;\tau},$$

where τ indicates the time when the data is stored into a database.

Database: A database $\mathbf{DB} = \{d_{\theta,t,s;\tau}\}$ is defined as a set of whole data stored in the database.

$\mathbf{DB}_{(0,\mathcal{T})} = \{d_{\theta,t,s;\tau} \in \mathbf{DB} | \tau < \mathcal{T}\}$ indicates a snapshot of the database at a certain time $\mathcal{T}$. In other words, $\mathbf{DB}_{(0,\mathcal{T})}$ is a set of data that are stored into $\mathbf{DB}$ before $\mathcal{T}$.

3.2 Version

We say a session of a simulation to indicate a run of the simulation using certain initial data and settings. Generally, a session is performed as follows:

1. Collect a set of data $\boldsymbol{D}_{\text{ground}}$ about some events $\boldsymbol{E}_{\text{ground}} = \{e_i | i\}$ from $\mathbf{DB}$ at the begin time of the simulation ($\mathcal{T}_{\text{begin}}$).
2. Estimate a set of data $\boldsymbol{D}_{\text{target}}$ about other events $\boldsymbol{E}_{\text{target}} = \{e_i | i\}$.
3. Store $\boldsymbol{D}_{\text{target}}$ into $\mathbf{DB}$ at the end time of the simulation ($\mathcal{T}_{\text{end}}$).

We call these set of data of input or output of a session of a simulation as a *version.* In other words, a session receives a version from a database as a ground of simulation and inserts another version of data into the database as a result of the simulation.

When we conduct an integrated simulation that consists of several sessions of sub-simulations, we need to pay attention to guaranteeing that each session is grounded on a right dataset. For example, a sub-simulation may depends on a collection of results of other sub-simulations as shown in section 2. In this case, we need to avoid that some results are grounded on inconsistent data.

In order to make it easy to check such consistency among dataset, we define *consistency* among *versions* as follows:

Version: A version v is defined as a tuple as follows:

$$\langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \boldsymbol{G} \rangle,$$

where $\boldsymbol{E} = \{e_i|i\}$ is a set of events, $\boldsymbol{D}$ is a dataset under the version, $\mathcal{T}$ is a time when $\mathbf{DB}$ is accessed to manipulate the dataset, and $\boldsymbol{G} = \{u_i|i\}$ is a set of versions on which $\boldsymbol{D}$ is directly grounded. These elements must satisfy the following condition:

$$\boldsymbol{D} \subset \mathbf{DB}_{(0,\mathcal{T})}/E = \{d_{\theta,t,s;\tau}|\, \langle\theta, t\rangle \in \boldsymbol{E}, \tau < \mathcal{T}\}$$

Generating a New Version: There are four types of operation to generate a new version, *sprout*, *extraction*, *production*, and *union.*

A version is *sprouted* when a new dataset is collected from $\mathbf{DB}$, where all data in the dataset should be *primary one that come from sensors directly.* A sprouted version consists of a tuple:

$$\langle \boldsymbol{E}, \mathcal{T}, \boldsymbol{D}, \phi \rangle \, ,$$

where $\boldsymbol{E}$ is a set of related events, $\boldsymbol{D}$ is a collected dataset from $\mathbf{DB}$, and $\mathcal{T}$ is a time to collect $\boldsymbol{D}$.

A version v can be *extracted* from another version $u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$, where the dataset of v ($\boldsymbol{D}_v$) is a subset of Ds_u. The extracted version v is denoted as a tuple:

$$\langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \{u\} \rangle \, ,$$

where $\boldsymbol{D}_v \subseteq \boldsymbol{D}_u$, $\boldsymbol{E}_v = \{e|d_{\theta,t,s} \in \boldsymbol{D}_v, e = \langle\theta, t\rangle\}$, and $\mathcal{T}_v$ ($> \mathcal{T}_u$) is a time of extraction.

When a session of a simulation outputs a version v as a result using version u as a ground, we call the version v as a *production* of the version u, or denote $u \rhd v$. The *production* v is defined as a tuple:

$$u \rhd v \Leftrightarrow v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \{u\} \rangle \, ,$$

where $\boldsymbol{E}_v$ is a set of simulated events, $\boldsymbol{D}_v$ is the result of the simulation, and $\mathcal{T}$ is a time to finish the simulation [2].

A *union* of two versions u and v ($u \oplus v$) is a version to imply a sum of the two versions. A *union* is a virtual version that includes empty sets of events and data, so that the unified version is constructed as follows:

$$u \oplus v = \langle \phi, \mathcal{T}, \phi, \{u, v\} \rangle \, ,$$

where $\mathcal{T}$ is a time to unify two versions.

We define a version v is *grounded* in version u (or denote $u \succ v$) iff v is generated by *union*, *production*, or *extraction* from u. As defined above, the following relation is satisfied:

$$u \succ v \Leftrightarrow u \in \boldsymbol{G}_v$$

where $v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$.

[2] We suppose that results of simulations are stored immediately after the simulations.

A *footmark* is a relation of two versions which are linked recursively by groundness relationships as follows: We say that version u is a *footmark* of version v iff

$$u \succeq^{*} v \Leftrightarrow \begin{cases} u = v \\ \text{ or} \\ u \succ u', u' \succeq^{*} v \end{cases}$$

We use a *trail* as a set of a version $(\{u_i|i\} = \mathbf{Tr}(v))$ iff when each version u_i in the set is a footmark of version v.

Consistency: We define that two versions, u and v, are *primarily consistent* (or denote $u \dot{\sim} v$) iff

$$\forall e \in \boldsymbol{E}_u \cap \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e,$$

where $u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$ and $v = \langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$.

We define that two versions, u and v, are *consistent* (or denote $u \sim v$) iff

$$\forall u' \succeq^{*} u, \forall v' \succeq^{*} v : u' \dot{\sim} v'.$$

In order to guarantee a unified version $u \oplus v$ is dependable, u and v should be consistent.

We define that a version v is *self-consistent* iff

$$v \sim v.$$

3.3 World Line

As mentioned at the definition of *data*, a certain event can have multiple *data* in a **DB**. Some of these data are treated as true and others are ignored in analyses and simulations. We use the word '*world line*' to indicate a set of data that are treated as true and taken into account for the analysis.

Because multiple analyses can be conducted in parallel, there can exist multiple *world lines* on a **DB**. While these *world lines* are inconsistent with each other, all data in a *world line* should be consistent. Here, '*consistent*' means that

> *each data in a world line is sensed or simulated based on the same conditions and dataset.*

The purpose of the version control is to check and keep the consistency when a simulation extends a world line.

Formally, *world line* and related concepts are defined as follow:

World Line: A world line $\boldsymbol{w}$ is any subset of a database **DB**, that is, $\boldsymbol{w} \in \mathbf{DB}$. When a world line $\boldsymbol{w}$ includes multiple data for a certain event e, the value of the event e is treated as a distribution of probability by Monte Carlo interpretation. When a world line $\boldsymbol{w}$ includes no data for a certain event e, the event is treated as "don't-care".

There are no restriction for a world line to select data in a database. Therefore, whole set of world line is same as power set of whole data, $2^{\mathbf{DB}}$.

A world line is independent from changes of the database.

Projection of World Line: $\boldsymbol{w}/t = \{d_{\theta,t',s;\mathcal{T}} \in \boldsymbol{w} | t' = t\}$ indicates a projection of a world line w at time t (or simply, *time slice* of a world at time t).

$\boldsymbol{w}/\theta = \{d_{\theta',t,s;\mathcal{T}} \in \boldsymbol{w} | \theta' = \theta\}$ indicates a projection of a world line w about a certain thing θ.

$\boldsymbol{w}/e = \{d_{e',s;\mathcal{T}} \in \boldsymbol{w} | e' = e\}$ indicates a projection of a world line w about a certain event e.

Consistency of World Line. We define a world line $\boldsymbol{w}$ *supports* a version v, or denote $\boldsymbol{w} \vdash v$ iff the following condition is satisfied:

$$\forall u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u \rangle \succ^{*} v, \forall e \in \boldsymbol{E}_u : \boldsymbol{w}/e = \boldsymbol{D}_u/e$$

We define a world line $\boldsymbol{w}$ is *consistent* iff the following condition is satisfied:

$$\forall d_{\theta,t,s;\mathcal{T}} \in \boldsymbol{w} : s \text{ is a sensor id or } \boldsymbol{w} \vdash s.$$

In other words, each simulated data in a consistent world line should be included by a version supported by the world line.

It is important that only consistent world lines are meaningful in a database. If a world line is inconsistent, that is, some data are not supported by the world line, the world line is useless because the data is not well-grounded into the world line. Such situation should be avoided when we conduct simulations and update database.

3.4 Theorems About Keeping Consistency

Here, I will derive some theorems that show how generated versions are supported by consistent world lines.

Lemma 1. *When a version v is self-consistent and a world line $\boldsymbol{w}$ consists of data that belong to versions in a trail of v, any footmarks of v is supported by $\boldsymbol{w}$.*

$$\begin{aligned} \forall v \ : \ & v \textit{ is self-consistent}, \\ & \boldsymbol{w} = \{d | u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle \succ^{*} v, d \in \boldsymbol{D}_u\} \\ \rightarrow & \ \forall u \succ^{*} v : \boldsymbol{w} \vdash u \end{aligned}$$

Proof (Lemma 1). Suppose that there exists a footmark u of the version v, which is not supported by the world line $\boldsymbol{w}$. In other words,

$$\exists u = \{\boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u\} \succ^{*} v, \exists e \in \boldsymbol{E} : \boldsymbol{w}/e \neq \boldsymbol{D}_u/e.$$

Because of the definition, $\boldsymbol{w}$ is a super set of $\boldsymbol{D}_u$. So, $\boldsymbol{w}/e \neq \boldsymbol{D}_u/e$ is satisfied iff $\exists d \in \boldsymbol{w}/e, d \notin \boldsymbol{D}_u/e$. This means:

$$\begin{aligned} \exists u' \{\boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'}\} \in \mathbf{Tr}(v) : u' \neq u, \\ d \in \boldsymbol{D}_{u'} \end{aligned}$$

However, this violates the definition of self-consistency of version v. Therefore, any footmark u is supported by $\boldsymbol{w}$.

Using this lemma, we can derive the following theorem.

Theorem 1. *When a version v is self-consistent, there exists a world line $\boldsymbol{w}$ that is consistent and supports v.*

$$\forall v : v \textit{ is self-consistent} \rightarrow \exists \boldsymbol{w} : \boldsymbol{w} \textit{ is consistent}, \boldsymbol{w} \vdash v$$

Proof (Theorem 1). Consider a world line $\boldsymbol{w} = \{d | u = \langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle \overset{*}{\succ} v, d \in \boldsymbol{D}_u\}$. $\boldsymbol{w}$ supports any footmark of version v because of Lemma 1. On the other hand, for any simulated data $d_{\theta,t,s;\tau} \in \boldsymbol{w}$, the version s to which the data belongs is always in the trail of version v. Therefore, the version s is supported by the world line w.

This theorem tells that we need pay attention only to keeping self-consistency of newly generated versions in order to make the versions are meaningful.

Based on the first theorem, we can derive the following theorems about generation of new versions.

Theorem 2. *Any sprouted version is self-consistent.*

Proof (Theorem 2). Because of all data in a sprouted version come from sensor directly, the version is grounded itself [3].

Theorem 3. *When a version v is extracted from a self-consistent version u, v is self-consistent iff the following condition is satisfied:*

$$\begin{aligned} \forall v : & u \supset v, \\ & v \textit{ is self-consistent} \leftrightarrow \forall e \in \boldsymbol{E}_v : \boldsymbol{D}_u/e = \boldsymbol{D}_v/e, \end{aligned}$$

where u and v are $\langle \boldsymbol{E}_u, \mathcal{T}_u, \boldsymbol{D}_u, \boldsymbol{G}_u \rangle$ and $\langle \boldsymbol{E}_v, \mathcal{T}_v, \boldsymbol{D}_v, \boldsymbol{G}_v \rangle$, respectively.

Proof (Theorem 3). If the condition is satisfied, u and v is consistent. Because u is self-consistent, v is also consistent.

Suppose that the condition is not satisfied. In this case, there exists an event e that has different dataset in u an v, that is $\boldsymbol{D}_u/e \neq \boldsymbol{D}_v/e$. Because $\boldsymbol{E}_v$ is a subset of VE_u, e is included both in $\boldsymbol{E}_v$ and $\boldsymbol{E}_u$. Therefore, u and v is not consistent, so that v is not self-consistent.

Theorem 4. *When a version u is self-consistent, a production v of the version u is self-consistent and there exists a consistent world line that supports both versions v and u.*

$$\begin{aligned} \forall u : & \textit{ self-consistent} \rightarrow \\ & \qquad \forall v : u \rhd v \rightarrow v \textit{ is self-consistent}, \\ & \qquad\qquad \exists \boldsymbol{w} : \boldsymbol{w} \textit{ is consistent}, \boldsymbol{w} \vdash v \end{aligned}$$

Here, we suppose that a simulation never estimate data about the same event as one that it takes as input.

[3] Because of this reason, the sprouted version is defined with no grounding versions.

Proof (Theorem 4). Because of the definition, $\mathbf{Tr}(v)$ is $\{v\} + \mathbf{Tr}(u)$. Because $\boldsymbol{E}_u \cap \boldsymbol{E}_v = \phi$, any footmark of version u is consistent with version v. Therefore, the version v is self-consistent. As the result, there exists a consistent world line that supports v because of Theorem 1.

Theorem 5. *When two versions, v and u, are self-consistent and consistent with each other, a union of these versions is self-consistent, and there exists a consistent world line that supports the union.*

$$\begin{gathered}\forall u, v : \textit{self-consistent}, u \sim v \rightarrow \\ u \oplus v \ \textit{is self-consistent}, \\ \exists \boldsymbol{w} : \boldsymbol{w} \ \textit{is consistent}, \boldsymbol{w} \vdash u \oplus v\end{gathered}$$

Proof (Theorem 5). Because of the definition, $\mathbf{Tr}(u \oplus v)$ is $\{u \oplus v\} + \mathbf{Tr}(u) + \mathbf{Tr}(v)$. Suppose that x and y are versions in $\mathbf{Tr}(u \oplus v)$. When both of x and y are in $\mathbf{Tr}(u)$ or $\mathbf{Tr}(v)$, they are primary consistent with each other because u and v are self-consistent. When one of them is in $\mathbf{Tr}(u)$ and another in $\mathbf{Tr}(v)$, they are primary consistent with each other because u and v are consistent with each other. When one of x and y is identical with $u \oplus v$, they are primary consistent because the version $u \oplus v$ contain no events.

Therefore, the union $u \oplus v$ is self-consistent. As the result, there exists a consistent world line that supports $u \oplus v$ because of Theorem 1.

4 Operation Cost of Version Control

In order to realize version control for integrated simulations, we need to implement several additional functions into the database as follows:

- to store information about version. As its definition, the information should includes a set of related events($\boldsymbol{E}$), access time to the database($\mathcal{T}$), a set of data ($\boldsymbol{D}$), and a set of ground version ($\boldsymbol{G}$). In these items, handling of $\boldsymbol{D}$ is most important because the number of elements in $\boldsymbol{D}$ is generally large.
- to insert new data to the database with keeping self-consistency of each existing version. Especially, data that are reported with large delay should be managed carefully, because events of the data may already be used in some simulation sessions.
- to check consistency of two version. As shown in theorems in the previous section, the critical operation to generate a new version is mainly in *union* operation. To keep self-consistency of unified version, we must check consistency of two versions.

In the followings subsections, we investigate average cost for each operation under two typical implementations of the version control.

4.1 Positive Listing

A strait-forward way to store information about $\boldsymbol{D}$ of a version is to store a list of data (or identifiers of data) in $\boldsymbol{D}$. We call this method as *positive listing.*

Cost to Store Information. The order of the size of storage to keep a list $\boldsymbol{D}$ in the positive listing is $O(|\boldsymbol{D}|)$. This cost become a problem when a simulation handle a large-scale phenomena like earthquake, because the number of data is large in such simulations.

Cost to Insert New Data. When a new data is inserted to **DB**, there occur no additional operations for managing consistency of versions, because the new data does not affect on $\boldsymbol{D}$ of existing version.

Cost to Check Consistency. In order to check consistency between versions u and v, the following operation is required:

$$\forall u' \in \mathbf{Tr}(u), \forall v' \in \mathbf{Tr}(v), \forall e \in (\boldsymbol{E}_{u'} \cap \boldsymbol{E}_{v'}) : \\ \text{check } \boldsymbol{D}_{u'}/e = \boldsymbol{D}_{v'}/e$$

where $u' = \langle \boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'} \rangle$ and $v' = \langle \boldsymbol{E}_{v'}, \mathcal{T}_{v'}, \boldsymbol{D}_{v'}, \boldsymbol{G}_{v'} \rangle$. The order of the cost of this operation is $O(\sum |\boldsymbol{D}_{u'}| + \sum |\boldsymbol{D}_{v'}|)$.

4.2 Negative Listing

Another strategy to store information about $\boldsymbol{D}$ is to store a list of data that are related with events in $\boldsymbol{E}$ but do not appear in $\boldsymbol{D}$. We call this method as *negative listing*. In other words, the *negative listing* stores data that are not used in the version.

Generally, such not-used data will occur under the following situation:

- When multiple various data about a certain event is reported by different sensors, some of them may be considered as false data that are not used in a certain simulation session.
- When a simulation starts at time $\mathcal{T}$ using data of a certain event e, other data about e may be reported after $\mathcal{T}$.

In the negative listing method, a list of the following dataset is stored to keep information about version v:

$$\bar{\boldsymbol{D}}_v = \sum_{e \in \boldsymbol{E}_v} (\mathbf{DB}_{(0,\mathcal{T}_v)}/e - \boldsymbol{D}_v)$$

Cost to Store Information. The order of the size of storage to keep a list $\bar{\boldsymbol{D}}$ is $O(|\bar{\boldsymbol{D}}|)$. Generally, it is expected that this cost is smaller than the cost in the positive listing, because the data in $\bar{\boldsymbol{D}}$ is generally exceptional one.

Cost to Insert New Data. When a new data is reported to **DB** after a timestamp $\mathcal{T}_v$ of an existing version v, we may need to handle the new data is in the negative list of v, because the data is not used in the version. However, we can distinguish such negative data by comparing timestamps of the data (τ) and the version ($\mathcal{T}$). Therefore, there occurs no additional operation to manage the consistency.

Cost to Check Consistency. In order to check consistency between versions u and v, the following operation is required:

$$\forall u' \in \mathbf{Tr}(u), \forall v' \in \mathbf{Tr}(v), \forall e \in (\boldsymbol{E}_{u'} \cap \boldsymbol{E}_{v'}) :$$
$$\begin{cases} \text{check } \bar{\boldsymbol{D}}_{v'}/e - \bar{\boldsymbol{D}}_{u'}/e = \mathbf{DB}_{(\mathcal{T}'_u,\mathcal{T}'_v)}/e & \text{if } \mathcal{T}'_u < \mathcal{T}'_v \\ \text{check } \bar{\boldsymbol{D}}_{u'}/e - \bar{\boldsymbol{D}}_{v'}/e = \mathbf{DB}_{(\mathcal{T}'_v,\mathcal{T}'_u)}/e & \text{if } \mathcal{T}'_v < \mathcal{T}'_u \end{cases}$$

where $u' = \langle \boldsymbol{E}_{u'}, \mathcal{T}_{u'}, \boldsymbol{D}_{u'}, \boldsymbol{G}_{u'} \rangle$, $v' = \langle \boldsymbol{E}_{v'}, \mathcal{T}_{v'}, \boldsymbol{D}_{v'}, \boldsymbol{G}_{v'} \rangle$, and $\mathbf{DB}_{(\mathcal{T}_x,\mathcal{T}_y)}$ means a set of data that is inserted between time $\mathcal{T}_x$ and $\mathcal{T}_y$. The order of the cost of this operation is $O(\sum \left|\mathbf{DB}_{(\mathcal{T}_{v'},\mathcal{T}_{u'})}/\boldsymbol{E}\right| + \sum \left|\bar{\boldsymbol{D}}_{u'}\right| + \sum \left|\bar{\boldsymbol{D}}_{v'}\right|)$.

5 Discussion

It is not simple to determine which of positive and negative listings is better than another. In the case of large scale applications, however, the size of data is generally huge. Therefore, the negative listing has an advantage in both costs of storage and consistency checking.

In the evaluation of costs in section 4, the cost to projection operations (for example, $\mathbf{DB}_{(\mathcal{T}'_u,\mathcal{T}'_v)}/e$) is ignored. This is because general database systems like RDB have sophisticated low-cost facilities using indexing/hashing technique.

The cost to manage a set of events $\boldsymbol{E}$ is also ignored in the above evaluation. While datasets can be handled only by listing, a set of events can be represented a projection axis to the database. For example, in geographical application like disaster-rescue simulation, a set of events can be represented by type of features and area of interest. Therefore, we can assume the operation of the event sets is also abstracted and low-cost on storage and consistency checking.

There are several open issues on the formalization as follows:

- How to realize a facility to extract version that is consistent with a certain version. Such operation will happen when a simulation requires parts of result of two versions.
- How to formalize statistic operations of results of multiple simulations. One of purposes of the simulation is to calculate statistic value like averages and variances based on multiple simulation using different random seeds. Current formalization can not handle it, because such operations break consistency among versions.

References

1. Tadokoro, S.: An overview of japan national special project daidaitoku (ddt) for earthquake disaster mitigation in urban areas. In: Proc. of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003) Workshop on Advanced Robots and Information Systems for Disaster Response. (2003)
2. Tadokoro, S.: Japan government special project: development of advanced robots and information systems for disaster response (daidaitoku) — for earthquake disaster mitigation in urban areas —. In: Proc. of IEEE Workshop on Safety Security and Rescue Robotics (SSRR03). (2003)

Design, Dynamic Analysis and Optimization of a Rover for Rescue Operations

Hadi Tavakoli Nia[1], Seyed Hamidreza Alemohammad[1], Saeed Bagheri[1],
Reza Hajiaghaee Khiabani[1], and Ali Meghdari[2]

Center of Excellence in Design, Robotics and Automation
Department of Mechanical Engineering
Sharif University of Technology, P.O. Box: 11365-9567, Tehran, Iran
meghdari@sharif.edu

Abstract. In this paper a new approach to dynamic optimization of a rough terrain rover is introduced. Since rover wheels traction has a significant role in rover mobility, optimization is based on the minimization of traction at rover wheel-ground interfaces. The method of optimization chosen is Genetic Algorithm (GA) which is a directed random search technique along with the usual optimization based on directional derivatives. GA is a suitable and efficient method of optimization for nonlinear problems. The procedure is applied on a specific rough terrain rover called *CEDRA-I Shrimp Rover*. The present work resulted in design and manufacturing of the optimized rover called *CEDRA-II Shrimp Rover*.

1 Introduction

Rough terrain rovers are increasingly used for high risk situations such as rescue operations, planetary explorations and military missions. Future tasks will require robots with high mobility. Keeping stability without tip over or loss of contact with the ground is needed for those hazardous tasks.

In these cases having a flexible rover that can adapt itself to environment of challenging tasks, is very useful. Robots with passively articulated suspensions can improve rough-terrain mobility by modifying their suspension configuration and thus repositioning their center of mass.

Design and control of these rovers are based on dynamical analysis and simpler and more meaningful equations of motion may be helpful in many cases. The complication of equations of motion in rovers arises from several factors including: mechanism with complicated configuration, rough terrain usually with random bumps and nonholonomic constraints.

Much research has been done on rough terrain rovers dynamical analysis and modeling. Tai [1] presented dynamical modeling for mobile robots with suspension. Iagnemma and Dubowsky [2] presented a new method for rough terrain rovers'

[1] Graduate Student.
[2] Professor of Mechanical Engineering.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 290 – 300, 2006.

control. Their work is based on the static modeling of rovers due to the rovers low speed.

Kawabe *et al.* [3] performed substantial work on traction control of passenger vehicles on flat roads. This work is not applicable to low-speed, rough terrain rovers because in these vehicles wheel slip is caused primarily by kinematic incompatibilities. Reister and Unseren [4] studied the traction control for low-speed mobile robots on flat terrains. Sreenivasan and Waldron [5] have represented the displacement analysis of articulated wheeled vehicle configuration and have extended it to uneven terrain motion. Hacot [6] illustrated the traction control of a planetary rover called *Rocker Bogie*. In Ref. [7], optimization of a four-wheel rover is presented. The optimization objective function is finding the rover parameters so that the path traversed by the center of gravity tends to the straight line.

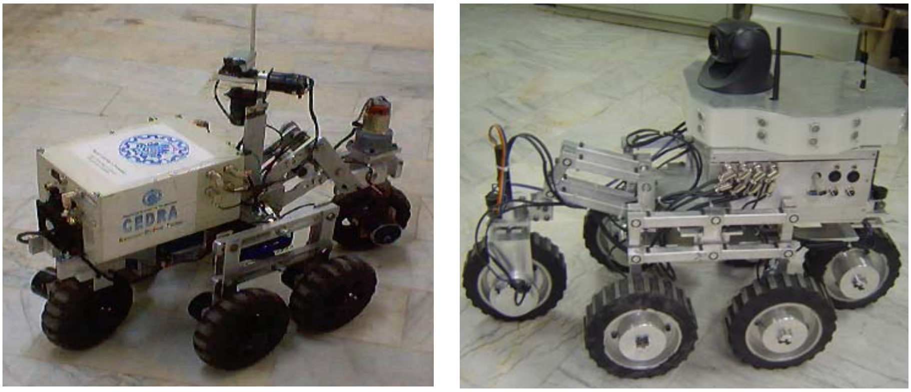
In this paper an innovative approach to dynamical optimization of a rough terrain rover with redundant drive wheels is presented. Optimization is based on the minimization of traction in the rovers' wheel-ground contact. The method of optimization is chosen to be Genetic Algorithm (GA) which is known as a directed random search technique. GA is a suitable and efficient method of optimization for nonlinear problems. In this paper dynamical equations are developed using Kane's method. Compared to other formulations, (Lagrange or Newton) Kane's method involves less arithmetic operations. So, the model simulation is fast and simple. Also, Kane's equations can be easily brought into closed form [8].

Finally the analysis is applied to *CEDRA-I Shrimp rover*, which is a complicated rough terrain rover with six-wheels. *CEDRA-I Shrimp rover* is a laboratory rover made at the Center of Excellence in Design Robotics and Automation (CEDRA) and ranked second in the Rescue Robot Competitions 2003 (RoboCup Rescue 2003). The main structure is based on the rover, which first was constructed at EPFL [9]. This robot is similar to Rocky7 [10] and Marsokhod [11] in some parts but a four-link mechanism added at the front of the robot has made it more efficient in encountering obstacles. The present work resulted in design and manufacturing of the optimized rover called *CEDRA-II Shrimp rove.* Both initial and the optimized rovers are shown in Fig. 1.

2 Dynamics Equations

2.1 Kinematic Analysis

The kinematic analysis is the base point in dynamics analysis and as a result the optimization. Analyses which have been done on rovers up to now are usually for surfaces with simple and specified geometry like an inclined surface; however for a rough terrain, the rover kinematical problem will change significantly. Rover kinematical analysis of complex geometrical surfaces leads to several nonlinear equations; the solution to these problems is the most time consuming part of the analysis. In this section the planar inverse kinematical analysis of the shrimp mechanism is dealt as an example.



(a) **(b)**

Fig. 1. (a) *CEDRA-I and* (b) *CEDRA-II* rescue robots

The shrimp mechanism (see Fig. 2) is a one DOF mechanism. Having the path geometry in hand, one can determine the position of all mechanisms linkages, using the position of the rear wheel. In general for a rover with *n* wheel-ground interface points one can obtain *n-1* close kinematic loop. In the Shrimp mechanism for interface points B, C and D one can write:

$$B_x = c\sin\alpha + e\cos(\alpha+\beta) \tag{1}$$

$$B_y = -c\cos\alpha + e\sin(\alpha+\beta) \tag{2}$$

$$C_x = c\cos\alpha - e\sin(\alpha+\beta) \tag{3}$$

$$C_y = -c\sin\alpha - e\cos(\alpha+\beta) \tag{4}$$

$$D_x = (a+b-g)\cos\alpha + f\sin\alpha + n\cos\gamma + m\cos\xi \tag{5}$$

$$D_y = (a+b-g)\sin\alpha - f\cos\alpha + n\sin\gamma + m\sin\xi \tag{6}$$

where

$$\begin{aligned}\xi = \gamma - \cos^{-1}&\left(\frac{n-h\cos(\pi/2-\gamma+\varphi)}{h^2+n^2-2nh\cos(\pi/2-\gamma+\varphi)}\right)\\ \text{-}\cos^{-1}&\left(\frac{h^2+n^2-2nh\cos(\pi/2-\gamma+\varphi)+l^2-k^2}{2l\sqrt{h^2+n^2-2nh\cos(\pi/2-\gamma+\varphi)}}\right)\end{aligned} \tag{7}$$

Angles α,β are shown in Fig. 2.

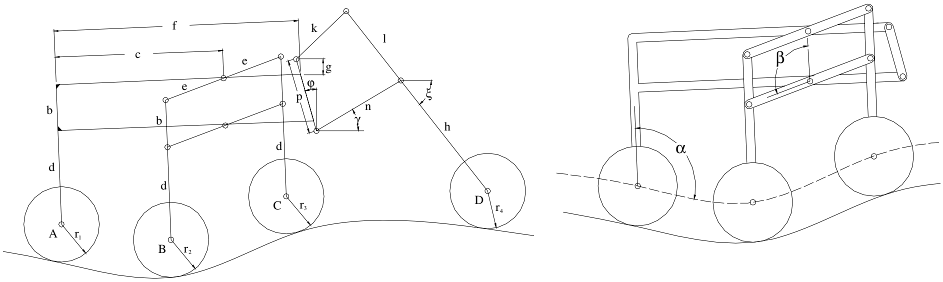


Fig. 2. Shrimp rover mechanism

For simplicity, the wheels center line path is considered rather than the real path. If the wheel center line path's function is shown as $y = path(x)$, then

$$B_y = path(B_x) \tag{8}$$

$$C_y = path(C_x) \tag{9}$$

$$D_y = path(D_x) \tag{10}$$

In this way by solving these three nonlinear equations, the three unknowns, α , β and γ are found and thus the mechanism configuration is determined.

As it was stated before, kinematical analysis of rovers in rough terrains is a cumbersome task. Regarding the shrimp mechanism a nonlinear set of equations should be solved for a position analysis. Since for a velocity or acceleration analysis a derivation process exists and in our case the position terms are not differentiable, a thorough position analysis is required in advance. Then one can use the resultant diagrams to obtain velocity and acceleration with numerical derivation methods.

As it is shown in Fig 2, the shrimp mechanism has 12 linkages. Assuming pure rolling for wheels, 18 two-DOF joints could be considered for this mechanism. According to Kutzbach criterion the system DOF is:

$$DOF = 3(n-1) - 2\times j_1 - j_2 = 3\times 12 - 2\times 18 - 0 = 0 \tag{11}$$

Though the Kutzbach criterion yields zero DOF for this mechanism, it is clear that the right number of DOF is one. This wrong prediction is due to redundant constraint in the middle parallelogram mechanism of the rover. There are several methods to remove this redundancy. Here the cylindrical joint inserting is selected out of available methods. This modification changes the Kutzbach criterion as follows

$$DOF = 3(n-1) - 2\times j_1 - j_2 = 3\times 13 - 2\times 19 - 0 = 1\,. \tag{12}$$

2.2 Kane's Method

In this section the inverse dynamical analysis of rovers, using the shrimp rover mechanism as an example is investigated. A dynamical analysis result is a must for objective function derivation, as will be stated in the next section.

Since the optimization procedure consist of several dynamical analyses at each stage, a computationally efficient dynamical analysis is inevitable. In this regard, due to Kane method's merits in complex systems, this method was chosen as the rover dynamical analysis method.

As was discussed in last section the shrimp mechanism's DOF is one. Like Lagrangian mechanics, in which generalized coordinates are employed in rover mechanics, generalized speed is also used to describe the system motion. The number of generalized speeds is equal to the system's DOF; in the case of the shrimp mechanism we need one generalized speed as follows:

$$u_1 = \dot{q}_1 \tag{13}$$

where $\dot{q}_1$ is the rear wheel angular velocity.

The other quantities which have a fundamental role in the construction of generalized forces are partial velocities and partial angular velocities. In order to calculate the partial velocities of a point one should first find the velocities at that point. Then the coefficients of generalized speed in velocity terms are in fact the partial velocities.

Regarding the shrimp mechanism, we have already found the numerical value of velocities. Since we have only one generalized speed in this case; we can obtain the partial velocities and partial angular velocities as follows:

$$\mathbf{V}^{\mathbf{P}}{}_{\mathbf{1}} = \mathbf{V}^{\mathbf{P}} / u_1 \tag{14}$$

$$\boldsymbol{\omega}^{\mathbf{P}}{}_{\mathbf{1}} = \boldsymbol{\omega}^{\mathbf{P}} / u_1 \tag{15}$$

Now, it is possible to get the generalized active and generalized inertial forces. The only external forces and torques are linkage weights and motor torques. Thus for generalized active forces we have:

$$F_1 = \sum_{i=1}^{m} \mathbf{M_i} \cdot \boldsymbol{\omega}_{\mathbf{1}}{}^{\mathrm{W_i}} + m_i \mathbf{g} \cdot \mathbf{V}^{\mathbf{P}}{}_{\mathbf{1}} \tag{16}$$

where m_i represents the mass of different parts of the mechanism and W_i represents mechanism wheels; with torque $\mathbf{M_i}$ applied to each.

The generalized inertial force equations are derived as below:

$$F_1^* = \sum_{i=1}^{m} (-\boldsymbol{\alpha}^{\mathbf{B_i}}) \cdot \boldsymbol{\omega}_{\mathbf{1}}{}^{\mathrm{W_i}} + (-\mathbf{a}^{\mathbf{B_i}}) \cdot \mathbf{V}^{\mathbf{P}}{}_{\mathbf{1}} \tag{17}$$

where $\mathbf{B_i}$ represents different parts of the mechanism which counted as n and $\mathbf{a}^{\mathbf{B_i}}$ is the acceleration of the center of mass of part $\mathbf{B_i}$ and $\boldsymbol{\alpha}^{\mathbf{B_i}}$ is the angular acceleration of part $\mathbf{B_i}$. In this way the generalized inertia and generalized active forces are found. Using Eqs. 16 and 17 the following equation of motion is derived:

$$F_1 + F_1^* = 0 \tag{18}$$

Which is equivalent to

$$\sum_{i=1}^{m} \mathbf{M_i} \cdot \boldsymbol{\omega}_1^{\mathbf{W_i}} = \sum_{i=1}^{m} (-\boldsymbol{\alpha}^{\mathbf{B_i}}) \cdot \boldsymbol{\omega}_1^{\mathbf{W_i}} + \sum_{i=1}^{m} (-\mathbf{a}^{\mathbf{B_i}}) \cdot \mathbf{V^P}_1 - \sum_{i=1}^{m} m_i \mathbf{g} \cdot \mathbf{V^P}_1 \tag{19}$$

The right hand side of the Eq. 19 is known, provided we have the accelerations and angular accelerations. Thus an equation for applying torques is obtained. With an appropriate assumed relation between wheel torques, it is possible to calculate each wheel torque. In the shrimp mechanism the following relations are considered between different wheel torques:

$$\begin{aligned} M_2 &= \frac{N_2}{N_1} M_1 \\ M_3 &= \frac{N_3}{N_1} M_1 \\ M_4 &= \frac{N_4}{N_1} M_1 \end{aligned} \tag{20}$$

where N_i is normal force in the wheel-ground interface of *i*th wheel. This assumption is equivalent with considering more wheel torque as normal force increases. Eqs. 19 and 20 are sufficient for finding wheel torques.

The next step is to calculate the normal forces. Normal forces are among constraint forces and can not be seen in equation of motion; this is expected, as Kane's method is based on energy. Several methods exist to obtain these forces which have been used both in Eqs. 20 and the optimization objective function.

The method introduced in Ref. [12] in order to bring the constraint forces into evident, is to define a set of generalized speeds that violate the constraints. This results in an increase in the numbers of partial velocities and number of governing equations from which the constraint forces and moments are determined. In Ref. [13] the same issue is solved by introducing Lagrange-multiplier-like scalars to adjoin the constraint matrix with Kane's equation for holonomic systems. The resulting equations together with the constraint equations are solved for these scalars and generalized speeds and, hence, for the constraint forces and moments. This approach is suitable for small systems with few degrees of freedom.

Lesser [14] suggested the method of projecting the active and inertia forces and moments on the orthogonal complement space of the configuration space spanned by the partial angular and partial velocities. This requires finding the spanning set of vectors to this defined space and solving the resulting complementary equations for constraint forces and moments.

In this paper the first method was employed for developing the constraint forces (i. e. normal forces).

3 Optimization

To have a rough terrain rover with high mobility that can traverse through unstructured surfaces without loss of wheel-ground contact or slipping, optimization of wheel-ground contact forces is performed. Power efficiency is also considered in rover dynamical optimization. Due to dynamical nonlinearities of the problem, Genetic Algorithm (GA) is selected as the optimization method. GA is a directed random search technique which is applicable to non-linear problems [15].

3.1 Optimization Criteria

The most common optimization criterion used for rovers is the minimization of traction in the wheel-ground interface. To avoid wheel slipping, the ratio of traction to normal force in wheel-ground contact point should be lower than a specific value. A function R_i that represents this ratio can be used as follows:

$$R_i = \frac{T_i}{N_i} \tag{21}$$

where T_i is traction and N_i is normal force in the wheel-ground interface. R_i is also called wheel slip ratio.

For the shrimp rover with four wheels the objective function is selected as the sum of maximum of slip ratios in wheels. Our objective function is in the form of:

$$OF = \sum_{i=1}^{4} \max\{R_i\}\,. \tag{22}$$

3.2 Problem Constraints

There may be physical constraints for optimization problems. The first one is to keep all wheels in contact with the ground; i.e. normal contact forces should be greater than zero:

$$N_i > 0 \qquad \textit{for} \;\; i = 1...4 \tag{23}$$

The second constraint is that the ratio of traction force to normal force (i.e. R) should not be greater than the ground-wheel coulomb coefficient of friction:

$$R_i \le \mu \qquad \textit{for} \;\; i = 1...4\,. \tag{24}$$

4 Simulation Results

In this section, performance of the optimized shrimp rover and the first version of the rover (i.e. *CEDRA-I Shrimp Rover*) are compared. Since stairs are challenging terrains, in most cases they are considered as a standard testing rough terrain. In this research the terrain is chosen to be standard stairs. Path specifications are listed in Table 1. A sketch of the path and rover are shown in Fig. 3.

CEDRA-I Shrimp Rover is used for the first run. The Rover's specifications including geometric parameters and dynamic properties are listed in Table 2.

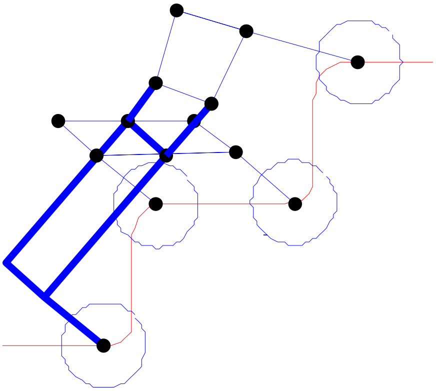
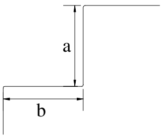


Fig. 3. Sketch of the simulated path and the rover

Table 1. Stairs geometric specifications

	*a*	0.25 m
	b	0.25 m

Traversing the stairs path, the rover is dynamically simulated. Dynamic parameters are obtained by solving dynamical equations.

Figure 4 illustrates the results of the simulation. This figure contains rover wheels slip ratio (i.e. *R* function in Eq. 21). As seen in the figure, the wheel-ground interface slip ratio is close to static coefficient of friction. High slip ratio can reduce the traction at wheel-ground interface; as a result, the rover mobility is reduced.

In the next step, the optimization is applied on the rover. The optimized rover traversed the path and dynamical parameters are obtained by solving dynamical equations obtained using Kane's method. Results containing wheel-ground interface slip ratio are shown in Fig. 5. The geometric parameters of the optimized rover are listed in Table 2.

Obviously the slip ratios are reduced considerably. It can be seen from comparison of Fig. 4 and Fig. 5 that, in some wheels the slip ratio is reduced up to 40 percent. Consequently, the traction at the wheel-ground interface is increased. This is equivalent to the increase in rover mobility that is an important point in rovers. Also it is inspected that there is no loss of contact in wheel-ground interface.

Table 2. *CEDRA-I Shrimp* rover and the optimized rover specifications

CEDRA-I Rover	
ITEM	**VALUE**
b	0.07 m
c	0.29 m
d	0.180 m
e	0.130 m
f	0.33 m
g	0.01 m
h	0.21 m
k	0.21 m
l	0.16 m
m	0.31 m
n	0.21 m
p	0.13 m
r1	0.05 m
r2	0.05 m
r3	0.05 m
r4	0.05 m
Mass	40 kg

Optimized Rover	
ITEM	**VALUE**
b	0.105 m
c	0.277 m
d	0.188 m
e	0.116 m
f	0.318 m
g	0.035 m
h	0.53 m
k	0.284 m
l	0.22 m
m	0.26 m
n	0.25 m
p	0.16 m
r1	0.07m
r2	0.07 m
r3	0.07 m
r4	0.07 m
Mass	40 kg

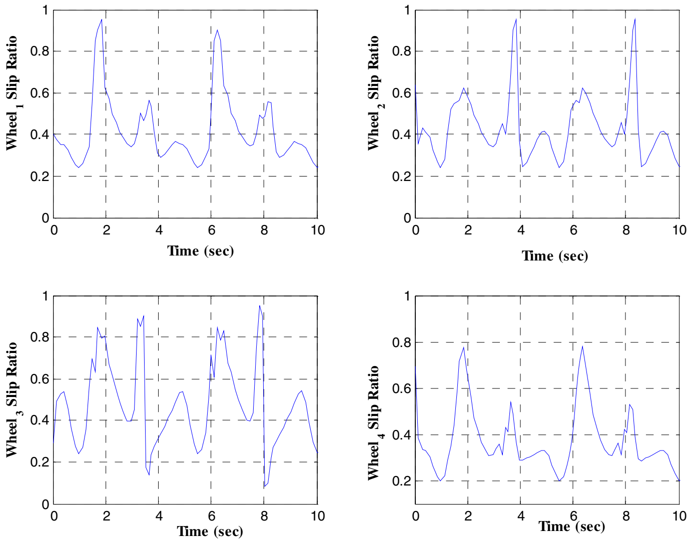


Fig. 4. Slip ratio of wheels for CEDRA-I Shrimp Rover

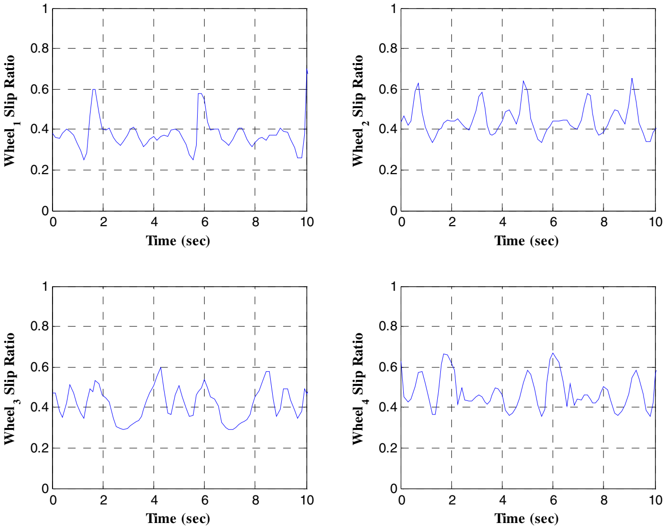


Fig. 5. Slip ratio of wheels for optimized rover

5 Conclusions

In this paper an innovative approach to dynamical optimization of rough terrain rovers is presented. Dynamical equations are obtained using Kane's method. Optimization is performed for the rover. Optimization criteria are the minimization of traction in rover wheel-ground interface. Analysis is applied on a 4-wheel rough terrain rover called *CEDRA-I Shrimp Rover*. Results show an improvement in rover traction, which has an important role in rover mobility.

Acknowledgements

This project was supported by a grant (82/51065) from the Industrial Development and Renovation Organization of Iran (IDRO). Many thanks are extended to Ms. Ronasi, Dr. Broomand, Dr. Moghaddam, Mr. Ahadi, Mr. Oryani and Ms. Rajabi, for their encouragement and support.

References

1. Tai M.: Modeling of Wheeled Mobile Robot on Rough Terrain ASME International Mechanical Engineering Congress. Washington D.C. (2003).
2. Iagnemma, K. , and Dubowsky, S.: Mobile Robot Rough-Terrain Control (RTC) for Planetary Exploration. Proceedings of the26th ASME Biennial Mechanisms and Robotics Conference, DETC (2000).
3. Kawabe T., Nakazawa M., Notsu I., Watanabe Y.: Sliding Mode Controller for Wheel Slip Ratio Control System. Journal of Vehicle System Dynamics. 5-6 (1997) 393-408.

4. Reister, D., Unseren, M.: Position and Constraint Force Control of a Vehicle with Two or More Steerable Drive Wheels. IEEE Transactions on Robotics and Automation. 9 (1993) 723-731.
5. Sreenivasan, S., and Waldron, K.: Displacement Analysis of an Actively Articulated Wheeled Vehicle Configuration with Extensions to Motion Planning on Uneven Terrain. Transactions of the ASME Journal of Mechanical Design. 118 (1996) 312-317.
6. Hacot, H.: Analysis and Traction Control of a Rocker-Bogie Planetary Rover. M.S. Thesis, Massachusetts Institute of Technology. Cambridge, MA (1998).
7. Meghdari A., Pishkenari H. N., Gaskarimahalle A. L., Mahboobi S. H., Karimi R.: Optimal Design and Fabrication of CEDRA Rescue Robot Using Genetic Algorithm. Proc. of the International Design Engineering Technical Conferences DETC. Salt Lake City, Utah (2004).
8. Kane T. R., Levinson D. A.: The use of Kane's Dynamical Equations in Robotics. Int. J. Robotics Research. 7 (1996) 333-342.
9. Estier T., Crausaz Y., Merminod B., Lauria M., Piguet R., Siegwart R.: An Innovative Space Rover with Extended Climbing Abilities. In Proceedings of Space & Robotics, the Fourth International Conference and Exposition on Robotics in Challenging Environments. Albuquerque, New Mexico (2000).
10. Volpe R., Balaram J., Ohm T., Ivlev R..: Rocky 7: A Next Generation Mars Rover Prototype. Journal of Advanced Robotics. 4 (1997).
11. Kemurdjian, A. L., Gromov, V., Mishkinyuk, V., Kucherenko, Sologub, P.: Small Marsokhod Configuration. International Conference on Robotics & Automation. Nice (1992).
12. Kane T. R., and Levinson D. A.: Dynamics: Theory and Applications. 1st edn. Series in Mechanical Engineering, McGraw-Hill, New York (1985).
13. Wang J. T., and Huston R. L.: Kane's Equations with Undetermined Multipliers-Application to Constrained Multibody Systems. Journal of Applies Mechanics. 2 (1987) 424-429.
14. Lesser M.: A Geometrical Interpretayion of Kane's Equations. Proceedings of the Royal Society of London, Series A: Mathematical and Physiscal Sciences. 1896 (1992) 69-87.
15. D.T. Pham, D. Karaboga: Intelligent Optimization Techniques. Springer-Verlag, New York (2000).

High Fidelity Tools for Rescue Robotics: Results and Perspectives

Stefano Carpin[1], Jijun Wang[2], Michael Lewis[2],
Andreas Birk[1], and Adam Jacoff[3]

[1] School of Engineering and Science
International University Bremen – Germany
[2] Department of Information Science and Telecommunications
University of Pittsburgh – USA
[3] Intelligent Systems Division
National Institute of Standards and Technology – USA

Abstract. USARSim is a high fidelity robot simulation tool based on a commercial game engine. We illustrate the overall structure of the simulator and we argue about its use as a bridging tool between the RoboCupRescue Real Robot League and the RoboCupRescue Simulation League. In particular we show some results concerning the validation of the system. Algorithms useful for the search and rescue task have been developed in the simulator and then executed on real robots providing encouraging results.

1 Introduction

Urban search and rescue (USAR) is a fast growing field that obtained great benefits from the RoboCup competition. Society needs robust, easy to deploy robotic systems for facing emergency situations. The range for potential applications is very wide. Fire fighters inspecting vehicles transporting hazardous materials involved in road accidents is one end, while locating people and coordinating big rescue teams after a major natural disaster like a earthquake or a tsunami is at the other side of the spectrum. The two leagues introduced in the Robocup competition somehow represent these two extremes. Currently, in the Real Robot League the issues being addressed concern mainly locomotion, sensing, mapping and localization. Up to now, very few attempts were made in the direction of autonomy and cooperation using teams of robots. The technical difficulties encountered while dealing with the formerly indicated aspects still dominate the scene. In the Simulation League, the problem is addressed from the other side. Large teams of heterogenous agents with high level capabilities have to be developed. Topics like coordination, distributed decision making, multiobjective optimization are some of the crisp matters being addressed. It is part of the overall vision that in the future the two scientific communities will move towards each other, and will eventually meet. It is nevertheless evident that this is not going to happen soon. Deploying a team of 20 autonomous robots performing a rescue task over an area of a few hundred square meters and for a time

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 301–311, 2006.

horizon of hours is beyond the current capacity. In this context, we illustrate a simulation project called USARSim that has been developed at the University of Pittsburgh. We envision that USARSim is the tool needed to foster and accelerate cooperation between the formerly described communities. On the one hand, USARSim allows a high fidelity simulation of real robots, and offers great flexibility when it comes to model new environments or hardware devices. On the other hand, the software allows the simulation of reasonably sized teams of agents. Section 2 describes the USARSim simulation software. Next, in section 3 we illustrate our current experience in using the USARSim software to develop algorithms to be used to control real robots. Finally, conclusions are offered in section 5.

2 USARSim

USARSim is a high fidelity simulation of USAR robots and environments intended as a research tool for the study of HRI and multi-robot coordination. USARSim supports HRI by accurately rendering user interface elements (particularly camera video), accurately representing robot automation and behavior, and accurately representing the remote environment that links the operator's awareness with the robot's behaviors. The current version of USARSim consists of: environmental models (levels) of the National Institute of Standards and Technology (NIST) Yellow, Orange, and Red Arenas, and Nike site which serve as standardized disaster environments for mobile robot studies, robot models of commercial and experimental robots, and sensor models. USARSim also provides users with the capabilities to build their own environments and robots. Its socket-based control API allows users to test their own control algorithms and user interfaces without additional programming. USARSim uses Epic Games' Unreal Engine 2 [1] to provide a high fidelity simulator at low cost. Unreal is one of the leading engines in the "first-person shooter" genre and is widely used in the gaming industry. It is also gaining a strong following in the academic community as more researchers use it in their work. Recent academic projects have included creating VR displays [2], studying AI techniques [3], and creating synthetic characters [4]. In addition to the egocentric perspective, there are several other features of the Unreal Engine that make it particularly appealing for HRI research.

- **Graphics:** The Unreal Engine provides fast, high-quality 3D scene rendering. It supports mesh, surface (texture) and lighting simulation, and can import models from other popular modeling tools such as Maya [5] and 3D Studio Max [6]. Moreover, its dynamic scene graph technology enables simulation of mirrors, glass and other semi-reflective surfaces. The high fidelity of these graphics allows the Unreal engine to simulate realistic camera video, the most critical feature in current approaches to human control of mobile robots.
- **Physics engine:** The Unreal Engine integrates MathEngine's Karma Engine [7] to support high fidelity rigid body simulation. The details of physical

simulation, including collision detection, joint, force and torque modeling are encapsulated within the high level game programming language. This feature lets the simulation replicate both the physical structure of the robot and its interaction with the environment.

- **Authoring tool:** The Unreal Engine provides a real-time design tool for developers to build their own 3D models and environments. The editing tool, UnrealEd, is fully integrated into Unreal Engine to provide users a what-you-see-is-what-you-get style authoring tool. UnrealEd permits HRI researchers to accurately model both robots and their environments.
- **Game Programming:** The Unreal Engine provides an object-oriented scripting language, UnrealScript, which supports state machine, time based execution, and networking on a programming language level. With UnrealScript, the rules of the simulation can be manipulated. This affords the ability to customize the interaction with the simulation to match the specifics of desired robot behaviors.
- **Networking:** The Unreal Engine uses an efficient client-server architecture to support multiple players. This embedded networking capability allows USARSim to support control of multiple robots without modification.

Figure 1 shows Unreal Engine components and the expandable library of robot-themed models and environments and control interfaces to acquire sensor data and issue commands we have added to create the USARSim simulation.

2.1 Robot Models

USARSim currently provides detailed models of six robots: the Pioneer P2AT and P2DX [8], iRobot ATRV-Jr, the Personal Exploration Rover (PER) [9], the Corky robot built for this project and a generic four-wheeled car. Figure 2 shows some of these simulated and real robots. These models were constructed by building the components of the robot and defining how these parts were connected using joints which serve as mechanical primitives for the Karma physics engine. Since the physics engine is mechanically accurate, the resulting movement of the aggregate robot is highly realistic. Karma uses a variety of computational strategies to simplify, speed up, and exclude non- interacting objects to achieve animation level speed without sacrificing physical fidelity. Because USARSim is intended for use by researchers from diverse backgrounds we have added a reconfigurable robot model to allow researchers to construct and customize their own robots without detailed mechanical modeling. Building a new robot model using this facility is as simple as 1) building geometric models for the robot, 2) configuring the robot model to specify the physical attributes of the robot and define how the chassis, parts and auxiliary items are connected to each other and 3) performing additional programming only if the robot needs features or behaviors not included in the robot model.

2.2 USAR Environments

USARSim includes detailed models of the NIST reference arenas [10], [11] and will soon include a replica of the fixed Nike site reference environment.

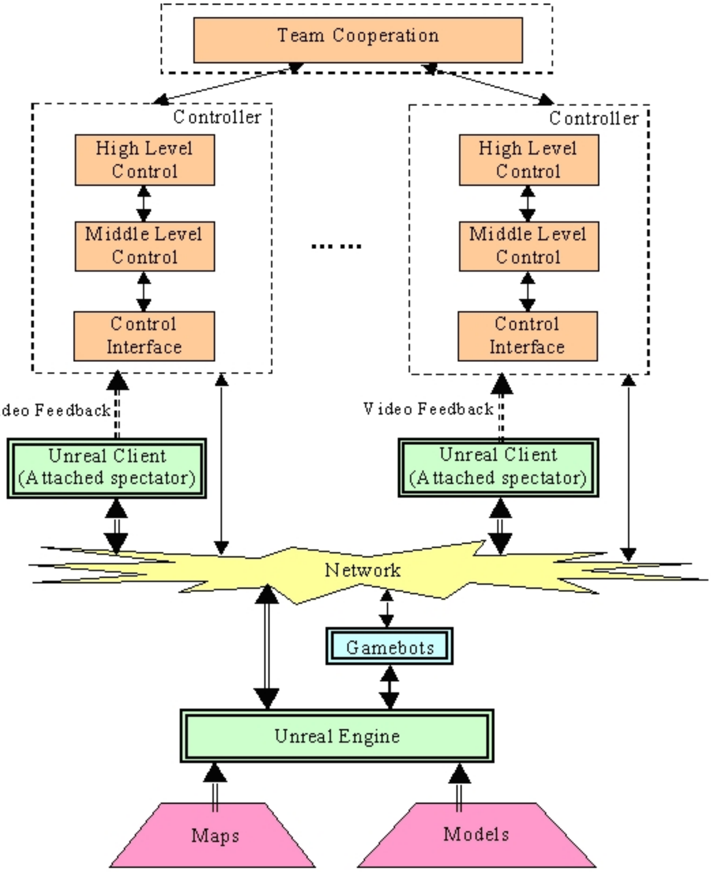


Fig. 1. System architecture

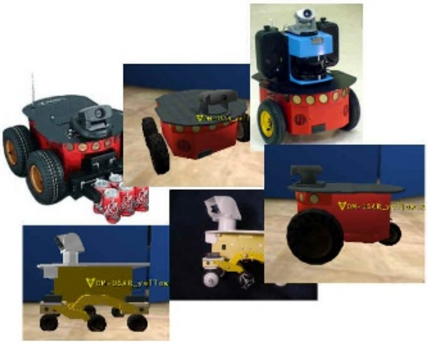


Fig. 2. Some robots in USARSim

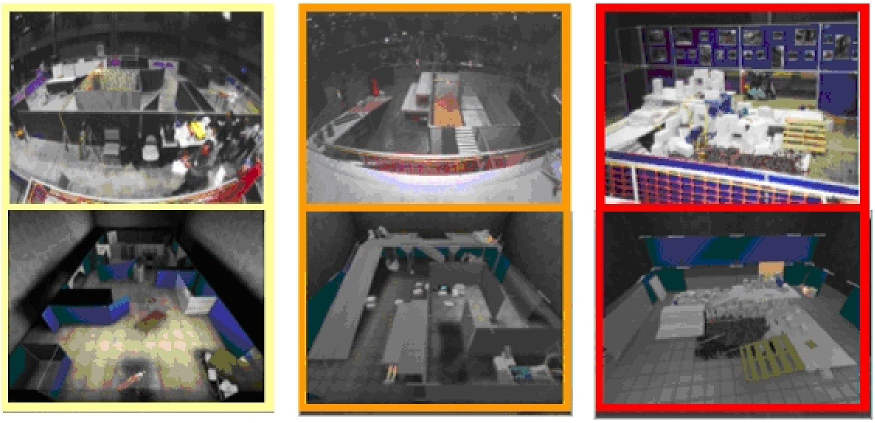


Fig. 3. The real and simulated USAR arenas

Significantly larger disaster environments are under development for the Virtual Robot USAR demonstration at RoboCup 2005. To achieve high fidelity simulation, 3D CAD models of the real arenas were imported into Unreal and decorated with texture maps generated from digital photos of the actual environments. This ensures geometric compatibility and correspondence between camera views from the simulation and actual arena. In addition to this basic structure, the simulated environments include the real and simulated USAR arenas (figure 3). A collection of virtual panels, frames, and other parts used to construct the portable arenas are included with USARSim. These efforts attempt to simulate specific, physical spaces. Using the UnrealEd tool, it is possible to rearrange these elements to quickly develop alternate USAR layouts in much the same way the arenas are reconfigured during USAR contests.

2.3 Sensor Models

Sensors are a critical part of the simulation because they both provide the basis for simulating automation and link the operator to the remote environment. USARSim simulates sensors by programmatically manipulating objects in the Unreal Engine. For example, sonar and laser sensors can be modeled by querying the engine for the distance given the sensor's position and orientation to the first object encountered. To achieve high fidelity simulation, noise and data distortion are added to the sensor models by introducing random error and tailoring the data using a distortion curve. Three kinds of sensors are simulated in USARSim.

- Proprioceptive sensors: including battery state and headlight state.
- Position estimation sensors : including location, rotation and velocity sensors.
- Perception sensors: including sonar, laser, sound and pan-tilt-zoom cameras.

USARSim defines a hierarchical architecture (figure 4) to build sensor models. A sensor class defines a type of sensor. Every sensor is defined by a set of attributes

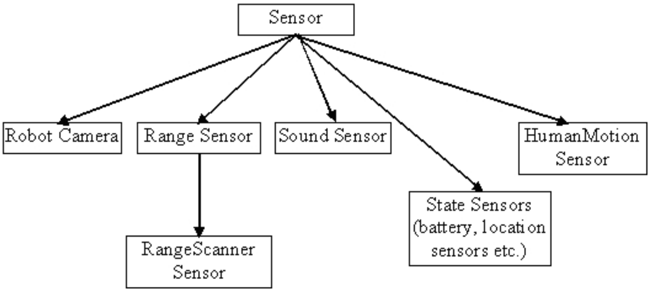


Fig. 4. Sensor Hierarchy Chart

stored in a configuration file. For example, perception sensors are commonly specified by range, resolution, and field-of-view. To get a sensor with specified capability, we can either directly configure a sensor class or derive a new sensor from an existing sensor class. Once the sensor is configured, it can be added to a robot model, by simply including a line in the robot's configuration file. A sensor is mounted on a robot specified by a name, position where it is mounted and the direction that it faces.

2.4 Simulating Video

Cameras provide the most powerful perceptual link to the remote environment and merit a separate discussion. The scenes viewed from the simulated camera are acquired by attaching a spectator, a special kind of disembodied player, to the camera mount on the robot. USARSim provides two ways to simulate camera feedback. The most direct is to use the Unreal Client as video feedback, either as a separate sensor panel or embedded into the user interface. While this approach is the simplest, the Unreal Client provides a higher frame rate than is likely to be achieved in a real robotic system and is not accessible to the image processing routines often used in robotics. The second method involves intermittently capturing scenes from the Unreal Client and using these pictures as video feedback an approach that is very close to how a real camera works. USARSim includes a separate image server that runs alongside the Unreal Client. This server captures pictures in raw or jpeg format and sends them over the network to the user interface. Using this image server, researchers are able to better tune the properties of the camera, specifying the desired frame rate, image format and communication properties to match the camera being simulated.

3 Validation: Preliminary Results

Mapping is one of the fundamental issues when rescue robots are used to assist humans operating in buildings. Maps help rescue operators while finding victims and avoiding dangerous areas. To this end, at the International University Bremen (IUB) we investigated different mapping strategies with real robots.

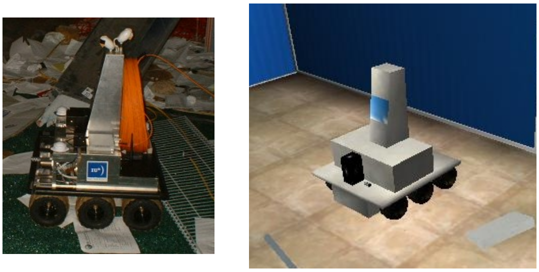


Fig. 5. On the left, the rescue platform developed at IUB. On the right, the model of the same robot while performing into the simulated yellow arena.

In particular, we focused on grid based maps, and we also tackled the problem of multi-robot map merging [12]. Recently, however, we have decided to move towards other approaches, like SLAM [13], that require the identification of features. In this context, we started to develop algorithms to extract natural landmarks in unstructured environments. The problem of features extraction has been faced first in simulation and currently on the real robots. The IUB rescue robots are self developed systems (see figure 5). The platform has a six-wheels differential drive, and is equipped with a number of different sensors, including a proximity range finder, odometry, an orientation sensor, and a set of different cameras [14]. Victim detection is human supervised, and is assisted by an infrared camera and a CO_2 sensor. Mapping is performed using the robot's pose (provided by odometry and orientation sensor) and the data coming from the range finder. Developing the model of the IUB robot for the USARSim software has been a straightforward process. USARSim is shipped with the models of several robots based on differential drive platforms. This allowed us to quickly develop the kinematic model of the robot. The proximity range sensor provided in the USARSim simulation environment can be configured in terms of the number of beams used to sample the sweeped area, the maximum reachable distance, and the noise. The real sensor we use (Hokuyo PB9-11) sweeps an area of 162 degrees with 91 beams. Its detection distance is 3 meters, and we experimentally determined that under the conditions found in the IUB rescue arena the signal to noise ratio is about 30 dB. These properties can be easily transferred to the parameters controlling the simulated range finder. We first run the simulated robot into the model of the IUB rescue arena [15] and gathered the data produced by the simulated proximity range finder. Then, we run the real robot into the real arena and collected the same data. Features extraction at the moment is based on the Hough transform, a widely used tool coming from image processing and used also in robotics. For line detection, the parameterized form of the line is used, i.e. $\rho = xcos\theta + ysin\theta$ where ρ is the perpendicular distance of the line from the origin and θ is the angle that the normal makes with the x axis. The basic idea is that the Hough Transform maps points from the Cartesian space to the (ρ,θ) Hough space. Each point in the Cartesian space corresponds to a sinusoidal curve in the Hough Space. Once the hough transform has been performed on the image, a simple voting scheme can be set up in the hough space. In this

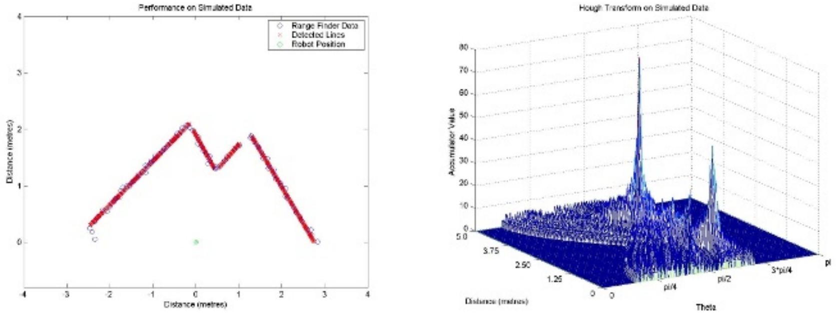


Fig. 6. On the left, the data collected with USARSim. On the right, the Hough transform calculated on the same data.

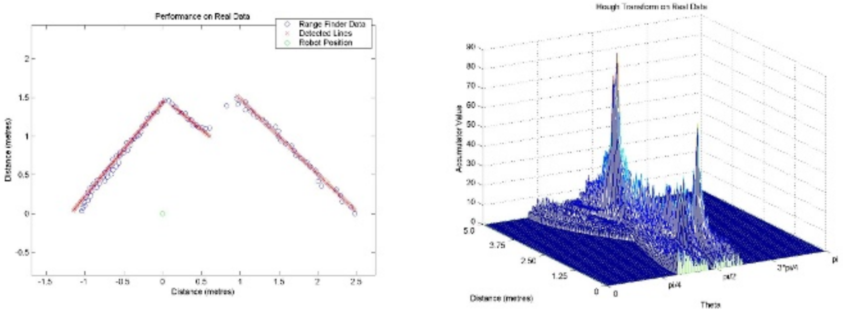


Fig. 7. On the left, the data collected with the real robot. On the right, the Hough transform calculated on the same data.

way, for a given range of values for ρ and θ, each point in the Cartesian space is mapped to the hough space which accumulates the values in a two-dimensional histogram. Local maxima of this histogram correspond to lines detected in the image. In the case of an image, local maxima can easily be found by an appropriate hill climbing algorithm. However in the case of range finder data, we have only a few data points and a rather vast space for ρ and θ. This results in a very sparse accumulator for which hill climbing is not ideally suited. So in this case, it makes sense to find the global maximum, remove those scan points that contribute to this line, and repeat the procedure until the global maximum drops below a certain threshold of the initial maximum. Note that the discretization of the Hough space must be tuned according to the problem. If the discretization is too fine, we might find several local maxima too close to each other in the Hough space. However, the degree of discretization directly affects the precision of the detected lines, so it should not be set too low. The following figures show a comparison between the data collected with the simulator (figure 6) and with the real robot (figure 7), as well as the corresponding Hough transforms. The fine tuning of parameters was completely performed within USARSim. No change was necessary when the code was later used to perform the same processing on real data.

4 Future Work

Though the initial results with USARSim appear promising, further developments are needed in order to make it a really effective tool. Specifically, more models of robot sensors are under planning and will be developed. Given the importance that video input has in real robots, the improvement of its simulation is a must. For example stereo vision, which proved to be highly successful in the Real Robots League will be available inside USARSim as well. Along the same lines, the possibility to simulate infrared cameras has to be considered, because of their high potential for victim recognition. Also other sensors like CO_2 probes, thermometers, and similar will be included.

Along the same lines, more robot models will be developed. This is particularly important in the search and rescue domain, where custom platforms with high mobility are often developed to overcome obstacles. With this respect, one of the more urgent aspects to address is the simulation of tracked vehicles. Finally, in order to make the migration of code developed under USARSim towards real robots an easy task, common interfaces need to be developed. This is already partially achieved by the Player [16] interface currently available, though at the moment few experiments in this direction have been performed.

5 Conclusions

In this paper we introduced USARSim, a high fidelity simulation tool which supports development of advanced robotic capabilities in complex environments such as those found in the urban search and rescue domain. We showed how USARSim's detailed models of the arenas used to host the RoboCupRescue Real Robot League competitions, along with kinematically correct robot models and simulated sensors, can provide a rich environment for development of robotic behaviors and innovative robot designs. We further showed initial experimental results captured at IUB which demonstrate that a high level of correlation can be obtained between the simulated environment and real arenas. This was shown through a feature extraction example using a simulated range sensor within the simulated environment and a real range sensor deployed on a robot within an actual arena. Such correlation reinforces expectations that algorithms developed, and shown to be effective, within the simulated environment can be transferred to real robots with reasonable expectations of effectiveness; thus USARSim can reduce the need for costly and problematic robot hardware to support iterative development and testing practices.

USARSim's usefulness in this regard will be on display to the community at this year's RoboCup 2005 in Osaka, Japan, where it will be demonstrated as the basis for a new league to compliment the existing RoboCupRescue leagues: the

Real Robot League and the Simulation League. Specific rules for this proposed league, called the RoboCupRescue Virtual Robot, are under development. But the performance metric used in the Real Robot League has been adopted to maintain close ties between league goals and approaches. Existing models of the rescue arenas housed at NIST, IUB, and elsewhere are already available for dissemination, along with several robot models and sensors described previously in this paper. The plan is for each year's competition to feature an entire building, with partially and fully collapsed sections, which will be made available as practice environments after the competition. This proposed league, if adopted after the Osaka demonstration, would provide a logical link between the required perception and negotiation of physical environments within the Real Robot League arenas, and the citywide responder allocation tasks associated with the Simulation League. The goal is to ultimately combine efforts in all three levels of abstraction to help develop and demonstrate comprehensive emergency response capabilities across a city and within particular buildings.

Acknowledgments

The authors thank the numerous students who conducted the preliminary validation at the International University Bremen and at the University of Pittsburgh.

References

1. Epic games: Unreal engine. www.epicgames.com (2003)
2. Jacobson, J., Hwang, Z.: Unreal tournament for immersive interactive theater. Communications of the ACM **45** (2002)
3. Laird, J.: Research in human-level ai using computer games. Communications of the ACM **45** (2003) 32–35
4. Young, M., Riedl, M.: Towards an architecture for intelligent control of narrative in interactive virtual worlds. In: ACM Conference on Intelligent User Interfaces. (2003)
5. Aliaswavefront: Maya. http://www.alias.com/eng/products-services/maya/index.shtml (2004)
6. Discreet: 3d studio max. http://www4.discreet.com/3dsmax (2004)
7. Karma: Mathengine karma user guide. http://udn.epicgames.com/Two/KarmaReference/KarmaUserGuide.pdf (2003)
8. Activemedia: Pioneer. http://www.activrobots.com/ (2005)
9. Nourbakhsh, I., Hamner, E., Porter, E., Dunlavey, B., Ayoob, E., Hsiu, T., Lotter, M., Shelly, S.: The design of a highly reliable robot for unmediated museum interaction. In: Proceedings of the IEEE International Conference on Robotics and Automation. (2005)
10. Jacoff, A., Messina, E., Evans, J.: Performance evaluation of autonomous mobile robots. Industrial Robot: An International Journal **29** (2002)
11. Jacoff, A., Messina, E., Weiss, B., Tadokoro, S., Nakagawa, Y.: Test arenas and performance metrics for urban search and rescue robots. In: Proceedings of the Intelligent and Robotic Systems (IROS) Conference. (2003)

12. Carpin, S., Birk, A.: Stochastic map merging in rescue environments. In: Robocup 2004: Robot Soccer World Cup VIII. Springer (2005)
13. Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H., , Csorba., M.: A solution to the simultaneous localisation and map building (slam) problem. IEEE Transactions of Robotics and Automation **17** (2001) 229–241
14. Birk, A.: The IUB 2004 rescue robot team. In: RoboCup 2004: Robot Soccer World Cup VIII. Springer (2005)
15. Birk, A.: The IUB rescue arena, a testbed for rescue robots research. In: IEEE International Workshop on Safety, Security, and Rescue Robotics, SSRR'04. (2004)
16. Vaughan, R., Gerkey, B., Howard, A.: On device abstractions for portable, reusable robot code. In: Proceedings of the IEEE/RSJ IROS. (2003) 2421–2427

Localizing Victims Through Sound and Probabilistic Grid Maps in an Urban Search and Rescue Scenario

Holger Kenn[1] and Andreas Pfeil[2]

[1] Technologie-Zentrum Informatik, Universität Bremen, Germany
kenn@tzi.de
http://www.tzi.de/
[2] School of Engineering and Science, International University Bremen, Germany
a.pfeil@iu-bremen.de
http://www.iu-bremen.de/

Abstract. Sound source localization can be used in the Robocup Rescue Robots League as a sensor that is capable to autonomously detect victims that emit sound. Using differential time of flight measurements through energy cross-spectrum evaluation of the sound signals, the angular direction to multiple sound sources can be determined with a pair of microphones for SNRs better than -8dB. Assuming that the robot pose is known, this information is sufficient to create probabilistic occupancy grid map of the sound sources in the environment and thus localize the victims in a global map. This has been demonstrated using example measurements in an urban search and rescue scenario.

1 Introduction

Urban search and rescue robotics has established itself as a new field for teleoperated and autonomous robots in the recent years. One of the major advances of the field has been the establishment of regular competition-based benchmarks such as the Robocup Rescue Robots league. ([2],[3],[4]) As the competition is using a scoring system that is adapted to new challenges each year, the focus within the competition can be shifted slowly from teleoperated systems with a main focus on mechanical engineering and mobility issues to more advanced systems with limited autonomy up to fully autonomous systems. Currently, most teams rely on teleoperation and visual feedback through onboard cameras. In the last year, mapping of the environment has become a standard capability of many teams, but victim localization is still done by the operator by using the visual feedback from the teleoperated robot. The development of sensors capable of automatic victim identification, localization and state assessment is therefore the next step towards fully autonomous rescue robots.

This paper presents a novel approach for an inexpensive victim localization sensor based on a stereo microphone pair.

The IUB Robocup Rescue team has been competing in the Robocup Rescue Robot League competitions in Fukuoka [5] and Padova [6].

All disaster scenarios set up at these competitions contain a number of victim dummies that have human appearance and are equiped with detectable features such as movement, sound, body heat and CO_2 emission. The performance of each team is evaluated through a common scoring function that analyzes the quality of the information

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 312–322, 2006.

gained and scores it against the number of human operators used by the team. This function thus both rewards autonomy and high-quality localization and multi-sensoric assessment of victims and their state.

During the 2002 competition, it became clear that the onboard autio sensor was useful within the competition as it was the only sensor that was capable of locating invisibly trapped or entombed victims. This finding has been one of the reasons for the extension of the robots with additional sensors with similar capabilities, such as body-heat detection through thermographic cameras.

With the introduction of the new control middleware FASTRobots [7] in the 2003 robot system, it became possible to add more sensors to the robot platform, first for non-victim related tasks such as localization and environment mapping. The generated LADAR-based map provided to be useful for robot localizaion [8]. However, as no automatic victim localizing sensor was available, the localization and identification of the victim dummies was still performed manually by the operator. In order to do this, the operator would carefully analyze all available sensors including the sound from the microphones, then note down the perceived signs of the presence and state of the victim in a paper victim sheet and then mark the victims location by using a mouse to click on the approximate position of the victim next in the LADAR map that is displayed on his operator control station screen together with the robots current position and orientation. This process is time-consuming and error-prone.

For automatic victim localization with bitmap sensors such as the visible light and thermographic cameras, computer vision based approaches may be used. However, these approaches are computationally expensive and their performance in the highly unstructured environment of a robocup rescue scenario is hard to predict.

The sensor described in this paper is capable of localizing a sound source in a global probabilistic occupancy grid map of the environment. These sources still have to be manually inspected and eventually identified as victims using the onboard cameras and other sensors but as their location is known now, their localization in the global map will be much more precise. The approach can easily be distributed over multiple robots, as no strict timing requirements for the acquisition of the sound data is required. To allow this, it is assumed that the sound sources stay at fixed positions in the environment.

The remaining part of this paper is structured as follows: The second section gives an overview over the system setup and an introduction into the theory of sound source localization. The third section describes an experiment to estimate the performance of the obtained sound source localization and shows that the measurements obtained are close to the theoretical boundaries. These results are then used to simulate the performance of a probabilistic map based on a occupancy grid. Then, an experiment is described that uses an existing robot to gather data in a rescue scenario to produce such a probabilistic map. The last section discusses the results obtained so far.

2 System Overview and Theoretical Analysis

A typical robot system used for Robocup rescue consists of one or more mobile robots and one or more operator control stations. For the IUB Robocup Rescue system, the communication between the mobile robots and the control station is implemented through the FAST-Robots middleware[7]. Each mobile robot runs an instance of the

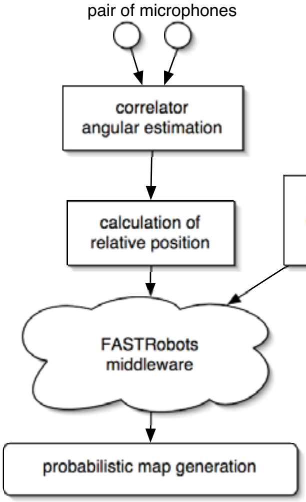

pair of microphones
correlator angular estimation
calculation of relative position
Other sensors (LADAR, pose tracking)
FASTRobots middleware
probabilistic map generation

Fig. 1. An overview of the sound source localisation system

platform component of the framework that instantiates multiple sensor and actuator driver objects that communicate with the corresponding sensor and actuator hardware. The platform communicates with its counterpart, the controlstation component running on the control stations through a TCP/IP network. The controlstation visualizes sensor data coming from the platform and transmits control commands to the platform.

The sensor described in this paper can easily be accomodated by this framework. Figure 1 shows an overview of the components that are part of the sound localization sensor system.

2.1 Microphone Phase Array

The problem of sound source localization can be solved in different ways. One approach is sound source localization based on beamforming techniques, such an approach has for example been presented in [1]. However, our approach is using the cross-energy spectrum of signals recorded at microphone pairs to evaluate the sound source directions. This is computaionally less expensive for a small number of microphones and allows for easier detection of multiple sound sources. This approach is explained in the following paragraph.

A simple way to model the localisation of a sound source (sometimes also called “passive sonar”) by using multiple microphones is the so-called linar microphone phase array. In this model, a number of microphones are located equidinstant along the x axis of our coordinate system. It is then possible to determine the position of a sound source in the coordinate system using differential time-of-flight measurements, i.e. the time difference for the signal of the same sound source to arrive at different microphones. This system however cannot detect the correct sign of the y coordinate, i.e. it cannot distinguish between sound sources in positive or negative y direction. When the array is limited to only two microphones, the position of the sound source can only be restricted

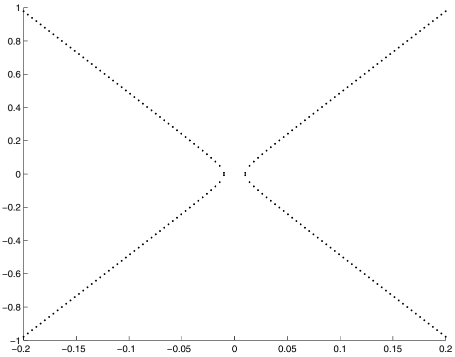


Fig. 2. Two hyperbolas indicating the possible location of the sound source for a given time-of-flight difference δt and $-\delta t$

to a hyperbola given by the path length difference of the sound signals from the source to the microphones. These hyperbolas can be approximated by their asymptotes for sound sources that are further away than the distance of the microphones, i.e. the direction to the sound source can be determined.

To determine the time delay between two incoming sound signals at the microphones, the cross-energy spectrum of the two signals is evaluated. Identical, but shifted signal portions produce peaks in this spectrum; the position of the peak is an indicator for the delay between the first and the second occurance of the same signal portion in the different signals. Should there be several distinct sound sources with different relative delays, one peak for every sound source can be detected. For the remainder of this section, it will be assumes that only a single sound source is being localized. It will be shown later that with multiple sound sources can be dealt with using probabilistic occupancy grids.

In order to process the signal from the microphones, it is sampled with the highest possible time resolution that the hardware offers. For the standard audio interfaces of PCs, this is typically 48khz, i.e. 20.8 microseconds between two samples. This consequentially is the shortest delay Δt_{min} that the system can distinguish. Together with the speed of sound c, this results in a quantization of the differential distance measurements into $c\Delta t_{min}$.

As the distance difference δ is quantized, the angular resolution of the microphone pair detector significantly differs for different angular areas. Angles near the direction of the normal vector, i.e. “in front” of the microphone pair, can be measured with a fine resolution and angles in the direction of the line connecting the microphone pair, i.e. outside of the ’focal region’ can only be measured with a high uncertainty.

2.2 Angular Resolution

The resolution of the localization is strongly depending on the resolution of δ. Given a sampling frequency f_s and the speed of sound c, the maximal time difference of k

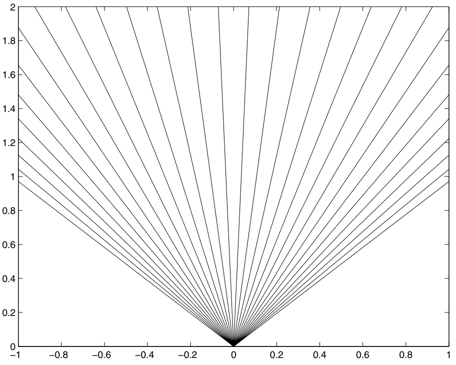


Fig. 3. The angular sectors that can be distinguished by one microphone pair 10cm apart, using 48kHz sampling frequency

samples is reached, when a signal is coming from the x-axis outside of the microphone pair. It is:

$$k = \frac{m \cdot f_s}{c} \tag{1}$$

δ varies from $-k$ to k samples.

Evaluating the angles for all possible k, the half-circle in front of the microphone pair can be divided into different zones, a sample resolution is shown in figure 3.

With several angular measurements from different positions, the position of the sound source can be determined through triangulation.

Assuming that the sound source is immobile, these angular measurements can as well be done sequentially by one robot only. The robot needs to measure at one point, move for a precise distance and measure the angle again. Using both angles and the base length, triangulation can be performed.

Assuming that the current pose of the robot is known, the system has sufficient information to create a probabilistic occupancy grid map [12] of the environment in a world coordinate system. Unlike the occupancy grid map used for robot navigation[8], this map does not contain information about the probability of cells being occupied by obstacles but with the probability of cells being the location of a sound source.

This type of map has been chosen over other approaches to probabilitstic mapping ([9],[10],[11] or see [13] for an excellent overview of the topic) as we assume that it is hard to extract features from the sensory input that could be redetected in the future. Moreover, the location of sound sources will not provide much structure as the location of walls in an office environment would give us. As this sensor is not intended for robot self-localization but only for sound source localization, it is assumed that an accurate estimation of the current pose of the robot is provided by other means. Note that this information could be provided through other means of probabilistic mapping and localization such as SLAM[11], but this mapping would then use other sensors such as LADAR.

The probabilistic map building algorithm is implemented in a straight-forward way: For every grid cell a value is calculated that represents its change in probability of being a sound source based on the current sensor data and this value is added to the current value stored for the grid cell.

The calculation of this change in probability does not only depend on the current sensor value but also on the properties of the sensor, i.e. on a sensor model. Here we assume that the sensor only gives good information for sound sources that are neither too faint (i.e. far away) nor are outside of the focus area where angular information is unreliable. Information concerning these areas is ignored.

If a sound source is located within the focus area of the sensor, its signal energy level is compared against a threshold T. If there is no signal higher than the treshold, the angular area reaching from the robot to a constant maximum reliability distance D is considered free of sound sources and every cell that has its centerpoint in this angular area receives a negative probabiltity change $-\varDelta$. If a sound source is detected in the angular area, every cell receives a positive probability change $\varDelta$. The probability values in the cells are then updated and limited to reasonable positive and negative maxima P_{MAX} and P_{MIN}.

Initially, all cells are initialized with a value of 0 that corresponds to a maximum of uncertainty for this cell, we neither know that it is a sound source nor we know that it is one.

It can easily be seen that the occupancy grid can solve the triangulation from two different robot poses provided that the sound source is within the detection range from both poses. If the robot is in the first pose A, it will increase the probability value of all cells between its location and its detection range in the direction of the sound source. All other cells within the detection range will receive a decrease in probability value. After a number of sensor readings are analyzed, the probability value for all cells between the robots current position and the sound source will converge to a value of P_{MAX} and all other cells of the grid will either remain 0 or will converge to P_{MIN}. If the robot is now moved to a pose B and if the sound source is still in the detection range of the robot, it will further increase the probability value in all cells in between of the current position of the robot and the sound source and it will decrease the probability value for all cells that are not in the direction of the sound source, thus the probability value of all cells in the proximity of the sound source will remain at P_{MAX} and all other cells will either converge to P_{MIN} or remain 0.

Unfortunately, a sector that has received a positive probability from pose A and is not in the detection range from pose B will remain with P_{MAX} probability value. However, this value is misleading as it only depends on a single measurement and therefore is not a true triangulated value. These sectors would lead to false positives, i.e. the detection of a sound source when there is none. In order to eliminate these false positives, additional measures have to be taken. A true triangulation consists of two measurements that use different angular directions to establish the triangulation. To distinguish true triangulations from false positives, the robot taking the measurement and incrementing the probability value in a cell additionally computes an angular sector ID in world coordinates. This angular sector ID is an integer that numbers the angular sectors of the semicircle from 0 to AS_{MAX} so that every direction gets a distinct ID. If a robot finds a different sector ID in the grid cell it is about to increment, it sets a flag in the cell indicating that it contains the result of a true triangulation.

This algorithm uses a number of parameters. The parameters that specify the size of the distinguished angular areas are determined by the geometric properties and the sampling frequency of the sensor. The treshold energy T and the reliability distance D parameters are dependent on the properties of the transmission system formed by the sound sources to be detected, the transmission medium and the microphones. The Parameters Δ, P_{MAX} and P_{MIN} determine the number of iterations that are needed for convergence. Additionally, the model described here assignes the same probability value increase to all grid cells in a sector. This does not reflect the real probabilities as the sector becomes wider when the cells are further away from the sensor. Consequently an individual cell that is further away should receive a linearly lower probability increase than a cell that is close to the sensor, but the simulations have shown that for rather small angular sensors, a fixed value is a reasonable approximation.

3 Experimental Results

The perfomance of the whole system was evaluated using a combination of simulations and measurements.

First, the performance of the angular detector in the presence of (white-gaussian) background noise was simulated and it was concluded that the system is rather immune to this kind of disturbances, provided that the SNR at the receiver is above $-8dB$.[1] Then, the predicted angular resolution of the sensor was verified in an experiment. In this experiment, a mobile sound source and the microphone pair were set up on a desk. In this setting, all angular sectors could be detected correctly. (see [14]).

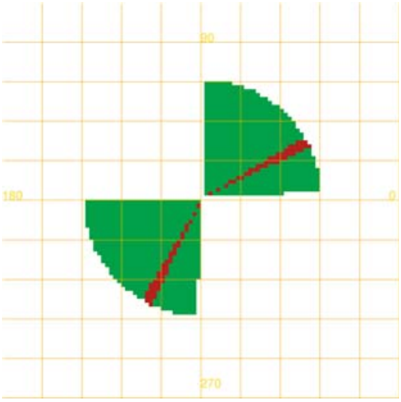


Fig. 4. One of the zones that can be distinguished by the sensor

3.1 Probabilistic Mapping

In order to to estimate the performance of the sensor in a probabilistic grid map, a sensor model has been derived from the data gained so far. The sensor model has a number of

[1] Altough white-gaussian background noise is far from realistic, it is a good test pattern for this situationas it has minimal cross-correlation, any correlated noise would result in additional sound sources being detected with only minor impact on the detection of the primary source.

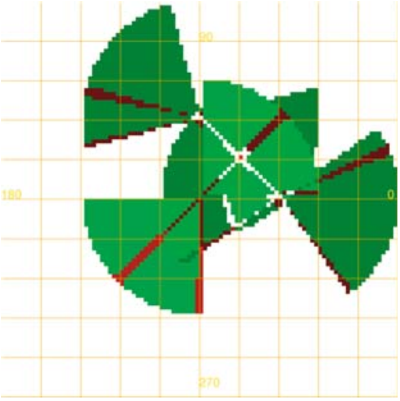


Fig. 5. A simulation of three sensor readings of a single sound source with a simple sensor model

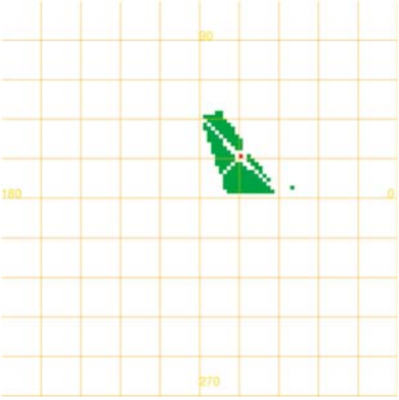


Fig. 6. A simulation of three sensor readings of a single sound source with a better sensor model using a touchcount filter

different zones that can be distinguished, each zone consisting of two angular sectors in positive and negative y direction as shown in Figure 4. To produce this figure, a sensor in the origin with a normal orientation of 45 degrees and 16 distinguishable angular sectors and a sound source at position x=2/y=1 was simulated. As the sensor cannot distinguish the exact position of the sensor in the zone, the probability of the presence in the zone is uniformly increased (red areas) and the probability of it not being in any other zone is uniformily decreased (green areas). The sensor is assumed to have a fixed range and for sources that are further away, it is assumed that the source is lost in the background noise, so it will not be detected. From the simulation result, it can be seen that a single sensor measurement is quite ambiguous.

In Figure 5, the simulation results for a sound source from three different sensor positions are shown. In this case, the sound source at position x=1/y=1 is clearly indicated with a positive probability. However, there are other parts of the map that receive positive probability. This occurs due to the fact that these areas are only covered by a single sensor reading, so the probability is increased by the sensor model of that one sensor reading, but is never decreased by the model of another sensor reading. This sensor model is formally correct, as there could be indeed three independent sources that are each only detectable by a single sensor. However, it is much more likely that only a single source creates the sensor readings. Therefore, we add an additional counter to each cell of the probabilistic map that counts how many sensor readings have contributed to the final value of the cell. By comparing this value against a threshold and filtering the result by this, we obtain the simulation result shown in Figure 6. Here the sound source can clearly be distinguished as the single point with positive probability that remains.

3.2 Full Sytem Experiment

To test the performance of the mapping system under the intended working conditions, a robot was placed in a scenario with one sound source present (see Figure 7). It was driven into 4 different poses and the perceived sound recorded. The data was analyzed and fed to the mapping algorithm yielding the map shown in Figure 8. The position of the robot were obtained by measuring the marks after the robot had been removed, thus eliminating errors created by the self-localization system of the robot.

It is important to note that, due to a design difference in the sensor system, this robot is using a sensor base length of $m = 20cm$, which is wider than in the systems used for simulation and thus leads to 57 distinguishable sectors. Using this high number of sectors did not improve results, so the angular resolution was then reduced again by joining several neighboring sectors to produce results comparable to the simulation. Additionally, the sensor range in the sensor model was increased as it was found that the attenuation model in simulation was overestimating, so sound sources further away could be detected by the sensor.

On Figure 8, it can be seen that the region detected as sound source is much wider than in the theoretical simulation. This can be explained by the fact that the robot position is further away from the sound source than in the simulated case and that the

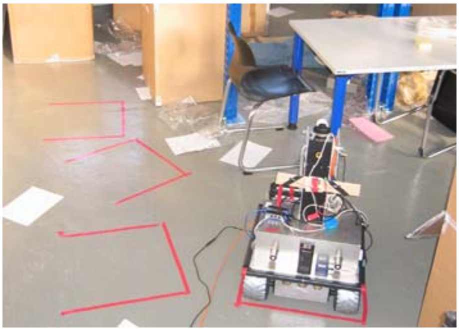


Fig. 7. An image of the scenario. The robot pose is marked with red tape, the open side of the rectangle being the back of the robot.

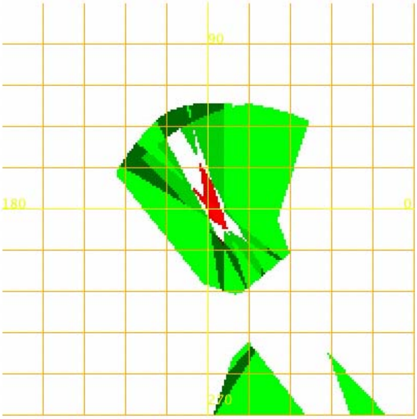


Fig. 8. The map created from real sensor readings. There is one source present and measurements have been taken from 4 different poses.

robots were all measuring the sound source from similar positions, this leads to a fuzzy and elongated localization. Additionally, it was found that the angular resolution of the sensor and the resolution of the grid map are important parameters to be tuned. In this case, a map resolution of 5 centimeters per grid cell and an angular resolution of 7.5 degrees produced the best results, for finer angular resolutions an more coarse grids, the sensor beams would not overlap sufficiently to allow for a good detection.

4 Conclusion

The problem of automatic victim localization in RoboCupRescue has been presented. A solution using microphones mounted on mobile robots and differential time-of-flight measurements of sound has been simulated and its accuracy shown to be sufficient in a simple experiment. A mapping algorithm using occupancy grids has been presented based on the experimental finding and it has been shown in simulation that is able to localize a sound source in a global map.

The next step will be the implementation of the sensor on a robot of the IUB Robocup Rescue team and the comparision of the simulation results with the real performance of the sensor. This will be an improvement over current victim localization techniques that are entirely based on human operators.

Acknowledgment

The measurements for this work have been made at the North-European Robocup Urban Search and Rescue Arena at the International University Bremen. The authors espechially thank Prof. Birk at IUB for his support.

References

1. Mattos, L., Grant, E.:Passive Sonar Applications: Target Tracking and Navigation of an Autonomous Robot. Proceedings of the IEEE International Conference on Robotics and Automation. IEEE Press, 2004.
2. Kitano, H., Tadokoro, S.: Robocup rescue. a grand challenge for multiagent and intelligent systems. AI Magazine 22 (2001) 39-52
3. Takahashi, T., Tadokoro: Working with robots in disasters. IEEE Robotics and Automation Magazine 9 (2002) 34-39
4. Osuka, K., Murphy, R., Schultz, A.: Usar competitions for physically situated robots. IEEE Robotics and Automation Magazine 9 (2002) 26-33
5. A. Birk, H. Kenn et al., The IUB 2002 Smallsize League Team, Gal Kaminka, Pedro U. Lima and Raul Rojas (Eds), RoboCup 2002: Robot Soccer World Cup VI,LNAI, Springer, 2002
6. A. Birk, S. Carpin and H. Kenn, The IUB 2003 Rescue Robot Team RoboCup 2003: Robot Soccer World Cup VII, LNAI, Springer, 2003
7. H. Kenn, S. Carpin et al., FAST-Robots: a rapid-prototyping framework for intelligent mobile robotics, Artificial Intelligence and Applications (AIA 2003), ACTA Press, 2003
8. S. Carpin, H. Kenn and A. Birk, Autonomous Mapping in the Real Robots Rescue League, RoboCup 2003: Robot Soccer World Cup VII, LNAI, Springer, 2003
9. McLachlan, G., Krishnan, T.: The EM Algorithm and Extensions. Wiley-Interscience, 1996
10. Kalman, R.: A new approach to linear filtering and prediction problems. Transactions of ASME. Journal of Basic Engineering 83, 1960
11. Dissanayake, G., Newman, P., Clark, S., Durrant-Whyte, H., , Csorba., M.: A solution to the simultaneous localisation and map building (slam) problem. IEEE Transactions of Robotics and Automation 17, 229-241, 2001
12. Moravec, H.P.: Sensor fusion in certainty grids for mobile robots. AI Magazine, 1988
13. Thrun, S.: Robot mapping: a survey. Technical Report CMU-CS-02-111, Carnegie Mellon University, 2002
14. Kenn, H., Pfeil, A.: A sound source localization sensor using probabilistic occupancy grid maps. Proceedings of the Mechatronics and Robotics Conference 2004, IEEE Press 2004

Successful Search and Rescue in Simulated Disaster Areas

Alexander Kleiner, Michael Brenner, Tobias Bräuer, Christian Dornhege,
Moritz Göbelbecker, Mathias Luber, Johann Prediger,
Jörg Stückler, and Bernhard Nebel

Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
{kleiner, brenner, braeuer, dornhege, goebelbe, luber, prediger,
stueckle, nebel}@informatik.uni-freiburg.de

Abstract. RoboCupRescue Simulation is a large-scale multi-agent simulation of urban disasters where, in order to save lives and minimize damage, rescue teams must effectively cooperate despite sensing and communication limitations. This paper presents the comprehensive *search and rescue* approach of the *ResQ Freiburg* team, the winner in the RoboCupRescue Simulation league at RoboCup 2004.

Specific contributions include the predictions of travel costs and civilian lifetime, the efficient coordination of an *active* disaster space exploration, as well as an any-time rescue sequence optimization based on a genetic algorithm.

We compare the performances of our team and others in terms of their capability of extinguishing fires, freeing roads from debris, disaster space exploration, and civilian rescue. The evaluation is carried out with information extracted from simulation log files gathered during RoboCup 2004. Our results clearly explain the success of our team, and also confirm the scientific approaches proposed in this paper.

1 Introduction

The RoboCupRescue simulation league is part of the RoboCup competitions and aims at simulating large scale disasters and exploring new ways for the autonomous coordination of rescue teams [8]. These goals are socially highly significant and feature challenges unknown to other RoboCup leagues, like the coordination of heterogeneous teams with more than 30 agents, the exploration of a large scale environment in order to localize victims, as well as the scheduling of time critical rescue missions. Moreover, challenges similar to those found in other RoboCup leagues are inherent to the domain: The simulated environment is highly dynamic and only partially observable by a single agent. Agents have to plan and decide their actions asynchronously in real-time. Core problems in this league are *path planning*, *coordinated fire fighting* and coordinated *search and rescue* of victims.

This paper presents the comprehensive *search and rescue* approach of the *ResQ Freiburg* team, the winner in the RoboCupRescue Simulation league at RoboCup 2004. We show how learning and planning techniques can be used to provide predictive models that allow to act rationally despite the high dynamics of the simulation. Specific

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 323–334, 2006.

contributions include the prediction of the life-time of civilians based on *CART* [4] regression and *ADABoost* [6], travel cost prediction and its integration into target selection, the efficient coordination of an *active* disaster space exploration based on sensing, communication and reasoning, as well as an any-time rescue-sequence optimization based on a genetic algorithm. These techniques have in common that they provide ResQ agents with formal models for reasoning about the present and future state of the simulation despite its high dynamics. These models allow deliberative instead of purely reactive behavior, a capacity that we believe to be the reason for our team's success.

We have conducted an extensive comparison of the performance of our team with the performance of other teams in terms of the capability of extinguishing fires, freeing roads from debris, disaster space exploration, and civilian rescue. The evaluation is carried out with information extracted from simulation log files that were gathered during the RoboCup competition 2004. This evaluation gives much more information about a team's strengths and weaknesses than the standard scoring in the RoboCup Rescue Simulation league; yet it can be automated and therefore provides an instrument for precise analysis of teams. The results of our study clearly explain the success of our team, and also confirm the scientific approaches proposed in this paper.

The remainder of this paper is structured as follows. We present the active search and exploration approach in Section 2. The civilian rescue based on sequence optimization is described in Section 3. Path planning and travel cost prediction are covered in Section 4. Finally, an extensive evaluation and analysis of the 2004 RoboCupRescue competition is given in Section 5 and concluded in Section 6.

2 Exploration

In a partially observable environment like the RoboCupRescue simulation, exploration is the key means for agents to enlarge their knowledge. It is especially important to find injured civilians as quickly as possible without losing time by redundant exploration of the same area by several agents. Our agents achieve this ability by maintaining a *Knowledge Base* (KB) that keeps track of information collected on civilians during the search. Each agent maintains locally its own KB that is updated from senses, communication and reasoning. The KB allows them to efficiently focus and coordinate the search for civilians. It maintains the knowledge of an agent on the relation between the set of civilians C and the set of locations L. This is carried out by maintaining for each civilian $c \in C$ a set of locations L_c that contains all possible locations of the civilian. Furthermore, we maintain for each location $l \in L$ a set of civilians C_l that contains all civilians that are possibly situated at location l. Initially, $\forall c \in C, L_c = L$ and $\forall l \in L, C_l = C$.

The KB allows us the calculation of the expectation of the number of civilians situated at any location l. This is achieved by calculating the probability that civilian c is situated at location l, given the current state of the knowledge base:

$$P(loc(c) = l \mid KB_t) = \begin{cases} \frac{1}{|L_c|} & \text{if } l \in L_c \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Which yields the expectation on the number of civilians situated at location l:

$$E[|C_l|] = \sum_{i=0}^{|C|} P(loc(c_i) = l \mid KB_t) \tag{2}$$

Note that it follows from the above that initially the expectation for each location l is given by $E[|C_l|] = \frac{|C|}{|L|}$. That means that we expect civilians to be uniformly and independently distributed on the map. This is clearly not the case if buildings have a different size or different degree of destruction. As an improvement, one could incooperate this information as well. The KB is updated by either visual or auditory perception, communication of perception from other agents, and reasoning with respect to the agent's sensor model. The sensor model returns for any location l the set of locations V_l and A_l that are in visual (10m) or auditory (30m) range of l, respectively [11].

The KB is implemented as a $|C|x|L|$ boolean matrix, whereas C is the set of civilians and L the set of locations. Any entry $\langle c, l\rangle$ is set to *false* if a civilian c is definitely not at location l, and set to *true* otherwise (including the case of uncertainty). Initially, all entries are set to *true*. Based on the sensor model, one can perform either *positive* or *negative* update operations on the KB:

1. Positive updates:
   (a) **Civilian c seen at location l**
   We can reduce the set of possible locations for civilian c to l: $L_c := \{l\}$ and reduce [1] the set of possible civilians at location l to c: $C_l := \{c\}$;
   (b) **Civilian c heard at location l**
   We can remove civilian c from all civilian lists that are not in range of the sensor: $\forall l' : l' \notin A_l \Rightarrow C_l' := C_l' \setminus \{c\}$ and reduce the set of possible locations for civilian c to all locations that are in range of the sensor: $L_c := L_c \cap A_l$
2. Negative updates:
   (a) **Civilian c not seen at l**
   We can reduce the set of possible locations for civilian c by the set of locations within visual range: $L_c := L_c \setminus V_l$ and remove civilian c from all civilian lists for locations within visual range: $\forall l' : l' \in V_l \Rightarrow C_l' := C_l' \setminus \{c\}$
   (b) **Civilian c not heard at l**
   No safe conclusion possible

These update rules are very efficient due to the fact that the perception of the agents is free of noise. It is assumed that the agent is always able to see any civilian within the range of its sensors. Certainly this is not true in a real disaster situation. If, for example, victims are covered by rubble, they are even for humans hard to see. However, the proposed update rules can easily be enhanced towards probabilistic methods if there are probabilistic sensor models, which in turn have to be supported by the RoboCupRescue simulation system.

District exploration. District exploration is a multi-agent behavior for the coordinated search of buried civilians. The behavior guarantees that at any time each agent is assigned to a reachable and unexplored district on the map. In order to minimize the number of times an agent gets stuck in a blockade during exploration, districts have to consist of highly connected locations. The connectivity of two locations results from the number of alternative paths between them, the number of lanes and degree of blockage of each single road, and the degree of uncertainty on the state of the road. Due to the fact that blockades on the map and hence the map's connectivity is unknown to the agents

[1] Note if we see more than one civilian at the location, the set of possible civilians at l has to contain all of them.

in advance, the clustering has to be revised continuously. We used *agglomerative* [3] and *KD-tree* [2] based clustering in order to calculate a near optimal separation of the map into districts and to approximate the connectivity between them. These methods calculate from a given connectivity graph $G = \langle V, E \rangle$ of a city, where V represents the set of locations and E the set of connections between them, a hierarchical clustering. The hierarchical clustering, represented by a binary tree, provides at each level a partitioning of the city into n districts, reflecting the reachability of locations on the map (e.g. locations with a high connectivity are found within the same cluster). Based on this clustering, each team member is exclusively assigned to one reachable and unexplored cluster that represents a district on the map.

Active Exploration. Active exploration is an extension to the previously described district exploration task in that the search focuses on locations with high evidence on civilian whereabouts. This is carried out by exploiting the knowledge collected from senses, communication, and reasoning in the KB. Evidence from the KB is utilized by calculating an utility value $U(l)$ which is equal to the number of civilians expected to be found at observable locations O_l:

$$U\left(l\right) = \sum_{k \in O_l} E[|C_k|] \tag{3}$$

which yields, after inserting equation 2:

$$U\left(l\right) = \sum_{k \in O_l} \sum_{i=0}^{|C|} P(loc(c_i) = k \mid KB_t) \tag{4}$$

The overall sum of utilities over time can be maximized by the selection of targets with high utility as well as targets that are reachable within a short amount of time. Hence, from the set of locations L_D that are within the agent's district, a target location l_t is decided based on the trade-off between utility $U\left(l\right)$ and travel cost $T\left(l\right)$:

$$l_t = \underset{l \in L_D}{argmax}\, U\left(l\right) - \alpha * T\left(l\right) \tag{5}$$

whereas α is a constant regulating the trade-off between the estimated travel costs and the exploration utility and has to be determined experimentally. The estimated travel costs $T\left(l\right)$ are provided by a path planner that estimates costs based on a pre-calculated Dijkstra matrix (see Section 4).

Active surveillance. Furthermore, it is important for the rescue team to have up-to-date information on the injured civilians that have been found during the exploration task. The prediction module, described in Section 3, provides predictions of the civilian life time that are the more accurate the more up-to-date the information on *buriedness*, *damage* and *health* is. As we will describe in Section 3, the number of civilians that can be rescued depends on the efficiency of the rescue team, which in turn, depends on the accuracy of predictions. Hence, we extended the active exploration behavior in that it assigns agents to the surveillance of known civilian locations after the map has been explored sufficiently. The surveillance behavior is carried out by randomly sampling interesting locations from the set of known civilian locations whereby locations with obsolete information are selected with high probability. In general, information on locations are considered as obsolete if they haven't been visited by agents for a long time.

The number of agents that are assigned to active search is limited to $\frac{|L|}{k}$, whereas L is the set of open locations and k a constant that has to be determined experimentally. All agents above the assignment limit are performing active surveillance. If $k = 1$ then there will be at least as many explorers as open locations. If $k < 1$ then the exploration speed will be increased, but in turn there might be obsolete information on the health state of known victims. If $k > 1$ then the quality of information on known victims will increase, but the search for new agents might take more time.

Team coordination. Besides the team coordination due to the assignment of districts, it is necessary to further coordinate the multi-agent search in order to prevent the multiple exploration of locations. This is carried out by communicating the information on found civilians as well as locations that have been visited. However, if agents select exploration targets from the same district (i.e. due to the overlap or the shortage of available districts), it might occur that they explore locations twice. We implemented two methods to locally reduce the probability of multiple target exploration. Firstly, agents select exploration targets from a probability distribution. Secondly, agents negotiate targets they plan to explore in the next cycle via the short range communication channel (*say* and *hear*). It turned out that the latter performs poorly if agents are able to move much longer distances in a cycle than they are able to observe, which is true for the current parameter setting of the RoboCupRescue kernel. The problem could be solved by performing the negotiation via the long-range communication. Unfortunately, this does no pay off since long-range communication is a limited resource. Hence, agents choose their exploration targets from a probability distribution that assigns to each target a probability that is proportional to the score following equation 5. Note that the local target selection strategy could further be improved by utilizing game-theoretic methods.

3 Civilian Rescue

Lifetime prediction. To achieve good results in the civilian rescue process, it is necessary to know a civilian's chance of survival. If there is a reliable prediction for the life time of a certain civilian, the scheduling of the rescue operation can be adapted accordingly. On the one hand, it is possible that a civilian does not need to be rescued at all because she will be alive at the end of the simulation. On the other hand, it is possible that a civilian would die within a short amount of time and therefore has to be rescued as soon as possible in order to survive.

For the ResQ Freiburg agents, machine learning techniques were used to gain a prediction of the civilian's life time and classification into survivors and victims. We created an *autorun tool* that starts the simulation and the agents simultaneously in order to collect data. The tool was used for several simulation runs on the Kobe, VC, and Foligno maps, from which a large amount of datasets were generated.

A data set consists of the values for *health* and *damage* of each civilian at each time step gained during the simulation. In order to reduce the noise in the data, simulations were carried out within 400 time steps, without rescue operations by the agents and without fires on the map. The latter two conditions are necessary in order to prevent unexpected changes of the damage to a civilian due to its rescue, resulting in zero damage, or due to fires, resulting in unpredictable high damage. For the calculation of the life time, there has to be determined a time of death for each dataset. Hence, the simulation time was chosen to be 400 rounds, which seemed to be a good compromise between an ideal simulation time of ∞ and the standard simulation time of 300 rounds that would lead to a non-uniform distribution of the datasets.

Regression and classification was carried out with the *WEKA* [12] machine learning tool. We utilized the *C4.5* algorithm (decision trees) for the classification task. The regression of the simulation time is based on Adaptive Boosting (Ada Boost) [6]. Since the current implementation of the *WEKA* tool does only provide Ada Boost on classification, we had to extend this implementation for regression [5], which then has been applied with regression trees (CART) [4].

The regression trees have been evaluated on test data sets in order to learn the confidence of a prediction in dependency of the civilian's damage and the distance between the query time and the predicted time of death. Confidence values are necessary since the higher the difference between the observation and the civilian's actual time of death, the less accurate predictions are. The sequence optimization, described in Section 3, relies on the confidence values in order to minimize sequence fluctuations.

Genetic Sequence Optimization. If the time needed for rescuing civilians and the life time of civilians is predictable, one can estimate the overall number of survivors after executing a rescue sequence by a simulation. For each rescue sequence $S = \langle t_1, t_2, ..., t_n \rangle$ of n rescue targets, an utility $U(S)$ is calculated that is equal to the number of civilians that are expected to survive. Unfortunately, an exhaustive search over all $n!$ possible rescue sequences is intractable. A straightforward solution to the problem is, for example, to sort the list of targets by the time necessary to reach and rescue them and to subsequently rescue targets from the top of the list. However, as shown in Section 5, this might lead to unsatisfying solutions. Hence, we decided to utilize a Genetic Algorithm (GA) for the optimization of sequences and thus the subsequent improvement of existing solutions [7].

The time for rescuing civilians is approximated by a linear regression based on the buriedness of a civilian and the number of ambulance teams that are dispatched to the rescue. Travel costs between two targets are estimated by averaging over costs sampled during previous simulation runs. This is much more efficient than the calculation of exact travel cost involving, in the worst case, the calculation of the Floyd-Warshall matrix in $O(n^3)$.

The GA is initialized with heuristic solutions, for example, solutions that *greedily* prefer targets that can be rescued within a short time or urgent targets that have a short lifetime. The fitness function of solutions is set equal to the previously described utility $U(S)$. In order to guarantee that solutions in the genetic pool are at least as good as the heuristic solutions, the so called *elitism* mechanism, which forces the permanent existence of the best found solution in the pool, has been used. Furthermore, we utilized a simple one-point-crossover strategy, a uniform mutation probability of $p \approx 1/n$, and a population size of 10. Within each cycle, 500 populations of solutions are calculated by the ambulance station from which the best sequence is broadcasted to the ambulance teams that synchronously start to rescue the first civilian in the sequence.

One difficulty of the sequence optimization is given by the fact that information stored in the KB on civilians changes dynamically during each round and thus might cause fluctuations of the rescue sequence. This can be caused by two reasons: Firstly, civilians are discovered by active exploration, which is executed by other agents at the same time. Secondly, predictions vary due to information updates from active or passive surveillance. The latter effect can be weakened by updating the sequence with respect to the confidence of predictions. Updates of the information on civilians are ignored, if they are not statistically significant with respect to their confidence interval.

The effect of information updates due to exploration has to be controlled by deciding between rescue permanence and rescue latency, i.e. how frequently change the ambulances their targets and how fast can they react on emergency targets. Therefore we implemented additionally a reactive mechanism that recognizes emergency rescue targets that have to be rescued immediately. A target is defined as an emergency target if it would die if not being rescued within the next round. However, any other target is only taken as emergency target, if the current target would safely survive if postponing its rescue.

4 Path Planning

Every rescue agent must do path planning in order to reach its selected target position. ResQ Freiburg agents, however, use path planning already *during* target selection and thus can account for the time needed to reach a target when calculating its utility. Such an approach is only possible with a very efficient path planner that can be queried several hundred times in each cycle.

The efficiency of the ResQ path planner stems from the following realization (explained in more detail in [9]): many nodes on the road graph of a RoboCup Rescue map connect exactly two roads plus one or more houses. If none of these houses is the source or destination of the path planner query, the node can only be crossed, leaving no choices for the planner. Only nodes with more than two adjacent roads constitute real *crossings*. The road segments and simple nodes between the crossings can be joined in one *longroad*, which has no inner crossings. Longroads and crossings form a new, much smaller graph on which shortest path algorithms can be run much more quickly than on the larger original graph.

Since every node n from the original graph lies on a longroad, each path to or from n must include one of the two endpoint crossings of that longroad, c_n^1 and c_n^2. An optimal path from nodes s and e from the original graph therefore has length

$$\min_{i,j}\left(\overline{sc_s^i} + \overline{P(c_s^i, c_e^j)} + \overline{c_e^j e}\right) \text{ where } i,j \in \{1,2\} \tag{6}$$

To solve this formula efficiently, the ResQ planner stores the direct routes from a location to its adjacent crossings. The optimal paths $P(c_s^i, c_e^j)$ between crossings are computed with Dijkstra's algorithm.

Adequacy of the path planner for the Rescue Domain is even more important than its efficiency. Most often agents want to know how long it will take to reach destinations. Therefore the cost functions used by the ResQ path planner have been designed not to return *path lengths* (although this is of course possible) but to predict the *time* it will take an agent to reach its destination. To compute this, the planner tries to consider not only the length of a path, but also partial blockades, acceleration/deceleration at crossings, right of way (depending on from where a crossing is entered), and other agents' trajectories. While in the RCR system, these factors are accurately simulated, it is necessary for the ResQ Path Planner to use predictive functions in order to obtain the speed for several hundred queries per second.

We have provided several such prediction functions which, depending on the situation, use different aspects of an agent's knowledge about the world. For example, agents may sometimes want to choose only among roads that are known to be unblocked, but in other cases may ignore blockades completely in order to find out the minimal time to reach a target. Since the complex metrics used account for many of the specific influences mentioned above, we have been able to give a quite accurate prediction of travel durations in many cases. This prediction is then utilized by other components, e.g. the sequence optimizer for civilian rescue (cf. Section 3).

The simulation is cycle-based. Hence, finding paths with minimal lengths or even minimal duration is often not the wisest choice, since two paths differing only by a few meters or, respectively, a few seconds can often be considered as equivalent as long as they will take the same number of cycles to travel. This allows agents to build equivalence classes among paths and, consequently, targets. Several selection mechanisms allow to optimize other criteria when, for a set of targets, the expected number of cycles to reach them is equal. It is thus possible for an agent to select the most important target among the ones most easily reachable or, vice-versa, the closest among the most important targets.

5 Results

During the competition, teams are evaluated by an overall score that is calculated based on the state of civilian health and building destruction. However, since this score incooperates the total performance of all agent skills, such as exploration, extinguishing, and rescuing, it is impossible to assess single agent skills directly. In order to compare our agents with agents from other teams, the performance of typical agent skills are

Table 1. Percentage of clean roads

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	74,68	**82,22**	71,79	70,43	N/A	N/A	N/A	N/A
Final-Random	77,84	**86,51**	77,66	63,10	N/A	N/A	N/A	N/A
Final-Kobe	92,25	**93,74**	92,08	92,05	N/A	N/A	N/A	N/A
Final-Foligno	96,41	**97,72**	97,22	96,07	N/A	N/A	N/A	N/A
Semi-VC	67,93	**79,57**	68,86	57,90	67,22	57,85	53,27	80,53
Semi-Random	82,53	**87,44**	77,47	81,93	82,26	79,53	80,30	78,76
Semi-Kobe	92,40	93,65	92,71	92,51	92,62	92,56	93,55	**99,72**
Semi-Foligno	95,45	**97,08**	95,58	96,37	96,93	97,07	95,92	83,44
Round2-Kobe	92,52	93,52	91,46	92,46	92,78	93,45	92,25	**99,50**
Round2-Random	87,74	90,03	87,62	87,71	87,86	88,73	85,03	**99,97**
Round2-VC	91,34	91,62	90,74	89,87	91,40	90,92	N/A	**98,86**
Round1-Kobe	89,19	89,51	87,78	88,21	88,30	87,70	**91,12**	81,17
Round1-VC	91,90	92,13	91,74	91,84	N/A	91,81	91,54	**99,82**
Round1-Foligno	95,84	96,92	96,52	96,36	94,19	96,62	**97,63**	80,15
Number of wins	0	**7**	0	0	0	0	2	5
AVG %:	87,72	**90,83**	87,09	85,49	88,17	87,62	86,73	90,19
STD %:	8,25	**5,09**	8,59	11,25	8,93	11,59	13,63	9,96

Table 2. Percentage of saved buildings

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	47,21	54,13	**81,67**	43,19	N/A	N/A	N/A	N/A
Final-Random	24,04	**26,38**	15,03	12,35	N/A	N/A	N/A	N/A
Final-Kobe	38,24	**61,89**	38,38	13,51	N/A	N/A	N/A	N/A
Final-Foligno	**91,15**	62,77	60,92	34,56	N/A	N/A	N/A	N/A
Semi-VC	23,45	23,60	25,49	27,14	19,12	25,10	26,36	**27,22**
Semi-Random	23,18	**28,73**	18,09	19,55	22,82	21,45	17,09	18,91
Semi-Kobe	**96,49**	76,76	94,32	95,41	24,32	90,54	55,27	94,19
Semi-Foligno	36,22	**38,06**	32,72	37,79	31,89	28,48	26,82	23,23
Round2-Kobe	70,27	37,03	59,73	95,41	48,38	61,49	10,54	**95,54**
Round2-Random	99,04	60,91	54,68	99,16	63,55	97,60	80,70	**99,52**
Round2-VC	10,23	11,57	10,23	13,53	12,67	**71,99**	N/A	36,51
Round1-Kobe	99,46	98,92	**99,73**	**99,73**	99,05	98,78	67,16	91,89
Round1-VC	97,25	99,53	79,70	**99,76**	N/A	98,90	99,53	99,53
Round1-Foligno	**98,99**	**98,99**	36,13	45,99	32,53	54,29	43,59	29,86
Number of Wins:	3	**5**	2	2	0	1	0	3
AVG %:	61,09	55,66	50,49	52,65	39,37	**64,86**	47,45	61,64
STD %:	37,80	34,11	31,83	37,50	27,28	**31,63**	30,49	36,70

Table 3. Percentage of explored buildings

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	83,48	83,24	**87,02**	67,27	N/A	N/A	N/A	N/A
Final-Random	69,62	72,62	**78,13**	49,92	N/A	N/A	N/A	N/A
Final-Kobe	89,19	92,97	89,73	**94,19**	N/A	N/A	N/A	N/A
Final-Foligno	84,15	85,25	**86,73**	74,29	N/A	N/A	N/A	N/A
Semi-VC	69,39	72,86	**77,42**	45,08	52,01	52,87	47,92	59,72
Semi-Random	**78,91**	68,73	71,91	54,36	59,36	70,27	46,18	46,18
Semi-Kobe	85,41	96,22	92,97	95,54	66,62	97,30	**99,46**	91,89
Semi-Foligno	74,75	89,12	84,98	62,49	65,35	**92,53**	79,08	20,74
Round2-Kobe	87,16	90,68	95,00	91,76	80,54	94,19	**99,46**	92,43
Round2-Random	81,18	80,94	88,61	84,53	60,67	**94,24**	82,61	87,89
Round2-VC	83,40	70,18	84,58	40,44	67,74	87,88	N/A	**89,54**
Round1-Kobe	87,43	90,27	94,05	96,08	96,62	97,70	**97,84**	80,95
Round1-VC	85,37	90,48	95,28	94,26	N/A	97,72	**100,00**	91,35
Round1-Foligno	83,78	**90,05**	**90,05**	60,00	54,65	88,57	67,37	13,00
Number of Wins:	1	1	**4**	1	0	2	4	1
AVG %:	81,66	83,83	**86,89**	72,16	67,06	87,33	79,99	67,37
STD %:	5,82	9,98	**7,87**	22,21	13,87	14,59	21,84	30,80

emphasized by an evaluation of log files that were collected during the 2004 competition. The following tables provide results from all rounds of all teams that passed the preliminaries. All values refer to the last round, i.e. the percentage of clean roads at round 300. Bold numbers denote the best results that have been achieved during the respective round.

Table 1 shows the percentage of blockades that have been removed by the police agents. The results show that particularly the teams *Damas Rescue* and *The Black Sheep* most efficiently removed blockage from the roads. Table 2 shows the percentage of buildings that have been saved by the fire brigades. Obviously the team *Damas Rescue* saved most of the buildings, whereas *SBC* reached a robust behavior, shown by the good average value. The efficiency of exploration turned out to be one of the most important criteria for the team evaluation. The more locations of civilians are known, the more efficiently rescue operations can be scheduled. Table 3 shows the percentage of buildings

Table 4. Percentage of found civilians

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	97,22	94,44	**100,00**	81,94	N/A	N/A	N/A	N/A
Final-Random	**90,91**	85,71	81,82	70,13	N/A	N/A	N/A	N/A
Final-Kobe	**98,77**	97,53	95,06	**98,77**	N/A	N/A	N/A	N/A
Final-Foligno	**96,67**	**96,67**	**96,67**	72,22	N/A	N/A	N/A	N/A
Semi-VC	77,92	77,92	**85,71**	45,45	53,25	53,25	50,65	63,64
Semi-Random	**88,51**	73,56	72,41	63,22	67,82	80,46	52,87	55,17
Semi-Kobe	**100,00**	**100,00**	**100,00**	98,61	79,17	**100,00**	**100,00**	97,22
Semi-Foligno	90,12	95,06	86,42	81,48	83,95	**97,53**	85,19	30,86
Round2-Kobe	98,89	98,89	97,78	95,56	91,11	**100,00**	**100,00**	98,89
Round2-Random	**98,89**	95,56	**98,89**	81,11	70,00	96,67	85,56	94,44
Round2-VC	**92,22**	78,89	90,00	45,56	72,22	88,89	N/A	87,78
Round1-Kobe	94,29	**100,00**	**100,00**	98,57	**100,00**	**100,00**	94,29	78,57
Round1-VC	**100,00**	**100,00**	**100,00**	97,14	N/A	**100,00**	**100,00**	98,57
Round1-Foligno	**100,00**	97,14	94,29	77,14	74,29	92,86	77,14	14,29
Number of Wins:	**9**	4	7	1	1	5	3	0
AVG %:	**94,60**	92,24	92,79	79,06	76,87	90,97	82,85	71,94
STD %:	**7,17**	10,53	9,03	20,75	13,73	14,69	19,35	30,25

Table 5. Number of saved civilians

	ResQ	Damas	Caspian	BAM	SOS	SBC	ARK	B.Sheep
Final-VC	42	43	**52**	34	N/A	N/A	N/A	N/A
Final-Random	**32**	25	29	16	N/A	N/A	N/A	N/A
Final-Kobe	**46**	45	**46**	30	N/A	N/A	N/A	N/A
Final-Foligno	**66**	54	50	29	N/A	N/A	N/A	N/A
Semi-VC	**18**	15	17	12	11	12	12	14
Semi-Random	22	**26**	16	14	20	14	15	15
Semi-Kobe	**57**	47	54	52	20	39	34	44
Semi-Foligno	37	**46**	44	43	42	28	29	24
Round2-Kobe	**57**	37	43	50	43	35	28	43
Round2-Random	**52**	48	39	45	47	44	50	37
Round2-VC	31	33	32	24	37	**51**	N/A	34
Round1-Kobe	45	**51**	47	43	47	31	25	34
Round1-VC	**62**	**62**	55	57	N/A	51	54	44
Round1-Foligno	**53**	**53**	37	33	37	41	30	23
#Wins:	**9**	5	2	0	0	1	0	0
Σ TOTAL:	**620**	585	561	482	304	346	277	312
Σ SEMI+PREM	**434**	418	384	373	304	346	277	312

that were visited by agents[2]. The result shows that *Caspian* explored most of the buildings. However, the percentage of explored buildings does not necessarily correlate with the percentage of found civilians, as shown by table 4[3]. This is due to the fact that communication as well as reasoning might increase the efficiency of exploration. At the end, more civilians were found by *ResQ Freiburg* than *Caspian*, although the latter team explored more buildings. Important for efficient rescue operations is the point in time when civilian whereabouts are known. The earlier civilians are found, the better their rescue can be scheduled. Fig. 1 shows the number of civilians found during each cycle on the *RandomMap*. The results confirm the efficiency of *ResQ Freiburg's* exploration: At any time, the agents knew about more civilians than agents of any other team.

[2] Note: Full communication of visited locations and exploitation of a sensor model was assumed.

[3] Note: Civilians are considered as being found if one of the agents was within their visual range.

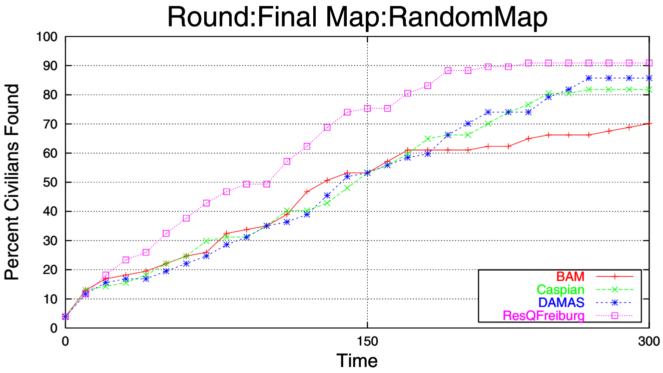


Fig. 1. Number of civilians found by exploration on a randomly generated map during the final

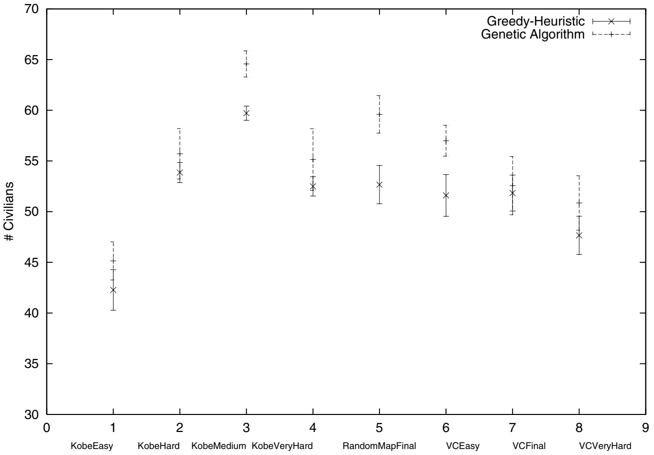


Fig. 2. Number of rescued civilians under two different strategies

Fig. 2 documents the difference between a greedy rescue target selection, i.e. preferring targets that can be rescued fast and selection based on an optimization by a genetic algorithm. It can be seen that an optimization of the rescue sequence clearly increases the number of rescued civilians. Finally table 5 shows the number of civilians saved by each team: *ResQ Freiburg* saved more than 620 civilians during all rounds, which are 35 more than the second best and 59 more than the third best in the competition.

6 Conclusion

The results presented explain the success of the *ResQ Freiburg* team during RoboCup 2004: While ResQ Freiburg's police agents (removal of blockades) and fire agents (extinguishing fires) performed comparably as good as agents from other teams (see table 2 and 1), the exploration and rescue sequence optimization abilities clearly outperformed the strategies of other teams (see table 4 and 5). Even during the competition's final on the *RandomMap*, which decided by only 0.4 points of the total score the positioning between *Damas Rescue* and *ResQ Freiburg*, *ResQ Freiburg* was able to rescue seven civilians more than the second best.

In total, our results provide an interesting insight into the RoboCupRescue simulation competition: In addition to strategies for extinguishing fires and the removal of blockades, as they were favored by teams during the last years, exploration and sequence optimization are crucial subproblems in the RoboCupRescue simulation league. The proposed analysis provides a methodology for the further study of different strategies in this complex domain. The scoring metric for team evaluation shown in this paper has been integrated into the new 3D viewer of the RoboCupRescue simulation league, which we contributed for the next RoboCup competitions [10].

Currently, our team started to develop robots for the RoboCupRescue Real Robot league. We are confident that the methods proposed in this paper are also helpful in this context. Likewise as agents in the simulation, these robots have to find victims autonomously in an unknown terrain. Sensors, such as thermo cameras or CO_2 detectors, are used to make the search more efficient, in fact they are used to search for victims actively.

In addition to the proposed methods, various tools for agent world modelling and communication were developed by our team. These tools and also all algorithms discussed in this paper, are freely available for download from the official home page of RoboCupRescue simulation 2005 [1].

References

1. Homepage RoboCupRescue 2005. `http://kaspar.informatik.uni-freiburg.de/~rcr2005`, 2005.
2. J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
3. H.H. Bock. *Autmatic Classification*. Vandenhoeck and Ruprecht, 1974.
4. L. Breiman, J.H. Friedman, R. A. Olshen, and C.J. Stone. *Classification and regression trees*. Wadsworth & Brooks, 1984.
5. Harris Drucker. Improving regressors using boosting techniques. In *Proc. 14th International Conference on Machine Learning*, pages 107–115. Morgan Kaufmann, 1997.
6. Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
7. J. H. Holland. *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.
8. H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. RoboCup Rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In *IEEE Conf. on Man, Systems, and Cybernetics(SMC-99)*, 1999.
9. A. Kleiner, M. Brenner, T. Braeuer, C. Dornhege, M. Goebelbecker, M. Luber, J. Prediger, J. Stueckler, and B. Nebel. Resq freiburg: Team description and evaluation. Technical report, Institut für Informatik, Universität Freiburg, Germany, 2005.
10. A. Kleiner and M. Goebelbecker. Rescue3d: Making rescue simulation attractive to the public. `http://kaspar.informatik.uni-freiburg.de/~rescue3D/`, 2004.
11. T. Morimoto. *How to Develop a RoboCupRescue Agent*, 2002. http://ne.cs.uec.ac.jp/~morimoto/rescue/manual/.
12. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.

3D Mapping with Semantic Knowledge

Andreas Nüchter[1], Oliver Wulf[2], Kai Lingemann[1], Joachim Hertzberg[1],
Bernardo Wagner[2], and Hartmut Surmann[3]

[1] University of Osnabrück, Institute for Computer Science,
Knowledge-Based Systems Research Group, Albrechtstraße 28,
D-49069 Osnabrück, Germany
nuechter@informatik.uni-osnabrueck.de
http://www.informatik.uni-osnabrueck.de/nuechter/

[2] University of Hannover, Institute for Systems Engineering (ISE/RTS),
Appelstraße 9A, D-30167 Hannover, Germany

[3] Fraunhofer Institute for Autonomous Intelligent Systems (AIS),
Schloss Birlinghoven, D-53754 Sankt Augustin, Germany

Abstract. A basic task of rescue robot systems is mapping of the environment. Localizing injured persons, guiding rescue workers and excavation equipment requires a precise 3D map of the environment. This paper presents a new 3D laser range finder and novel scan matching method for the robot Kurt3D [9]. Compared to previous machinery [12], the apex angle is enlarged to 360°. The matching is based on semantic information. Surface attributes are extracted and incorporated in a forest of search trees in order to associate the data, i.e., to establish correspondences. The new approach results in advances in speed and reliability.

1 Introduction

Rescue robotic systems are designed to assist rescue workers in earthquake, fire, flooded, explosive and chemical disaster areas. Currently, many robots are manufactured, but most of them lack a reliable mapping method. Nevertheless, a fundamental task of rescue is to localize injured persons and to map the environment. To solve these tasks satisfactorily, the generated map of the disaster environment has to be three-dimensional. Solving the problem of simultaneous localization and mapping (SLAM) for 3D maps turns the localization into a problem with six degrees of freedom. The x, y and z positions and the roll, yaw and pitch orientations of the robot have to be considered. We are calling the resulting SLAM variant 6D SLAM [10].

This paper addresses the problem of creating a consistent 3D scene in a common coordinate system from multiple views. The proposed algorithms allow to digitize large environments fast and reliably without any intervention and solve the 6D SLAM problem. A 360° 3D laser scanner acquires data of the environment and interprets the 3D points online. A fast variant of the iterative closest points (ICP) algorithm [3] registers the 3D scans in a common coordinate system and relocalizes the robot. The registration uses a forest of approximate kd-trees. The resulting approach is highly reliable and fast, such that it can be applied online to exploration and mapping in RoboCup Rescue.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 335–346, 2006.

The paper is organized as follows: The remainder of this section describes the state of the art in automatic 3D mapping and presents the autonomous mobile robot and the used 3D scanner. Section 2 describes briefly the online extraction of semantic knowledge of the environment, followed by a discussion of the scan matching using forests of trees (section 3). Section 4 presents experiments and results and concludes.

1.1 3D Mapping – State of the Art

A few groups use 3D laser scanners [1,5,11,14,15]. The RESOLV project aimed to model interiors for virtual reality and tele presence [11]. They used a RIEGL laser range finder on robots and the ICP algorithm for scan matching [3]. The AVENUE project develops a robot for modeling urban environments [1], using an expensive CYRAX laser scanner and a feature-based scan matching approach for registration of the 3D scans in a common coordinate system. Nevertheless, in their recent work they do not use data of the laser scanner in the robot control architecture for localization [5]. Triebel et al uses a SICK scanner on a 4 DOF robotic arm mounted on a B21r platform to explore the environment [14].

Instead of using 3D scanners, which yield consistent 3D scans in the first place, some groups have attempted to build 3D volumetric representations of environments with 2D laser range finders [7,8,13,15]. Thrun et al. [7,13] use two 2D laser range finder for acquiring 3D data. One laser scanner is mounted horizontally, the other vertically. The latter one grabs a vertical scan line which is transformed into 3D points based on the current robot pose. The horizontal scanner is used to compute the robot pose. The precision of 3D data points depends on that pose and on the precision of the scanner. Howard et al. uses the restriction of flat ground and structured environments [8]. Wulf et al. let the scanner rotate around the vertical axis. They acquire 3D data while moving, thus the quality of the resulting map crucial depends on the pose estimate that is given by inertial sensors, i.e., gyros [15]. In this paper we let rotate the scanner continuously around its vertical axis, but accomplish the 3D mapping in a stop-scan-go fashion, therefore acquiring consistent 3D scans as well.

Other approaches use information of CCD-cameras that provide a view of the robot's environment. Some groups try to solve 3D modeling by using a planar SLAM methods and cameras, e.g., in [4].

1.2 Automatic 3D Sensing

The Robot Platform Kurt3D. Kurt3D (Fig. 1) is a mobile robot platform with a size of 45 cm (length) × 33 cm (width) × 26 cm (height) and a weight of 15.6 kg, both indoor as well as outdoor models exist. Two 90 W motors (short-term 200 W) are used to power the 6 wheels. Compared to the original Kurt3D robot platform, the outdoor version has larger wheels, where the middle ones are shifted outwards. Front and rear wheels have no tread pattern to enhance rotating. Kurt3D operates for about 4 hours with one battery charge (28 NiMH cells, capacity: 4500 mAh) charge. The core of the robot is a laptop computer

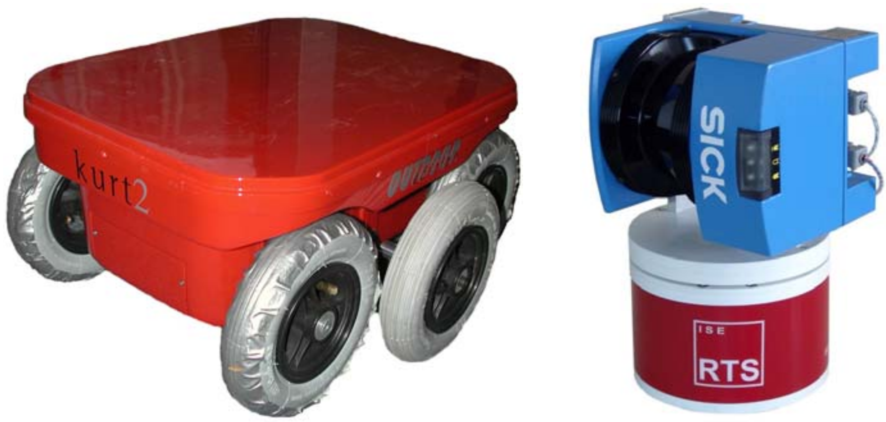


Fig. 1. The mobile robot platform Kurt3D offroad (left) and the 3D laser scanner (right) The scanner rotates around the vertical axis. It's technical basis is a SICK 2D laser range finder (LMS-200).

running a Linux operating system. An embedded 16-Bit CMOS microcontroller is used to process commands to the motor. A CAN interface connects the laptop with the microcontroller.

The 3D Laser Scanner. As there is no commercial 3D laser range finder available that could be used for mobile robots, it is common practice to assemble 3D sensors out of a standard 2D scanner and an additional servo drive [6, 12]. The scanner that is used for this experiment is based on a SICK LMS 291 in combination with the RTS/ScanDrive developed at the University of Hannover. Different orientations of the 2D scanner in combination with different turning axes result in a number of possible scanning patterns. The scanning pattern that is most suitable for this rescue application is the yawing scan with a vertical 2D raw scan and rotation around the upright axis (see Fig. 1). The yawing scan pattern results in the maximal possible field of view (360° horizontal and 180° vertical) and an uniform distribution of scan points.

As 3D laser scanner for autonomous search and rescue applications needs fast and accurate data acquisition in combination with low power consumption, the RTS/ScanDrive incorporates a number of improvements. One mechanical improvement is the ability to turn continuously, which is implemented by using slip rings for power and data connection to the 2D scanner. This leads to a homogeneous distribution of scan points and saves the energy and time that is needed for acceleration and deceleration of panning scanners. Another improvement that becomes more important with short scanning times of a few seconds is the compensation of systematic measurement errors. In this case the compensation is done by sensor analysis and hard real-time synchronization, using a Linux/RTAI operation system. These optimizations lead to scan times as short as 3.2s for a yawing scan with 1.5° horizontal and 1° vertical resolution (240x181 points). For details on the RTS/ScanDrive see [17].

2 Extracting Semantic Information

The basic idea of labelling 3D points with semantic information is to use the gradient between neighbouring points to differ between three categories, i.e., floor-, object- and ceiling-points. A 3D point cloud that is scanned in a yawing scan configuration, can be described as a set of points $\mathbf{p}_{i,j} = (\phi_i, r_{i,j}, z_{i,j})^T$ given in a cylindrical coordinate system, with i the index of a vertical raw scan and j the point index within one vertical raw scan counting bottom up. The gradient $\alpha_{i,j}$ is calculated by the following equation:

$$\tan \alpha_{i,j} = \frac{z_{i,j} - z_{i,j-1}}{r_{i,j} - r_{i,j-1}} \quad \text{with} \quad -\frac{1}{2}\pi \leq \alpha_{i,j} < \frac{3}{2}\pi.$$

The classification of point $\mathbf{p}_{i,j}$ is directly derived from the gradient $\alpha_{i,j}$:

1. floor-points: $\alpha_{i,j} < \tau$
2. object-points: $\tau \leq \alpha_{i,j} \leq \pi - \tau$
3. ceiling-points: $\pi - \tau < \alpha_{i,j}$

with a constant τ that depends on the maximal ascent being accessible by the robot (here: $\tau = 20°$).

Applied to real data, this simple definition causes two problems. As can be seen in Fig. 2(a) noisy range data can lead to wrong classifications of floor- and ceiling-points. Changing the differential quotient as follows solves this problem:

$$\tan \alpha_{i,j} = \frac{z_{i,j} - z_{i,j-k}}{r_{i,j} - r_{i,j-k}}$$

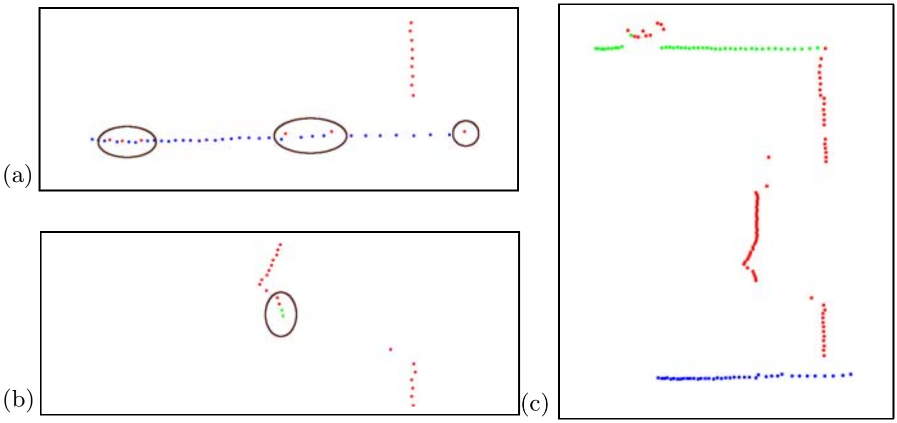


Fig. 2. Extracting semantic information using a slice of a 3D scan. (a) Problems with simple gradiant definition, marked with circles. (b) Problems with jump edges. (c) Correct semantic classification.

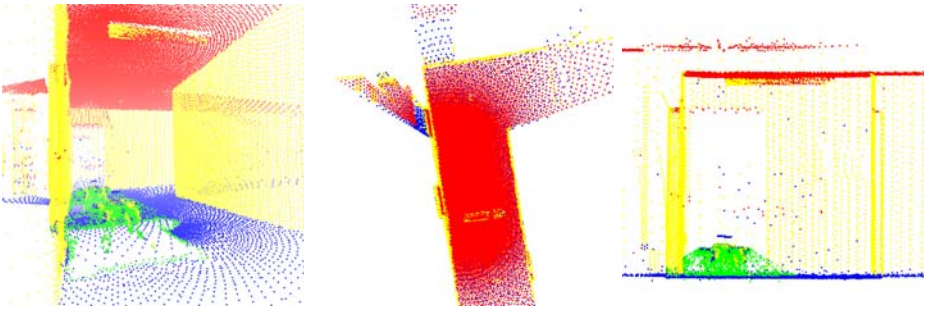


Fig. 3. Semantically labeled 3D point cloud from a single 360° 3D scan. Red points mark the ceiling, yellow points objects, blue points the floor and green points correspond to artefacts from scanning the RTS/ScanDrive and the robot.

with $k \in \mathbb{N}, k \geq 1$ the smallest number so that

$$\sqrt{(r_{i,j} - r_{i,j-k})^2 + (z_{i,j} - z_{i,j-k})^2} > d_{\min}$$

for a constant $d_{\min}$ depending on the scanner's depth accuracy σ (here: $\sigma =$ 30 mm, $d_{\min} = 2\sigma$).

The second difficulty is the correct computation of the gradient across jumping edges (see Fig. 2(b)). This problem is solved with a prior segmentation [16], as the gradient $\alpha_{i,j}$ is only calculated correctly if both points $\mathbf{p}_{i,j}$ and $\mathbf{p}_{i,j-k}$ belong to the same segment. The correct classification result can be seen in Fig. 2(c). Fig. 3 shows a 3D scan with the semantic labels.

3 Scan Registration and Robot Relocalization

Multiple 3D scans are necessary to digitalize environments without occlusions. To create a correct and consistent model, the scans have to be merged into one coordinate system. This process is called registration. If the localization of the robot with the 3D scanner were precise, the registration could be done directly based on the robot pose. However, due to the unprecise robot sensors, self localization is erroneous, so the geometric structure of overlapping 3D scans has to be considered for registration. Furthermore, Robot motion on natural surfaces has to cope with yaw, pitch and roll angles, turning pose estimation into a problem in six mathematical dimensions. A fast variant of the ICP algorithm registers the 3D scans in a common coordinate system and relocalizes the robot. The basic algorithm was invented in 1992 and can be found, e.g., in [3].

Given two independently acquired sets of 3D points, M (model set, $|M| = N_m$) and D (data set, $|D| = N_d$) which correspond to a single shape, we aim

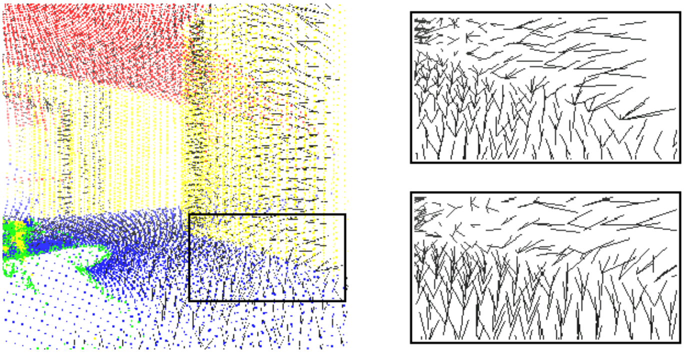


Fig. 4. Point pairs for the ICP scan matching algorithm. The left image show parts of two 3D scans and the closest point pairs as black lines. The right images show the point pairs in case of semantically based matching (top) whereas the bottom part shows the distribution with closest points without taking the semantic point type into account.

to find the transformation consisting of a rotation $\mathbf{R}$ and a translation $\mathbf{t}$ which minimizes the following cost function:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_j + \mathbf{t})\right|\right|^2 . \tag{1}$$

$w_{i,j}$ is assigned 1 if the i-th point of M describes the same point in space as the j-th point of D. Otherwise $w_{i,j}$ is 0. Two things have to be calculated: First, the corresponding points, and second, the transformation ($\mathbf{R}$, $\mathbf{t}$) that minimize $E(\mathbf{R}, \mathbf{t})$ on the base of the corresponding points. The ICP algorithm calculates iteratively the point correspondences. In each iteration step, the algorithm selects the closest points as correspondences and calculates the transformation $(\mathbf{R}, \mathbf{t})$ for minimizing equation (1). The assumption is that in the last iteration step the point correspondences are correct. Besl et al. prove that the method terminates in a minimum [3]. However, this theorem does not hold in our case, since we use a maximum tolerable distance $d_{\max}$ for associating the scan data. Here $d_{\max}$ is set to 15 cm for the first 15 iterations and then this threshold is lowered to 5 cm. Fig. 4 (left) shows two 3D scans aligned only according to the error-prone odometry-based pose estimation. The point pairs are marked by a line.

3.1 Computing the Optimal Rotation and Translation in 6D

In every iteration the optimal transformation ($\mathbf{R}$, $\mathbf{t}$) has to be computed. Eq. (1) can be reduced to

$$E(\mathbf{R}, \mathbf{t}) \propto \frac{1}{N} \sum_{i=1}^{N} \left|\left|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_i + \mathbf{t})\right|\right|^2 , \tag{2}$$

with $N = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j}$, since the correspondence matrix can be represented by a vector containing the point pairs.

In earlier work [10] we used a quaternion based method [3], but the following one, based on singular value decomposition (SVD), is robust and easy to implement, thus we give a brief overview of the SVD based algorithms. It was first published by Arun, Huang and Blostein [2]. The difficulty of this minimization problem is to enforce the orthonormality of matrix **R**. The first step of the computation is to decouple the calculation of the rotation **R** from the translation **t** using the centroids of the points belonging to the matching, i.e.,

$$\mathbf{c}_m = \frac{1}{N}\sum_{i=1}^{N}\mathbf{m}_i, \qquad \mathbf{c}_d = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d}_j \tag{3}$$

and

$$M' = \{\mathbf{m}'_i = \mathbf{m}_i - \mathbf{c}_m\}_{1,\ldots,N}, \tag{4}$$

$$D' = \{\mathbf{d}'_i \ = \mathbf{d}_i \ - \mathbf{c}_d\}_{1,\ldots,N}. \tag{5}$$

After replacing (3), (4) and (5) in the error function, $E(\mathbf{R},\mathbf{t})$ eq. (2) becomes:

$$E(\mathbf{R},\mathbf{t}) \propto \frac{1}{N}\sum_{i=1}^{N}||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i - \underbrace{(\mathbf{t} - \mathbf{c}_m + \mathbf{R}\mathbf{c}_d)}_{=\tilde{\mathbf{t}}}||^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2 \tag{6a}$$

$$-\frac{2}{N}\tilde{\mathbf{t}}\cdot\sum_{i=1}^{N}(\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i) \tag{6b}$$

$$+\frac{1}{N}\sum_{i=1}^{N}||\tilde{\mathbf{t}}||^2. \tag{6c}$$

In order to minimize the sum above, all terms have to be minimized. The second sum (6b) is zero, since all values refer to centroid. The third part (6c) has its minimum for $\tilde{\mathbf{t}} = \mathbf{0}$ or

$$\mathbf{t} = \mathbf{c}_m - \mathbf{R}\mathbf{c}_d. \tag{7}$$

Therefore the algorithm has to minimize only the first term, and the error function is expressed in terms of the rotation only:

$$E(\mathbf{R},\mathbf{t}) \propto \sum_{i=1}^{N}||\mathbf{m}'_i - \mathbf{R}\mathbf{d}'_i||^2.$$

Theorem: The optimal rotation is calculated by $\mathbf{R} = \mathbf{V}\mathbf{U}^T$. Herby the matrices **V** and **U** are derived by the singular value decomposition $\mathbf{H} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$ of a correlation matrix **H**. This 3×3 matrix **H** is given by

$$\mathbf{H} = \sum_{i=1}^{N}\mathbf{m}'^T_i\mathbf{d}'_i = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix},$$

with $S_{xx} = \sum_{i=1}^{N} m'_{ix} d'_{ix}$, $S_{xy} = \sum_{i=1}^{N} m'_{ix} d'_{iy}$, ... The analogous algorithm is derived directly from this theorem.

Proof: See [2] or [9].
Finally, the optimal translation is calculated using eq. 7) (see also (6c)).

3.2 Computing Point Correspondences

As mentioned earlier, the strategy of ICP is to always use closest points. To speed up computation, kd-trees have been proposed [3]. kD-trees are a generalization of binary search trees. Every node represents a partition of a point set to the two successor nodes. For searching points we use optimized, approximate kd-tree. The idea behind this is to return as an approximate nearest neighbor the closest point in the bucket region where the query point lies. This value is determined from the depth-first search, thus expensive ball-within-bounds tests and backtracking are not used. Here, optimization means to choose the split axis during construction perpendicular to the longest axis to minimize the amount of backtracking.

A forest of kd-trees is used to search the point correspondences. For every color, i.e., semantic label, a separate search kd-tree is created. The algorithm computes point correspondences according to the color. E.g., points belonging to the wall are paired with wall points of previous 3D scans. Fig. 4 shows the point correspondences in case of semantic based matching (top) in comparison with normal closest point matching (bottom). The points at the change of colors are paired differently. Fig. 5 shows extracted slices of the kd-trees for the colors red and yellow.

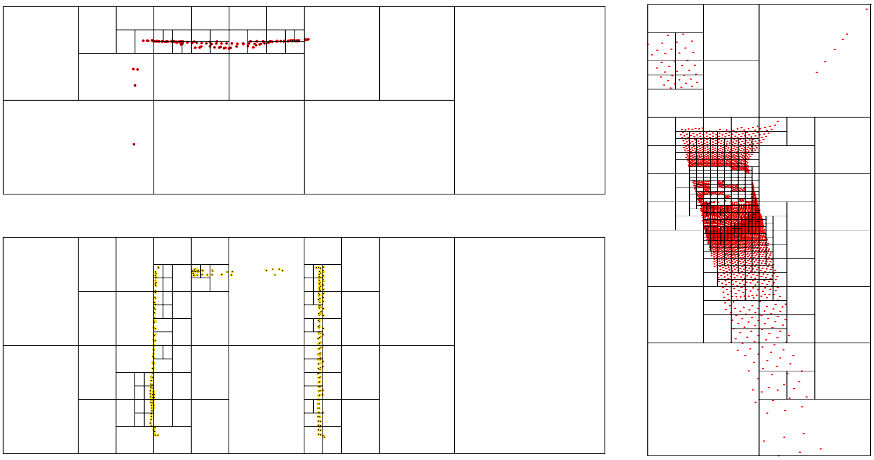


Fig. 5. A forest of kd-trees based on the semantic interpretation is used to compute the point correspondence. Left: Vertical slices through the trees (top: Ceiling points. Bottom: Wall points). Right: Horizontal slice using ceiling points.

Using semantic information helps to identify the correct correspondences, thus the number of ICP iterations for reaching a minimum is reduced. In addition, maximizing the number of correct point pairs guides the ICP algorithm to the correct (local) minimum leading to a more robost algorithm.

4 Results and Conclusion

The proposed methods have been tested on a data set acquired at RTS, Hannover. Fig. 3 shows a single 3D scan with semantic labeling. Fig. 6 presents the final map, consisting of five 3D scans, each containing 43440 points. Table 1 shows the computing time for matching of two 3D scans. Using semantically labeled points results in a speedup of up to 30% with no loss of quality.

Fig. 7 shows a detailed view of the ceiling. 3D points belonging to the lamp at the ceiling are also colored yellow. The correct match will be in no way affected

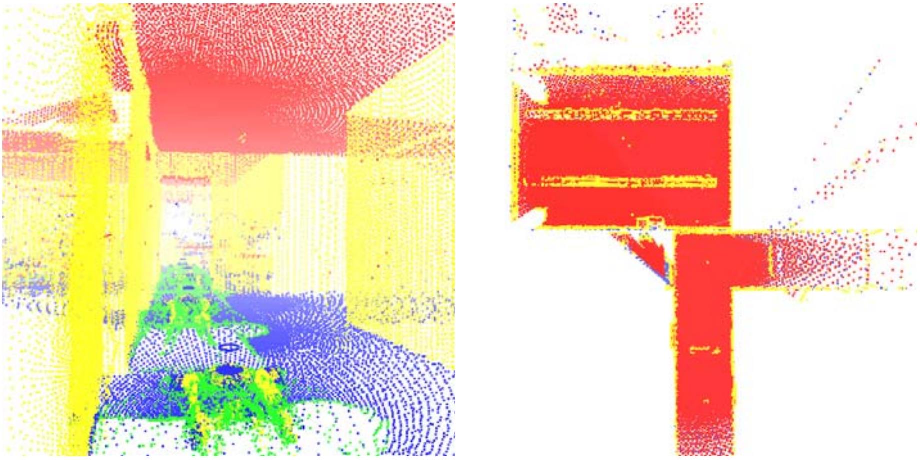


Fig. 6. The final 3D map of an office corridor / laboratory environment. The map consists of 5 3D scans and contains 217200 3D points. Left: Front view. Right: Top view.

Table 1. Computing time and number of ICP iterations to align all 32 3D scans (Pentium-IV-3200). In addition, the computing time for scan matching using reduced points are given. Point reduction follows the algorithm given in [10].

used points	search method	computing time	number of iterations
all points	*k*d-trees	17151.00 ms	49
reduced points	*k*d-trees	2811.21 ms	38
all points	forest of *k*d-trees	12151.50 ms	39
reduced points	forest of *k*d-trees	2417.19 ms	35

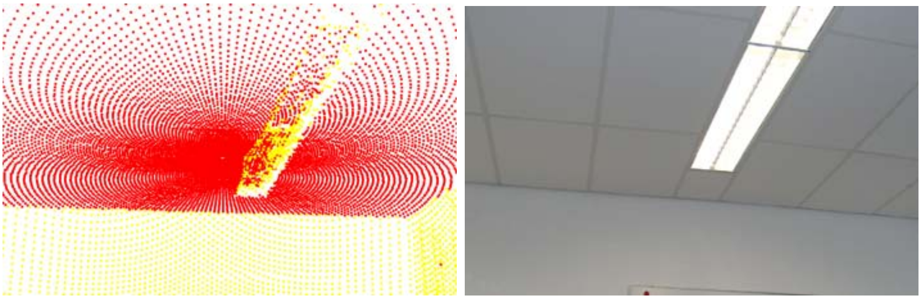


Fig. 7. Left: scanned 3D points of the ceiling including a lamp. Some 3D points of the lamp are marked yellow. The fingerprint like structure is the result of the scanning process. On plane surfaces the laser beam describes a circle. Right: Photo of the scene.

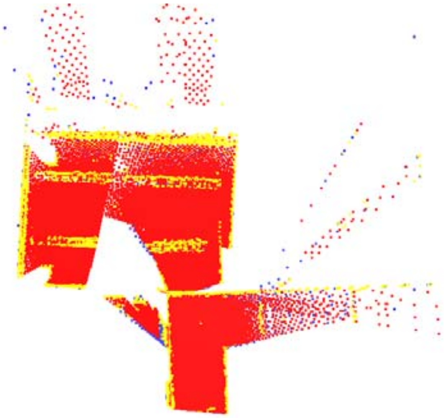


Fig. 8. Resulting 3D map (top view) if the scan matching algorithm uses only 3D points in front of the robot, i.e., the 3D scan is restricted to 180°.

by this fact. In fact, the semantic meaning is that data points of the lamp will be matched correctly with their correspondents.

Contrary to previous works, every single 3D scan is a full 360° scan. They are acquired in a stop-scan-go fashion to ensure consistency within the single 3D scans. In RoboCup Rescue the operator drives the robot and acquires 3D scans. In the 2004 competition we encountered that the overlap between two consecutive scans was sometimes too low, so that the operator had to intervene in the matching process. The new scanner will reduce this problem, since it provides backward vision. Fig. 8 shows the analogous map of Fig. 6 without backwards vision, i.e., the algorithm uses only points that lie in front of the robot. The 3D scans can no longer be matched precisely, the map shows inaccuracies for example at the lab door. In fact, doors and passages are a general problem of mapping algorithms, due to the small overlap. Fig. 9 shows the final map of an additional experiment with 9 3D scans and 434400 data points.

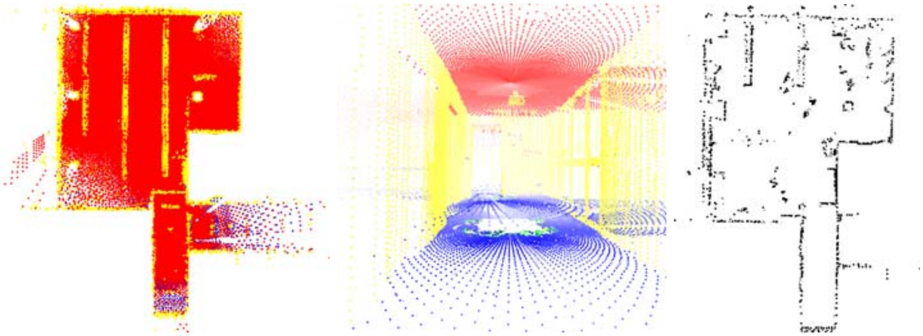


Fig. 9. Results of a second experiment. Left: 3D point cloud (top view). Middle: Some view of the points. Right: A floor plan extracted from the 3D model, i.e., only points with a height of 125 cm ± 15 cm are drawn.

This paper presented a robotic 3D mapping system consisting of a robot platform and a 3D laser scanner. The laser scanner provides a 360° vision; the scan matching software, based on the well-known ICP algorithm, uses semantic labels to establish correspondences. Both approaches improve previous work, e.g., [9, 10], in terms of computational speed and stability.

The aim of future work is to combine the mapping algorithms with mechatronic robotic systems, i.e., building a robot system that can actually go into the third dimension and can cope with the red arena in RoboCup Rescue. Furthermore, we plan to include semi-autonomous planning tools for the acquisition of 3D scans in this years software.

References

1. P. Allen, I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer. AVENUE: Automated Site Modeling in Urban Environments. In *Proceedings of the third International Conference on 3D Digital Imaging and Modeling (3DIM '01)*, Quebec City, Canada, May 2001.
2. K. S. Arun, T. S. Huang, and S. D. Blostein. Least square fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698 – 700, 1987.
3. P. Besl and N. McKay. A method for Registration of 3–D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239 – 256, February 1992.
4. P. Biber, H. Andreasson, T. Duckett, and A. Schilling. 3D Modeling of Indoor Environments by a Mobile Robot with a Laser Scanner and Panoramic Camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, Sendai, Japan, September 2004.
5. A. Georgiev and P. K. Allen. Localization methods for a mobile robot in urban environments. *IEEE Transaction on Robotics and Automation (TRO)*, 20(5):851 – 864, October 2004.
6. D. Hähnel and W. Burgard. Map Building with Mobile Robots in Populated Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, Lausanne, Switzerland, September 2004.

7. D. Hähnel, W. Burgard, and S. Thrun. Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot. In *Proceedings of the fourth European workshop on advanced mobile robots (EUROBOT '01)*, Lund, Sweden, September 2001.
8. A. Howard, D. F. Wolf, and G. S. Sukhatme. Towards Autonomous 3D Mapping in Urban Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, Sendai, Japan, September 2004.
9. A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, K. Pervölz, M. Hennig, K. R. Tiruchinapalli, R. Worst, and T. Christaller. Mapping of Rescue Environments with Kurt3D. In *Proceedings of the IEEE International Workshop on Rescue Robotics (SSRR '05)*, Kobe, Japan, June (submitted).
10. A. Nüchter, H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 6D SLAM with an Application in Autonomous Mine Mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1998 – 2003, New Orleans, USA, April 2004.
11. V. Sequeira, K. Ng, E. Wolfart, J. Goncalves, and D. Hogg. Automated 3D reconstruction of interiors with multiple scan–views. In *Proceedings of SPIE, Electronic Imaging '99, The Society for Imaging Science and Technology /SPIE's 11th Annual Symposium*, San Jose, CA, USA, January 1999.
12. H. Surmann, K. Lingemann, A. Nüchter, and J. Hertzberg. A 3D laser range finder for autonomous mobile robots. In *Proceedings of the of the 32nd International Symposium on Robotics (ISR '01)*, pages 153 – 158, Seoul, Korea, April 2001.
13. S. Thrun, D. Fox, and W. Burgard. A real-time algorithm for mobile robot mapping with application to multi robot and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, San Francisco, CA, USA, April 2000.
14. R. Triebel, B. Frank, J. Meyer, and W. Burgard. First steps towards a robotic system for flexible volumetric mapping of indoor environments. In *Proceedings of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '04)*, Lisabon, Portugal, July 2004.
15. O. Wulf, K. O. Arras, H. I. Christensen, and B. A. Wagner. 2D Mapping of Cluttered Indoor Environments by Means of 3D Perception. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, pages 4204 – 4209, New Orleans, USA, April 2004.
16. O. Wulf, C. Brenneke, and B. A. Wagner. Colored 2D Maps for Robot Navigation with 3D Sensor Data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '04)*, Sendai, Japan, September 2004.
17. O. Wulf and B. A. Wagner. Fast 3D Scanning Methods for Laser Measurment Systems. In *Proceedings of the International Conference on Control Systems (CSCS '03)*, Bucharest, Romania, July 2003.

The Color and the Shape: Automatic On-Line Color Calibration for Autonomous Robots

Ketill Gunnarsson, Fabian Wiesel, and Raúl Rojas

Freie Universität Berlin,
Institut für Informatik,
Takustraße 9,
14195 Berlin, Germany
{ketill, wiesel}@inf.fu-berlin.de

Abstract. This paper presents a method for automatic on-line color calibration of soccer-playing robots. Our method requires a geometrical model of the field-lines in world coordinates, and one of the ball in image coordinates. No specific assumptions are made about the color of the field, ball, or goals except that they are of roughly homogeneous distinct colors, and that the field-lines are bright relative to the field. The classification works by localizing the robot(without using color information), then growing homogeneously colored regions and matching their size and shape with those of the expected regions. If a region matches the expected one, its color is added to the respective color class. This method can be run in a background thread thus enabling the robot to quickly recalibrate in response to changes in illumination.

1 Introduction

Color classification in the RoboCup Mid-Size League usually involves tedious calibration procedures. A typical approach is to manually define which parts of the color space correspond to a color class using some kind of GUI tool. This involves capturing images from different positions on the field, defining the color-boundaries between color classes, or classifying individual color pixels into one of the color classes. This method is error-prone and time consuming. Furthermore, a classification obtained at one point can fail at another, if the lighting conditions are different. For this reason, all objects in the environment are strictly color-coded and the organizers try to provide lighting that is as steady, bright and uniform as possible.

Our method remedies the problem by automatically classifying regions of homogeneous color into the following four color classes: field, ball, yellow goal, and blue goal. Regions that do not fit the criteria of any of the classes are not classified and can be considered obstacles. The white field-lines are detected without the use of color information, and can be identified as non-obstacles. The output of the method is essentially a separate list of color values for each color class. These lists grow over time, as more and more colors are classified. In our

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 347–358, 2006.

application, we store these lists as a look-up table using the full 24-bit YUV color depth.

The method can run on-line during a game to compensate for changes in lighting, and is able to calibrate a whole image from scratch and error-free in 1-2 seconds. It is robust against false classification even with robots, humans and other objects cluttering the field.

For each color class the method consists of the following steps:

- localize the robot on the field using edge detection(see section 2).
- loop:
    - Grow a homogeneous color region (see section 3).
    - Compare the grown region's size and shape to the size and shape of the corresponding expected region (see section 4).
    - **if** the grown region is within the boundaries of the expected region, and fills more than a certain percentage of the expected size:
        - add all the colors in the grown region to the corresponding color class.
    - **else**
        - add no colors to the color class.

The homogeneity thresholds for the color growing are computed automatically (see section 5).

Related work includes [5], which presents a method for off-line, semi-autonomous color-calibration, implemented in the Mid-Size League. RETINEX, a biologically-inspired algorithm is used for improving color constancy, and k-means clustering is used for the adaptation of color classes. HSV thresholds are found from the clusters that determine each color class, which are then manually mapped to symbolic colors. This method analyzes the vicinity of colors in the color-space and then forms clusters that represent color-classes. In contrast, the method presented here relies on the expected geometrical shape of the objects belonging to a color-class and does not rely on color-space analysis. Furthermore, our method is fully automatic, not requiring manual mapping to symbolic colors.

A method called KADC (Knowledge-based Autonomous Dynamic Colour Calibration) is presented in [9]. KADC is a method for autonomous on-line color classification, implemented in the Sony Legged League. KADC also utilizes the geometric information of the field to define color classes, and then updates them with the help of a color cluster similarity metric called EMD. Our method is also based on geometric knowledge of the field, but we combine this with the color-less detection of the robot's position. We then update the classification using only geometric criteria without having to incorporate any color similarity metrics to previously established classification. This enables us to deal with an abrupt increase/decrease in illumination, which is reported to be troublesome when applying KADC(by [9]). What further differentiates our method from [9] is that we also handle the ball color class.

In [6], Juengel et al. present an efficient object detection system (also implemented in the Sony Legged League) which only requires a linear division of the UV-color space. This system is extended in [7] to obtain a more fine-grained classification. The division is calibrated automatically, and the objects are heuristically detected. However, such a linear division and the use of heuristics may be inadequate for more demanding situations. For example when color classes are not linearly separable, or when numerous color classes are required.

Austermeier et al. ([10]) find the color-correspondence between two illumination settings by using self-organizing feature maps (SOM) in color-space. Two SOMs are built, one for the cloud of color points under the initial reference illumination and another one for the new illumination settings. The distance between corresponding grid locations in the two maps is then used as a correction vector for the set of color points belonging to that volume element. This solves the problem of maintaining a classification, but does not generate it. Unfortunately, the method is also very computationally expensive.

The method we present was tested on our Mid-Size robots, which are equipped with an omni-directional vision system and use conventional sub-notebooks for processing (see [4]). We now proceed with a step-by-step description of our calibration method. We then describe how the thresholds used for the color growing are found and adapted. Finally, we present experimental results.

2 Color-Less Localization

In order to allow image coordinates to be transformed into world coordinates, the robot needs to localize itself in an initialization step. We do this by using a region tracker (described in [2]), which stops growing a region when a possible field-line is encountered. The stopping criterion is based on the assumption that

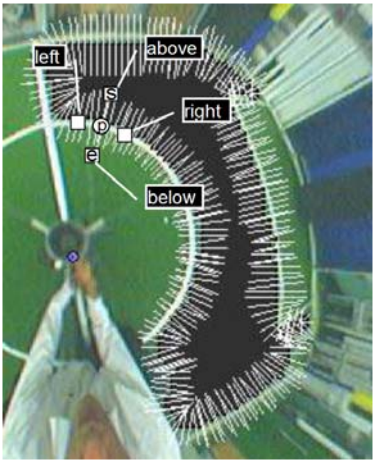


Fig. 1. A tracked region (painted black) and white scan-lines along its edges. Here a green region is being tracked. s and e are the start and end-points of the scan-line. p is the actual point to be checked for edge criteria, and the points marked above, below, left and right, are its neighbors used to determine the brightness around p.

a field-line is bright compared to the neighboring points. In our implementation we use 4 pixels to the left, right, above and below the actual point p, which have a pixel distance from it corresponding to the expected field-line width (see Fig. 1). We decide that a field-line is found if p is one standard deviation σ brighter than at least two of its neighbors, where σ is the standard deviation of the brightness of those pixels in the image which correspond to points on the field. The value of σ is calculated on-line by sampling a certain number of random pixels in the image.

Now we need to extract the pixels that coincide with the points of the white field-lines. To accomplish this, we search for dark-white-dark transitions on short scan-lines perpendicular to the edges of each of the tracked regions. This is done in the following manner: find the brightest point p on the scan-line. Inspect the endpoints s and e of an extended scan-line centered on p and having length twice the expected field-line width (the length was tuned experimentally, the idea is that s and e do not lie on the field-line, see Fig. 1). If p is σ-brighter than s and e, declare it a white pixel corresponding to a point on a field-line.

The set of points obtained in this way is the input for our localization algorithm. The localization exploits the presence of certain features in the field's line-model (center circle, corners, etc. see Fig. 2) and localizes the robot using them and a force field matrix (Hundelshausen et al. describe this localization technique in [1] and [3]). The localization we obtain this way is uniquely determined up to the symmetry of the field because we have no information about the two goal box colors. Nonetheless, the method can proceed without it, as we explain in the next section.

A drawback of this localization is that more false field-line points are found than with our conventional localization, which tracks green regions. It is also potentially slower since more pixels are processed. Even though we could localize

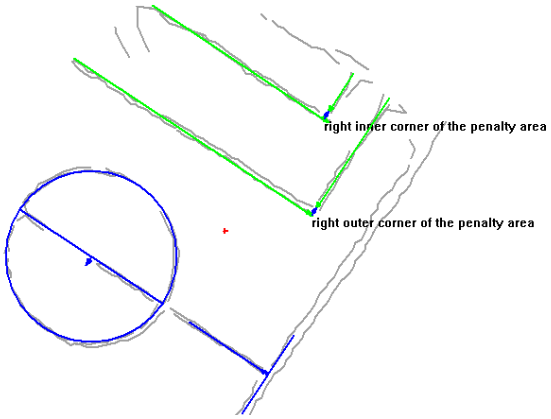

right inner corner of the penalty area
right outer corner of the penalty area

Fig. 2. Example of features found in the field-line contours. Here the center circle, a T-junction and the inner and outer right penalty area corners have been successfully detected. With these features the robot can localize itself on the field.

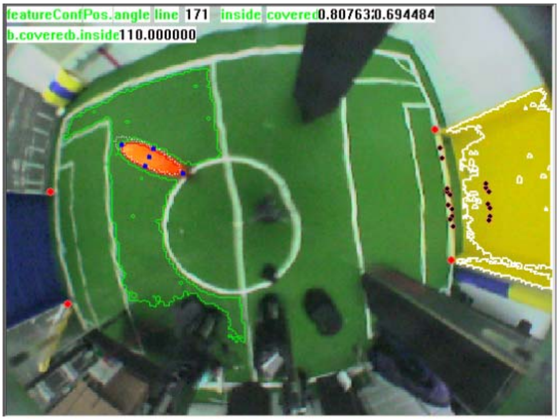
the robot by calibrating the goal colors only (to break the field's symmetry), there is still a need for calibrating the color of the field. Without it we cannot identify obstacles on the field.

3 Choosing Regions by Color Growing

The second step is to grow homogeneous color regions. It is not important to start growing a region at a specific pixel, but choosing them intelligently can accelerate the classification. This is achieved by building different sets of starting pixels for each expected region. Subsequently, one pixel is chosen at random from each set and separate color region growing processes are started (see figure 3). The grown color regions are then passed along for further validation (see section 4). Different pixel criteria are required to obtain the various pixel-sets we use for the expected regions of the field, the ball, and the goals.

Since the green field color usually covers a large area of the image, a possible method to obtain the field pixel-sets would be to pick a certain amount of randomly chosen pixels and assign them to the pixel-set of the expected region they correspond to. Instead, we gather pixels from our field-line detection procedure which provides us with pixels that are close to a line, but not on one. These pixels - excluding the ones corresponding to points lying outside the field, are then assigned to the pixel-set of the expected world region they correspond to (there are 10 such field-regions, see Fig.4).

A goal-box pixel-set consists of pixels corresponding to points lying behind one of the two goal lines in the field-model. The two goal-box pixel-sets are separated by the spatial location of the goals. We decide later to which goal-box the two sets belong (see section 4).



```
featureConfPos.angle line 171  inside covered0.80763:0.694484
b.coveredb.inside110.000000
```

Fig. 3. Regions grown successfully for an expected ball region, one of the expected field regions, and an expected goal region. The grown goal region is enclosed by a white curved line, the ball's by a red curved line and the field's by a green curved line.

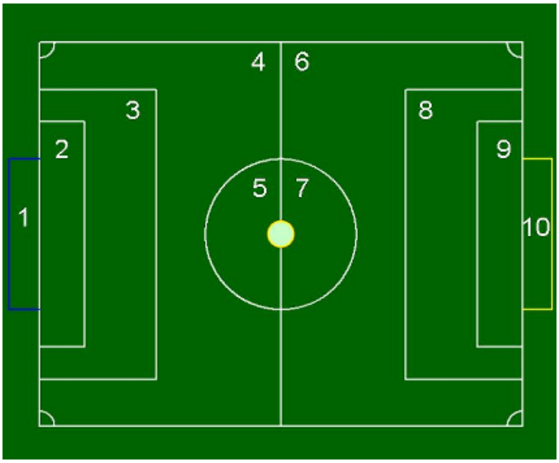


Fig. 4. The 10 expected field regions

Since the ball can be small in the image and we don't know where to look for it (even if the robot's position is known), it pays off to pick the pixel-set for the ball color-growing carefully in order to make its classification faster. The procedure we use only considers pixels corresponding to field-points (because the ball is on the field). It then checks if a pixel has either been classified as the ball color in a previous iteration of the calibration procedure, or if it could be the center of an expected ball at that point. If this is the case, we add the pixel to the ball pixel-set. Essentially this is a form of pre-validation that verifies if a region starting from this pixel could ever grow into the expected ball at this pixel. It does this by checking if pixels along the axes of the expected ball ellipse are unclassified.

Growing a homogeneous color region works in the following manner: Starting with a pixel p, neighboring pixels are inspected. If their color is within a certain homogeneity threshold with respect to the color at p, they are added to the color region. The neighbors of the newly added pixels are inspected in the same manner, always comparing them to the color of the first pixel p. The homogeneity thresholds are adapted automatically (see section 5).

4 Validating Grown Color Regions

After picking one point from each pixel-set and growing separate color regions (one region for the ball, ten regions for the field, and two regions for the goals), we need to make sure that they belong to the corresponding expected regions. To ensure this, the regions have to pass through validation criteria. The criteria are similar for each color class, and are based on the following observation: if a grown color region r is totally inside and covers an expected region of a color class C, then the colors in r are members of C. The expected regions are defined assuming ideal conditions such as an obstacle-free field and perfect self-localization.

In the case of the field color class, we can define 10 expected regions in world coordinates using the current field-model (see Fig. 4). In accordance with our general guidelines, a grown field-color region should lie entirely inside one of the expected regions. After checking that the region fulfills this criterion, we check if it covers enough of the expected region. In our implementation we require a field-color region to cover 70% of the corresponding expected region, and all of its points to lie inside it. If the criteria are not fulfilled, we do not add any colors from the region to the field color class in this iteration.

In the case of the goal-box color classes, we can use the field line-model and the robot's position to calculate at which angle in the image we expect a grown goal-box-color region to appear. Furthermore, we know that this region cannot be inside any of the expected field regions. In our implementation we require a goal-color region to lie between the angle defined by the goal-box's posts, and to cover 70% of the angle. We also require the goal-box to be clearly visible given the current position of the robot, e.g. that the expected angle to the left and right posts is sufficiently large. Furthermore, no point of the region can lie in any of the expected field regions. If the criteria are not fulfilled, we do not add any colors from this region to the respective goal-box color class in this iteration.

Once a grown goal-color region has been successfully validated, and its colors have been associated with one of the arbitrarily chosen goal-boxes, the symmetry of the field has been broken, and the sides can be labeled and recognized. Future validated goal-color regions will therefore be assigned to the correct goal-box color class. This is based on the assumption that the robot does not de-localize and flip sides, while the illumination simultaneously changes to prevent goal-identification. However, since there is a convention in RoboCup to paint one of the goals yellow, and the other one blue, we compare a grown goal-color region to a blue and a yellow reference color in our implementation. We then add it to the class whose reference color is closer to the region's mean color. This automates the setup of the robot and also increases the robustness of the classification.

In the case of the ball color class, an expected ball region in the image has an elliptic form where the size of the minor and major axis depends on the distance from ball to robot. We represent the expected ball by storing ball-fitting ellipses at different distances from ball to robot. One ellipse data entry consist of the pixel distance from the robot to the center of the ellipse (the robot being in the center of the image), as well as the minor and major axis of the ellipse, measured in pixels. In our implementation we require all pixels in a ball-color region to be inside the expected ball region, and the area to be more than 40% of the expected area. If the criteria are not fulfilled, we do not add any colors from the region to the ball color class in this iteration.

5 Adaptive Thresholds for Color Growing

The thresholds for color growing are in general not the same for each color class, and vary with lighting conditions. Furthermore, it is advantageous to use various thresholds for the same color class in one and the same scene. This is especially

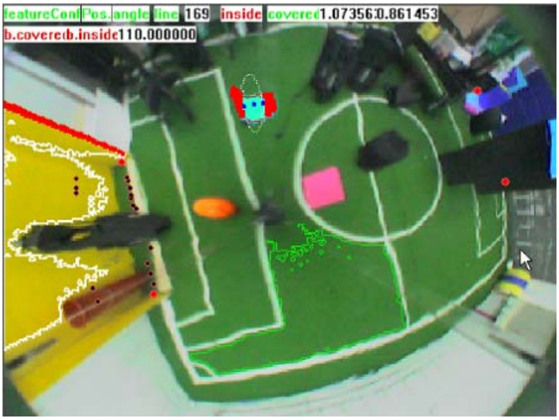


Fig. 5. Grown regions which fail to meet the validation criteria. Pixels of the goal-box and ball regions are outside the expected region, or the field region does not cover a required percentage of the expected region.

advantageous in the case of the field class, because it is large and can therefore have wide variations in color homogeneity. Accordingly, we deploy three separate sets of thresholds, one for each expected region of the field, the ball and the goals. These sets are initialized with a constant amount of random thresholds. The thresholds are then adjusted with the help of the validation criteria outlined previously in section 4. Essentially, this means decreasing the threshold if the region grown using it was too big, and increasing it, if it was too small.

Before a region is grown, a threshold is picked at random from the corresponding set. If a certain threshold was involved in a successful growing, it "survives" and is still part of the set in the next iteration of the calibration procedure. If a region growing has failed a certain number of times using the same threshold, the threshold "dies" and a new randomly initialized threshold takes its place in the set. Each time a region grown with a threshold is too big, we decrease the threshold by a small random amount. If the region is too small, we increase the threshold, and try to grow a region at the same point. We continue increasing the threshold until the region is successfully grown, or grows outside the expected region.

6 Results

The method we present in this paper is able to adapt to changes in illumination in a few seconds. We tested the method on our robots which are equipped with a lightweight laptop having a Pentium III M 933 MHz processor, and 256 MB RAM. For the run-time tests of the method we recorded how long it took to adapt to an abrupt change in illumination. The scene we processed can be considered a standard RoboCup scene with the ball in view, and a robot in the goal, except that a foreign pink piece of paper is present in the field. Note that it will not be classified since it does not fit any expected region. The results of the classification

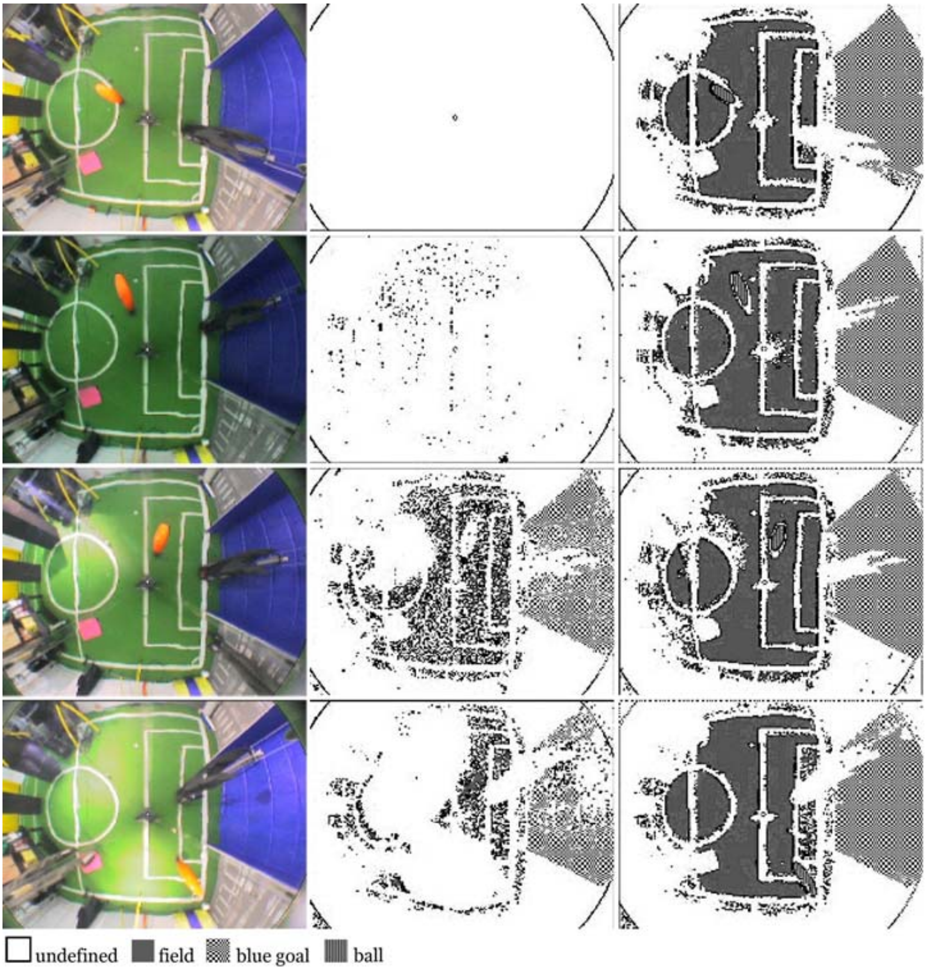


Fig. 6. Row 1 to 4 (counting from top to bottom): Original images on the left, static classifications from previous illumination setting in the middle, and the result of the automatic classifications on the right. The CPU-time it took the method to produce the automatic classifications (right), starting with the classifications achieved from the prior illumination (middle) for row 1-4 was: 0.5, 0.8, 2.0, and 2.9 seconds, respectively. The color code is: white = unclassified, gray = field, check-board-pattern = blue goal, diagonal-pattern = ball.

can be seen in Fig.6. The first column shows the original images. The second column shows a segmented image using the automatic classification from the previous illumination setting without adjusting to the new illumination (the first row has no previous classification). As we can see, under the new illumination the color segmentation is poor. The third column shows a segmented image after adapting the classification from the previous scene with our automatic calibration method.

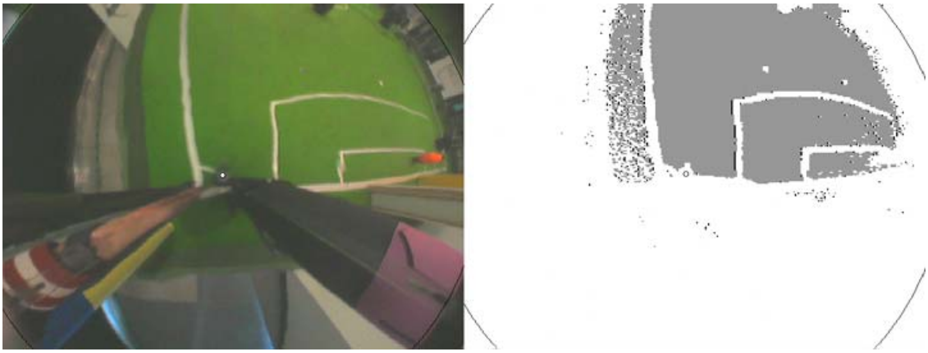


Fig. 7. The classification (on the right) was obtained after a few second with no previous classification. Here the ball and the goals are too far away to be calibrated so only the field can be calibrated. The color code is: white = unclassified, gray = field.

The runtime for the adaptation for row 1-4 was: 0.5, 0.8, 2.0, and 2.9 seconds, respectively. The current implementation does not try to match the field-regions inside the goal (regions 1 and 10 in Fig. 4), and therefore its colors are not classified in any of the scenes. The regions on the other side of the field are also not classified since they are obscured, and hence can not be matched to the corresponding expected regions. The first row demonstrates the classification under a mixed neon - and indirect floodlight. All regions that are clearly visible have been successfully classified. The same goes for the second row, which displays a darker scene with neon lighting only. The third row shows a classification under neon lighting, and with one floodlight directed down on the field. Here the method fails to classify some of the brightest green colors lying under the floodlight after 2.0 seconds, but after letting the method run for about 12 seconds, the classification improves (not illustrated), without managing to classify some of the extremely bright green colors. The fourth and last row of images was captured under neon lighting and with three floodlights directed down on the field. A successful classification of this scene was obtained after 2.9 seconds. Note however, that the colors of the inner penalty area are not classified. This is due to the fact that the goalie is placed in the middle of it, and thereby cuts the homogeneous color region in half. It can therefore not be matched properly to the expected area of the inner penalty area.

Fig.7 illustrates the performance of the method where the robot is close to a border line and sees a large area outside the field. The classification (on the right) was obtained after a few second with no previous classification. Here the ball and the goals are too far away to be calibrated.

7 Future Work and Summary

The method encounters certain problems when colors belonging to a color class are present in regions not belonging to the class. This is for example the case when the blue goal is so dark that some colors in it are indistinguishable from the ones appearing in robots. In this case, a very dark-blue or black region is grown inside the goal, which is found to correspond to the expected goal region. The method then defines these colors as belonging to the blue goal color-class even though they are encountered in robots as well. Another potential weakness of the method is that a color does not become “outdated”, e.g. a color cannot loose a previous classification. This can present a problem when the lighting is changed, for example from white neon lighting to a more warm, yellow lighting. Now, colors that were previously classified as the yellow goal can appear on the white field-lines. A possible solution would be to mark a color as not belonging to a class if it occurs in an unexpected region. Another approach used in [9], would be to incorporate a color decay factor.

A method for tuning the camera-parameters is presented in [8], and could be combined with our method to enable the robot to operate in a wider range of lighting-conditions. The “ground truth” (manually defined color classes) needed in that method could be provided by our automatic calibration.

In this paper we presented a method that can be used for automatic color calibration of autonomous soccer playing robots. It is based on a color-less localization of the robot, a geometric line-model of the field and a geometric model of the ball. The method needs no manual calibration and can deal with various difficult lighting conditions that change abruptly over time. It can be integrated into existing systems and will be used by our robots at RoboCup 2005 in Osaka.

References

1. Felix von Hundelshausen, Michael Schreiber, Fabian Wiesel, Achim Liers and Raúl Rojas: *MATRIX: A force field pattern matching method for mobile robots*, Technical Report B-08-03, Freie Universität Berlin, Institute of Computer Science, Takustr. 9, 14195 Berlin, Germany, available at: http://robocup.mi.fu-berlin.de/docs/matrix.pdf, link checked: 2.feb 2005.
2. Felix von Hundelshausen, and Raúl Rojas: *Tracking Regions*, in Daniel Polani et al.(editors):RoboCup-2003: Robot Soccer World Cup VII (Lecture Notes in Computer Science), Springer, 2004. Available at: http://robocup.mi.fu-berlin.de/docs/RegionTrackingRoboCup03.pdf, link checked: 2.feb 2005.
3. Felix von Hundelshausen, *A constructive feature-detection approach for mobile robotics.* Proceedings of the RoboCup 2004 International Symposium, Lisboa, Italy July 5-7, 2004.
4. Felix von Hundelshausen, Raúl Rojas, Fabian Wiesel, Erik Cuevas, Daniel Zaldivar, and Ketill Gunnarsson: *FU-Fighters Team Description 2003*, in D. Polani, B. Browning, A. Bonarini, K. Yoshida (Co-chairs): RoboCup-2003 - Proceedings of the International Symposium.
5. Mayer, G., Utz, H., Kraetzschmar, G.K.: *Towards autonomous vision self-calibration for soccer robots.* IEEE/RSJ International Conference on Intelligent Robots and Systems (2002) 214-219.
6. Jüngel, M., Hoffmann, J., Lötzsch, M. (2004). *A real-time auto-adjusting vision system for robotic soccer.* In: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence, Padova, Italy, 2004. Springer (to appear).

7. Matthias Jüngel. *Using Layered Color Precision for a Self-Calibrating Vision System.* Daniele Nardi, Martin Riedmiller, Claude Sammut, José Santos-Victor (Eds.): RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Computer Science 3276 Springer 2005, ISBN 3-540-25046-8.
8. Erio Grillo, Matteo Matteucci, Domenico G. Sorrenti: *Getting the Most from Your Color Camera in a Color-Coded World.* Daniele Nardi, Martin Riedmiller, Claude Sammut, José Santos-Victor (Eds.): RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Computer Science 3276 Springer 2005, ISBN 3-540-25046-8.
9. D. Cameron, N. Barnes, *Knowledge-Based Autonomous Dynamic Colour Calibration*, in Proc. Robocup Symposium, 2003, Padua, Italy, July, 2003. RoboCup 2003 award-winning paper: *Engineering Challenge Award.*
10. Austermeier, H., Hartmann, G., Hilker, R.: *Color-calibration of a robot vision system using self-organizing feature maps.* Artificial Neural Networks - ICANN 96. 1996 International Conference Proceedings (1996) 257-62.

ChipVision2 – A Stereo Vision System for Robots Based on Reconfigurable Hardware

Lars Asplund, Henrik Bergkvist, and Thomas Nordh

Mälardalen University

Abstract. A system utilizing reconfigurable hardware of 1 million gates and two CMOS cameras is used in an image analysis system. The system is a part of a sensor system for a robot, and can deliver data about the robots position as well as relative distances of other objects in real time.

1 Introduction

Many algorithms used in digital signal processing for image analysis require large computational resources. The most commonly approach is to use a DSP (Digital Signal Processor), such as Texas Instruments TMS320C6201 and C6701, Philips TriMedia TM1100 and Analog Devices Sharc ADSP 21160M. These processors are variations of SIMD architectures, and they contain several processing units. By the internal pipelining and by using the processing units in an optimal way quite high throughputs can be achieved. Standard PCs are naturally used for image analysis, but for real-time applications these systems has in the past not been powerful enough.

Reconfigurable hardware has most often been regarded as a means for speeding up standard processors or DSP's. Operating systems implemented in an FPGA as an accelerator can in a Real-Time system guarantee that the system is fully predictable [16].

There have also been attempts to use reconfigurable hardware for Image Processing, [12] and [1].

The current project aims at building a stereo vision system (a vision system using image analysis in configurable hardware – FPGA, Field Programmable Gate Array) in the new robot design, Aros. ChipVision will analyze the output from two digital cameras in order to find objects (football, goal, opponents, lines etc.). The output from ChipVision will be data to the main computer in the robot about distance and angles to found objects and the estimated position and direction of the robot itself.

The design is based on an estimated throughput of 15-30Hz. ChipVision is mounted on a free rotating disc, rotated by a stepper motor. Communication with other parts of the robot is by means of an optical CAN–bus.

2 Overview of the Camera System

The FPGA used in this work is equivalent to one million gates and is mounted on a six layer PCB. On the board there is also 64 MB RAM and 32 MB Flash

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 359–370, 2006.

EPROM. The two CMOS cameras are from OmniVision Technologies Inc. type OV7610 and are directly coupled to the FPGA. The output from the camera is either RGB or YUV. In the current design only 8 wires are used for data transfer, and the format of data used is RGB. The cameras are used in non-interlaced format and the colour filter pattern is the Bayer-pattern, this implies that the green-value is new for each pixel, but the red and blue are alternating. We treat two pixels as one pixel, and thus giving full information of both Red, Green and Blue. The camera chip also has an I^2C bus. This serial bus uses two wires and it can be used to control the camera. Typical controls are gain and white balance.

The internal hardware design (further on referred to as the program) of the FPGA is stored in the Flash–memory in a so called bit–file. A CPLD (Complex Programmable Logic Device) handles the loading of the program into the FPGA. There is a selector–switch with which one of eight programs can be selected for loading. This selection can also be performed by the micro controller (AVR ATmega16) mounted on the PCB for the disc. One of the eight programs is a program for loading the Flash. Thus, each camera–disc can have seven different programs stored in the Flash, thus seven different algorithms can be selected during run-time. The time to load a new program is less than one second.

In this paper three different programs are described. The first program is using one camera for finding the ball and the other camera for finding white lines. The two cameras have different gain–settings for the individual colours. A stereo matching algorithm is the second program. Detected feature transitions from a row for both cameras are stored in a CAM for fast comparison and matching. The stereo matching program successfully detects and matches feature transitions and depth information, this can be obtained from two camera images in real-time. The third program implements Harris and Stephens combined corner and edge detection algorithm, and it uses both cameras. With a frame–rate of 13Hz corners are outputted from left and right cameras.

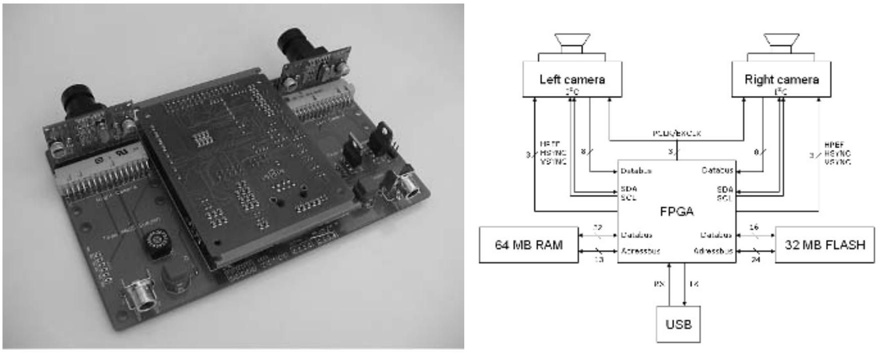


Fig. 1. PCB holding the FPGA-board and the two cameras and the corresponding block diagram

3 Algorithms for Image Analysis

The proposed solution is based on algorithms that are suitable for hardware implementation, i.e. the inherent parallel nature of an FPGA can be fully utilized. In cases where an algorithm is described in a sequential language such as C, there are tools that can convert the C program into a hardware description language such as VHDL [7]. In ref [2] a language called Handel–C has been used. Another way is to use a language that has been developed for the application domain, such as SA–C, a Single Assignment variant of C. In ref [4] this language has been used to implement image analysis algorithms in an FPGA. The performance is in ref [5] compared to the performance of a Pentium processor. The FPGA has in this test been used as a co processor to a rather slow processor. Due to communication the performance in some of the tests are not convincing, but for other tests, the FPGA outperforms the Pentium processor. In ref [2] a true comparison between a high speed DSP (TMS320C44) and a 400k gate FPGA is shown. The number of clock cycles in the FPGA are 64 for a cluster analysis. The corresponding number in the DSP is 16000 cycles. With cycle times of 30 ns for the FPGA and 33 ns for the DSP the time per cluster is 21μs for the FPGA and 5.3 ms for the DSP.

The vision system presented in this paper is aimed to be used in a robot system, and there are numerous industrial applications where vision is an important part. A survey of applications and tools is given in the paper presented by Malamas et al [10].

The size of an FPGA today is far above 1 million gates. This amount of gates allow the full system to be implemented on one FPGA. This can be compared to the ARDOISE system [3], which is a modular system based on smaller FPGA's.

In [9] a hardware setup which resembles our system is described. Instead of using an analogue video source we are using a camera with a digital output. Since our system communicates with the host computer using the CAN–bus, there is no need for a second FPGA. Due to increased performance of circuits, our FPGA and the memory chips have better performance both in speed and number of gates and memory size.

In [8] the ISODATA algorithm is presented, and this algorithm does a mapping from RGB–colour information into classes. The algorithm is used as a first stage and the number of bits in the further processing is reduced by a factor of six. The filter solution in this project is based on a two–dimensional filter, corresponding to the filter used in [11].

For the higher level analysis some version of the object recognition models presented in [6] is used. Still for high performance it is important to have a pipe–line architecture and in the stage after classifying the data there are tables for start and stop pixels for all possible objects.

3.1 Overview of Algorithms

The following steps are the main algorithms of the image analysis. First the RGB representation is transferred into HSI representation. One reason for this is that the analysis should not be sensitive for differences in light or shading effects. The HSI representation includes a separate variable/channel for intensity I. H is the variable which represents colour tone. H is usually represented by a disc with 0-255 levels . By this representation the colour is described with in one dimension instead of three. From this representation the interesting objects could be thresholded out using their colour values.

Every pixel is given a number indicating their class, depending on their class and the class of the neighbors the pixels are labeled. Noise reduction is performed to minimize the number of pixels with erroneous object classification.

3.2 RGB to HSI Conversion

Mathematical formulas for transformation from RGB to HSI. These formulas take for granted that the RGB values are normalized to [0, 1].

$$I = \frac{R+G+B}{3} \tag{1}$$

$$S = I - \frac{3}{R+G+B} \min(R, G, B) \tag{2}$$

$$H = \arccos\left[\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right] \tag{3}$$

If $B > G$ then $H = 360^o - H$.

For Robocup the Saturation is not required since the colours are well separated in the Hue-circle. Thus only the H channel (and maybe the I channel) are relevant for finding interesting objects. S never needs to be calculated. H is a disc with radius 0-360 and can be represented as 256 gray levels (8 bits).

The equations above are not well suited for implementation in VHDL. The formula 3 is quite complex. Since the resulting H–value is to be represented as an 8–bit value the following algorithm which is a simplification of the Kender's algorithm [17] has proved to have high enough accuracy. In this implementation the full circle is represented between 0 and 252.

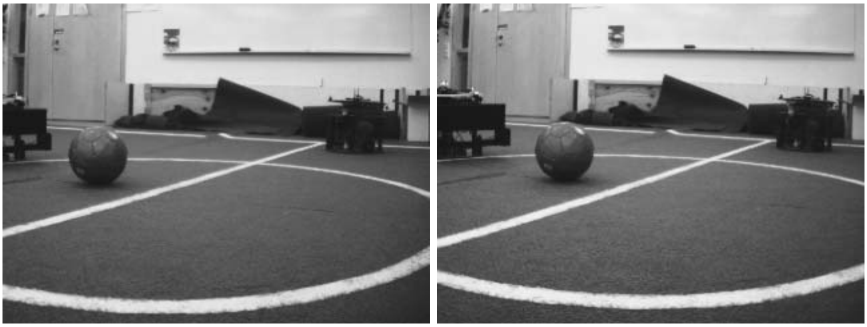


Fig. 2. The original pictures from the camera (RGB)

```
 Max = MAX (R,G,B);        H'=42*(MID(R,G,B)-MIN(R,G,B))/Max
 if R=Max              if G=Max              if B=Max
  if G>B                if R>B                if R>G
    H=H'                  H=84+H'               H=168+H'
  else                  else                  else
    H=252-H'              H=84-H'               H=168-H'
```

By sorting the three values of R,G,B the right sector (6 in total) of the colour circle can be found. The range 0 to 255 is not suitable since it can not be divided by 6. The closest value is therefore 0 to 252. Within in each sector the linear expression H' is accurate enough for finding the H-value within the sector.

3.3 Segmentation

The segmentation will take place concurrently as the calculation of H.

1. Is the pixel white? $I > Th_{white}$ gives white
2. Is the pixel black? $I < Th_{black}$ give black
3. Calculate H for pixel N.
4. Segment using multiple thresholds, etc.
   - $x > H$ or $H > X$ Red
   - $y < H < Y$ Orange
   - $z < H < Z$ Green
   - $u < H < U$ Yellow
   - $v < H < V$ Light blue
   - $r < H < R$ Magenta

Each pixel is now represented by its class ID. There are eight different colour classes and one background class. These can be represented by 4 bits. What is left is a picture matrix with 16 gray levels.

3.4 Noise Reduction

This algorithm describes how noise reduction is performed. This step is performed once the segmentation is completed. A median filter sets a pixels class value based on its neighbor's class values. Of this follows that the image will be more homogeneous in terms of object classes. This since erroneous pixels will be adjusted. If this step is performed, over segmentation, which would complicate the classification, will be avoided.

For each pixel, look in a $n \times n$ neighborhood and assign the current pixel the group belonging of which most of its neighbors belong to.

So, given a pixels x and y coordinate, loop around its neighbors and let counters describe the number of pixels belonging to each class. Set the current pixels class belonging to the class which has the largest number of members in the current surrounding.

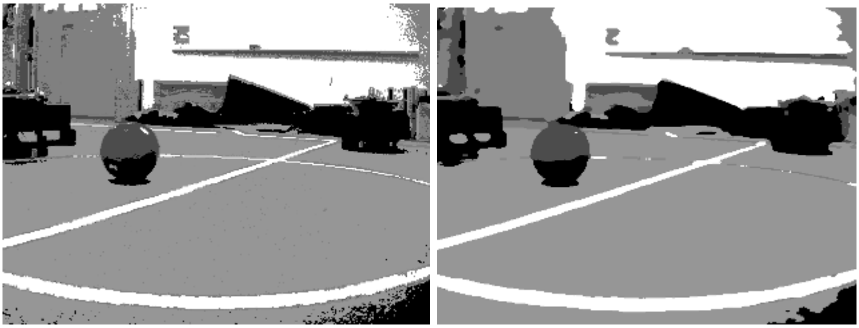


Fig. 3. Pictures from the right camera after multi segmentation and noise reduction

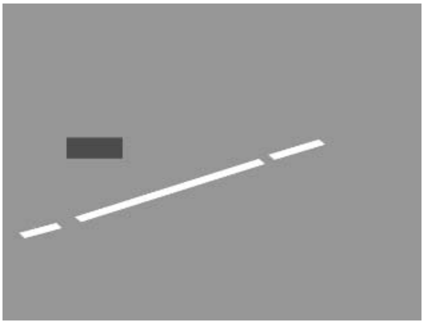


Fig. 4. Final result with found straight lines and ball

3.5 Ball Finder

The Ball Finder is using the output from the noise reduction stage.

When two consecutive red points is found, the min and max x-values are updated and the row is marked as a row with red on it. If red points are found and whose x–values differs more than 50 pixels from previous min and max values they are discarded as noise. When red is found on a row the max y–value is updated, and if red was not found on the previous row the min y–values is updated as well. After the last pixel in the row, the max and min values are checked and if big enough the ball is found. If no red points are found on a row, but was found on the previous row all max and min values are reset. If no ball is found a min–value greater than max is reported.

3.6 Linealyzer – A simple Non-horizontal Straight-Line Finder

When a line segment, a green-white-green edge, has been detected the row, width and center of that segment is stored in a FIFO queue. After a segment is read, check if there is a stored line with a center that differs at most ten pixels, if not, start a new line and save x, row and width of the first line. If the segment is a part of a previous line, update the center of last line and the number of rows that the line consists of.

When eight consecutive rows have been found, compute the slope of that line part. When sixteen consecutive rows have been found compute the slope of the line part and compare it against the slope of the first part. If the slope is within an error marginal of two pixels the line is considered as a straight line. Mark the line as valid and update the data for the endpoint of the line. Compute the slope of every eight–row block and compare it to the first block. If the slope is within the error marginal update the endpoints, otherwise don't update the line anymore.

When the FIFO is empty, the memory is searched for bad lines, i.e. lines that have been started but not found at least sixteen consecutive rows with the same slope. A line is considered bad when starting row plus number of rows is less than the current row in the picture and is not marked as valid.

All lines start– and endpoints are reported at the beginning of the next frame.

3.7 Stereo Matching

Three FIFO queues (FIFO_l, FIFO_r, FIFO_res) and one content addressable memory, CAM are used in the stereo matching.

First detect all feature transitions for both cameras and store the row, x position and colour in FIFO_l and FIFO_r. When all transitions in a row have been detected, the first eight (or as many that has been found) transitions and positions from FIFO_l are read to memory and the transitions are stored in CAM for fast comparisons.

The first transition of FIFO_r is compared to the transitions in the CAM. If a match is found, the corresponding transitions position is loaded and if the difference in x position is within the predetermined range (150 pixels in our case to detect objects that is about 3 dm away) it is considered a good match. If it is a good match, the transition is stored with the row number, disparity and mean x value of the two images and is stored in FIFO_res for further analysis. If it isn't a good match, the next possible match in the CAM is checked in the same

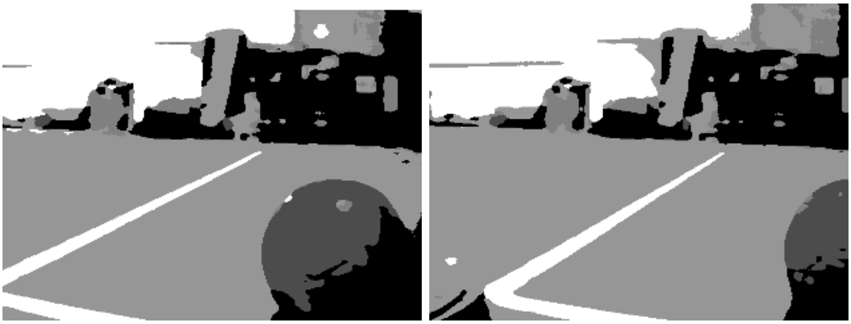


Fig. 5. Pictures from left and right camera for Stereo matching

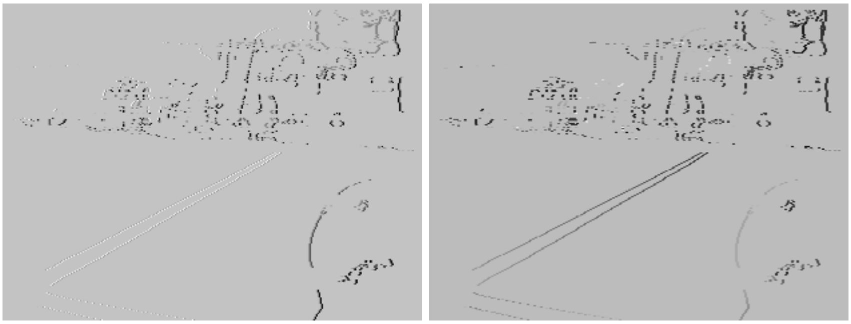


Fig. 6. Left image shows points found in both right and left camera. Right image shows the distance as gray-scale.

way. When a good match is found, the transition at that and lower addresses are shifted out CAM, and the CAM is filled with any transitions that are left at the same row in FIFO_l.

If no good match is found a check for partial matches is started. A partial match could happen when for example, the ball lies close to a line and one camera sees ball-line and the other sees ball-grass-line, or just a mismatch in sensitivity in the cameras.

For a partial match, first the transition from a colour in FIFO_r is searched for and if found, the to colour in FIFO_r is searched. Only if both from and to colours are found any match is considered good. When both from and to colours are found, two transitions is stored in FIFO_res, first the transition that matched the from colour and then the transitions matching the to colour, effectively inferring one extra transition in the original point. If no good match is found the next entry in FIFO_r is analyzed, and if necessary the entries in CAM is updated with new points from FIFO_l.

The pictures in fig (5) are the starting images for the stereo matching. The result is shown in fig (6) and it shows all matched edges to the left and to the right the distances are displayed. White is close and black is far away. The shift to the ball is 72 pixels. The shift to the beginning of the line is 69 pixels and to the end 30 pixels.

There are some errors in the background and in the ball due to the differences in colours and lighting in the pictures

3.8 Harris and Stephens Combined Corner and Edge Detector

For Robocup 2005 a new algorithm has been implemented in the vision system that complement existing functionality. The widely–used Harris and Stephens combined corner and edge detector [14] has been implemented in hardware. It's based on a local autocorrelation function and it performs very well on natural images. The hardware implementation in this paper takes as input RGB-signals from the two synchronized OV7610 cameras in real–time. Pixels whose strength is above an experimental threshold are chosen as visual features. The purpose is to extract the image feature in a sequence of images taken by the two cameras. The obtained data is detected feature–points on each image and can be future analyzed by for example, feature matching algorithms, stereo vision algorithms and visual odometry.

Harris and Stephens combined corner and edge uses small local window w to analyze the image I given by the mathematical formula 4. Note that the small local window only traverses the image with small shifts:

$$E(x, y) = Ax^2 + 2Cxy + By^2 \tag{4}$$

where:

$$A = X^2 \otimes w, B = Y^2 \otimes w, C = (XY) \otimes w \tag{5}$$

where X and Y are approximated by:

$$X = I \otimes \begin{bmatrix} -1\,0\,1 \\ -1\,0\,1 \\ -1\,0\,1 \end{bmatrix} = \partial I/\partial x \tag{6}$$

$$Y = I \otimes \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \partial I/\partial y \tag{7}$$

The response to this function is noisy caused by the rectangle window, so we introduced as Harris and Stephens declared a smooth circular Gaussian window equ (8) instead:

$$w_{u,v} = e^{-(u^2+v^2)/2\sigma^2} \tag{8}$$

With a lot of potential corners detected from the function described above, we then apply the experimental threshold to sort out the true corners from the less distinctive ones. Finally a corner filter has been applied by only storing the most distinctive corners within a 3x3 pixel window sliding over the final feature set. This filter eliminates all corners that are to close to each other and will reduce the amount of data to analyze in the next step.

To implement this algorithm on the 1 million gates FPGA used on the vision system the VHDL language has been used. But to parallelize Harris and Stephens combined corner and edge detector; the system was first implemented in Matlab to test different approaches, local window sizes and thresholds. In Matlab real images from the vision system could be analyzed and also detected corners could be plotted on the analyzed images for comparison.

To gain real-time speed of the system the algorithm was designed as a pipeline, so each step execute in parallel. This means that it takes some cycles for the first two pixels to traverse the entire pipeline, but when the first two pixels has been analyzed the next two pixels will come the immediately at the end of next cycle. So when the system is running two pixels will be analyzed every cycle, note that one cycle is not the same as one tick on the system clock.

Three different window generators are used; the first one is before the calculation with derivative masks, next is before the factorization stage and the last one is before the comparison stage. Each window generator store values incoming from the stage before and will not generate any output until the entire window is full. When the window is full the values within it will be transferred to the next stage and the window will be shifted. The size of the last two window generator gave us a satisfied result and was decided when the algorithm was tested in Matlab. The size of the first window generator was not optional because it must have the same size as the derivative mask which is 3–by–3.

In general we can say that the first real stage of the pipeline is to calculate the pixels within the first small window with the derivative masks described by formula 6 and 7 above. The next step in the pipeline is to factorize (5) and apply the Gaussian filter (8). The stage after that is to calculate formula 4 which will give us a corner strength value. And the last stage is to compare corners with the experimental threshold and filter our corners that are to close to each other on the image.

When testing and design of a parallel version of Harris and Stephens corner and edge detector was complete the system was implemented in VHDL. When each stage of the pipeline was implemented the stage was tested in ModelSim and the result was always compared and verified with Matlab as reference.

When the entire algorithm was completely implemented, the system was tested in reality. To test it an image capture and corner plotting program was implemented and the result can be seen in Figure 7).

From the resulting images we can clearly see that many of the corners that have been found in both images are the same corners. This will facilitate for feature matching algorithms to match pair corners from both images. When corners in both images have been pair, stereo vision algorithm can be applied and visual odometry can be obtained from a sequence of paired features. More research will be done on these subjects in the near future.

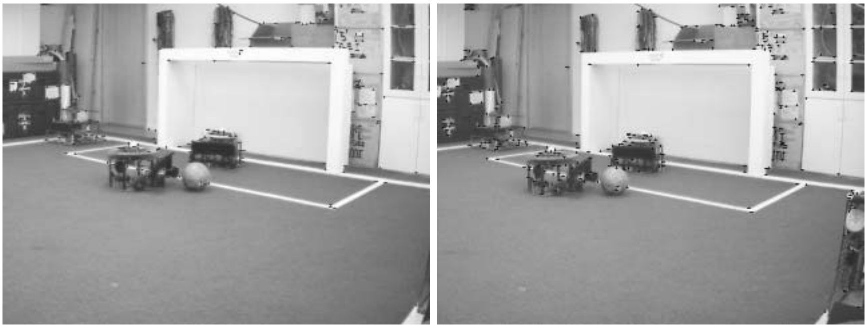


Fig. 7. The corner detection result of one of the first real tests of the system. These two images is a pair out of a sequence of images analyzed by the hardware implemented algorithm.

The frame rate obtained when the system is running was approximately 13.7 frames per second on our 50 MHz FPGA.

4 Results and Discussion

For any kind of robot the sensor system is crucial for its observation of the environment. Of various sensors used in Robocup, vision is the most powerful. The main way vision is implemented today is on ordinary computers. Although a modern PC has very high performance there is always a trade–off between frame–rate and resolution.

The results from this study shows that; by using an FPGA with only 1 million gates, it is possible to achieve a frame–rate of 13Hz on a stereo–camera setup, where all corners are detected in real–time. There are plans of implementing the stereo matching in the FPGA as well as a variant of the ICP–algorithm, which can be used for both localization and movement of the robot. A modified version of the ICP–algorithm (Iterative Closest Point)[15], can also be used to monitor the movements of other robots as well as the ball.

The limitation of frame–rate to 13Hz is due to the number of hardware multipliers, and the clock–frequency of the FPGA. By increasing the number of multipliers the frame–rate can be turned up to the maximal 25Hz, which is then limited by the frame–rate of the cameras.

Acknowledgments

A special thanks to Xilinx Inc., for their support in terms of circuits.

References

[1] K Benkrid, D Crookes, J Smith and A Benkrid, "High Level Programming for Real Time FPGA Based Video Processing", IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM2001, April 2001.

[2] P M Curry, F Morgan and L Kilmartin, "Xilinx FPGA Implementation of an Image Classifier for Object Detection Applications", IEEE Signal Processing Society, International conference on image processing, 2001, pp. 346-349.

[3] D Demigny, L Kessal, R Bourgiba, N Boudouani, "How to Use High Speed Reconfigurable FPGA for Real Time Image Processing", Fifth IEEE International Workshop on Computer Architectures for Machine Perception (CAMP'00), 2000.

[4] B. Draper, W. Najjar, W. Bøhm, J. Hammes, R. Rinker, C. Ross, J. Bins, "Compiling an Optimizing Image Processing Algorithms for FPGAs", IEEE International Workshop on Computer Architecture for Machine Perception (CAMP). Padova, Italy, Sept. 11-13, 2000.

[5] B.A. Draper,W Bøhm, J Hammes, W Najjar, R Beveridge, C Ross, M Chawathe, M Desai and J Bins, "Compiling SA-C Programs to FPGAs: Performance Results", International Conference on Vision Systems, Vancouver, July 7-8, 2001. p. 220-235.

[6] S Herrmann, H Mooshofer, H Dietrich and W Stechele, "A Video Segmentation Algorithm for Hierarchical Object Representation and Its Implementations", IEEE Trans on Circuits and Systems for Video Technology, Vol. 9, No. 8, Dec. 1999.

[7] M Jacomet, J Goette, J Breitenstein and M Hager, "On a Development for Real-Time Information Processing in System-on-Chip Solutions", University of Applied Sciences Berne, Biel School of Engineering and Architecture, Technical Report.

[8] M Leeser, N Kitayeva and J Chrisman, "Spatial and Color Clustering on an FPGA-based Computer System", Proc. SPIE, Vol. 3526, Nov. 1998, pp. 25-33.

[9] W Luk, P Andreou, A Derbyshire, D Dupont-De Dinechin, J Rice, N Shiraz and D Siganos, "A Reconfigurable Engine for Real-Time Video Processing", FPL, Springer Lecture Notes in Computer Science, 1998, pp. 169-178.

[10] E.N. Malamas, E.G.M. Petrakis, M Zervakis, L Petit and J.D. Legat, "A Survey on Industrial Vision Systems", Applications and Tools, Technical Report.

[11] K.J. Palaniswamy, M.E. Rizkalla, A.C. Sihna, M. El-Sharkawy and P Salama, "VHDL Implementation of a 2-D median filter", IEEE, 1999.

[12] L Petit and J.D. Legat, "Hardware Techniques for the Real-Time Implementation of Image Processing Algorithms", Microelectronics Laboratory, University Catholique de Louvian, Belgium.

[13] L Larson, "An EPLD Based Transient Recorder for Simulation of Video Signal Processing Devices in a VHDL Environment Close to System Level Conditions". In Proceedings of the Sixth International Workshop on Field Programmable Logic and Applications, volume 1142, pages 371-375. Darmstadt, Sept. 1996. LCNS

[14] C. Harris and M. Stephens, "A combined corner and edge detector", In M. M. Matthews, editor, Proceedings of the 4th ALVEY vision conference, September 1988, pp. 147-151.

[15] P.J. Besl and N.D. McKay, "A Metod for Registration of 3-D Shapes", IEEE Trans. on Pattern Analysis and Machine Intelligence. vol. 14, no. 2, February 1992, pp. 239-256.

[16] L. Lindh, "FASTHARD - A Fast Time Deterministic Hardware Based Real-Time Kernel", IEEE press, Real-Time Workshop, Athens, January, 1992.

[17] J. Kender, "Saturation, Hue and Normalized Color", Carnegie-Mellon University, Computer Science Dept., Pittsburgh PA, 1976.

Enhancing the Reactivity of the Vision Subsystem in Autonomous Mobile Robots Using Real-Time Techniques*

Paulo Pedreiras[1], Filipe Teixeira[2], Nelson Ferreira[2], Luís Almeida[1],
Armando Pinho[1], and Frederico Santos[3]

[1] LSE-IEETA/DET, Universidade de Aveiro Aveiro, Portugal
{pedreiras, lda, ap}@det.ua.pt
[2] DET, Universidade de Aveiro Aveiro, Portugal
{a23082, a21085}@alunos.det.ua.pt
[3] DEE, Instituto Politécnico de Coimbra, Coimbra, Portugal
fred@mail.isec.pt

Abstract. Interest on using mobile autonomous agents has been growing, recently, due to their capacity to cooperate for diverse purposes, from rescue to demining and security. In many of these applications the environments are inherently unstructured and dynamic, requiring substantial computation resources for gathering enough sensory input data to allow a safe navigation and interaction with the environment. As with humans, who depend heavily on vision for these purposes, mobile robots employ vision frequently as the primary source of input data when operating in such environments. However, vision-based algorithms are seldom developed with reactive and real-time concerns, exhibiting large variations in the execution time and leading to occasional periods of black-out or vacant input data. This paper addresses this problem in the scope of the CAMBADA robotic soccer team developed at the University of Aveiro, Portugal. It presents an evolution from a monolithic to a modular architecture for the vision system that improves its reactivity. With the proposed architecture it is possible to track different objects with different rates without losing any frames.

1 Introduction

Coordinating several autonomous mobile robotic agents in order to achieve a common goal is currently a topic of intense research [11,7]. This problem can be found in many robotic applications, either for military or civil purposes, such as search and rescue in catastrophic situations, demining or maneuvers in contaminated areas. One initiative to promote research in this field is RoboCup [7] where several autonomous robots have to play football in a team to beat the opponent.

As for many real-world applications, robotic soccer players are autonomous, though potentially cooperative, mobile agents that must be able to navigate in and

* This work was partially supported by the European Community through the ARTIST2 NoE (IST-004527) and by the Portuguese Government through the project FCT-POSI/ROBO/43908/2002, also partially funded by FEDER.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 371 – 383, 2006.

interact with their environment. Some of these actions exhibit real-time characteristics, although with different levels of criticality. For instance, the capability to timely detect obstacles in the vicinity of the robot can be regarded as a hard activity since failures, either in the temporal or value domains, can result in injured people or damaged equipment. On the other hand, activities like self-localization or tracking the ball, although important for the robot performance, are inherently soft since failing in these activities simply causes performance degradation. The capability to timely perform the required image-processing activities at rates high enough to allow visual-guided control or decision-making is called real-time computer vision (RTCV) [13].

The RoboCup soccer playfield resembles human soccer playfields, though with some (passive) elements specifically devoted to facilitate the robots navigation. In particular the goals have solid and distinct colors and color-keyed posts are placed in each field corner. This type of environment can be classified as a *passive information space* [2]. Within an environment exhibiting such characteristics, robotic agents are constrained to rely heavily on visual information to carry out most of the necessary activities, leading to a framework in which the vision subsystem becomes an integral part of the close-loop control. In these circumstances the temporal properties of the image-processing activities (e.g. period, jitter and latency) strongly impact the overall system performance. Therefore, the application of real-time techniques to these activities so as to improve their temporal behavior by reducing mutual interference and limiting processing demand seems adequate to improve global performance.

In this paper we propose a new modular architecture for the vision subsystem, where different objects are tracked by independent processes. Using appropriate operating system services, these processes are then scheduled according to their relative importance, with preemption. The result is a noteworthy improvement of the temporal behavior of the processes deemed to have greater impact on the overall system performance.

The remainder of the paper is structured as follows: Section 2 presents the generic computing architecture of the CAMBADA robots. Section 3 shortly describes the working-principles of the vision-based modules and their initial implementation in the CAMABADA robots. Section 4 describes the new modular architecture that has been devised to enhance the temporal behavior of the image-processing activities. Section 5 presents experimental results and assesses the benefits of the new architecture. Finally, Section 6 concludes the paper.

2 The CAMBADA Computing Architecture

The computing architecture of the robotic agents follows the biomorphic paradigm [9], being centered on a main processing unit (*the brain*) that is responsible for the higher-level behavior coordination (Fig. 1). This main processing unit handles external communication with other agents and has high bandwidth sensors (*the vision*) directly attached to it. Finally, this unit receives low bandwidth sensing information and sends actuating commands to control the robot attitude by means of a distributed low-level sensing/actuating system (*the nervous system*).

The main processing unit is currently implemented on PC-based computers that deliver enough raw computing power and offer standard interfaces to connect the other systems, namely USB. The PCs run the Linux operating system over the RTAI (Real-Time Applications Interface [5]) kernel, which provides time-related services, namely periodic activation of processes, time-stamping and temporal synchronization.

The agents software architecture is developed around the concept of a real-time database (RTDB), i.e., a distributed entity that contains local images (with local access) of both local and remote time-sensitive objects with the associated temporal validity status. The local images of remote objects are automatically updated by an adaptive TDMA transmission control protocol [3] based on IEEE 802.11b that reduces the probability of transmission collisions between team mates thus reducing the communication latency.

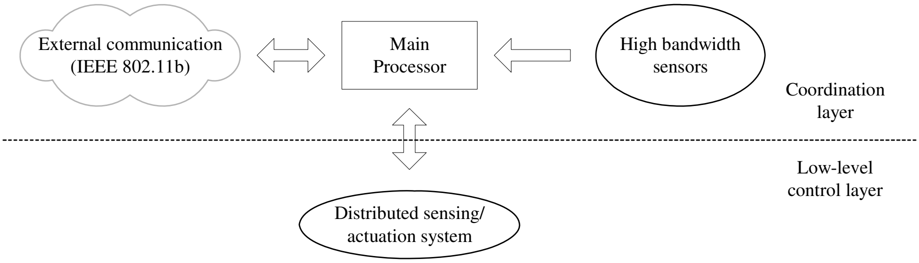


Fig. 1. The biomorphic architecture of the CAMBADA robotic agents

The low-level sensing/actuating system follows the fine-grain distributed model [8] where most of the elementary functions, e.g. basic reactive behaviors and closed-loop control of complex actuators, are encapsulated in small microcontroller-based nodes, interconnected by means of a network. This architecture, which is typical for example in the automotive industry, favors important properties such as scalability, to allow the future addition of nodes with new functionalities, composability, to allow building a complex system by putting together well defined subsystems, and dependability, by using nodes to ease the definition of error-containment regions. This architecture relies strongly on the network, which must support real-time communication. For this purpose, it is used the CAN (Controller Area Network) protocol is used [1], which has a deterministic medium access control, a good bandwidth efficiency with small packets and a high resilience to external interferences. Currently, the interconnection between CAN and the PC is carried out by means of a gateway, either through a serial port operating at 115Kbaud or through a serial-to-USB adapter.

3 The CAMBADA Vision Subsystem

The CAMBADA robots sense the world essentially using two low-cost webcam-type cameras, one facing forward, and the other pointing the floor, both equipped with wide-angular lenses (approximately 106 degrees) and installed at approximately 80cm above the floor. Both cameras are set to deliver 320x240 YUV images at a rate of 20 frames per second. They may also be configured to deliver higher resolution video frames (640x480), but at a slower rate (typically 10-15 fps). The possible combinations between resolution and frame-rate are restricted by the transfer rate allowed by the PC USB interface.

The camera that faces forward is used to track the ball at medium and far distances, as well as the goals, corner posts and obstacles (e.g. other robots). The other camera, which is pointing the floor, serves the purpose of local omni-directional vision and is used for mainly for detecting close obstacles, field lines and the ball when it is in the vicinity of the robot. Roughly, this omni-directional vision has a range of about one meter around the robot.

All the objects of interest are detected using simple color-based analysis, applied in a color space obtained from the YUV space by computing phases and modules in the UV plane. We call this color space the YMP space, where the Y component is the same as in YUV, the M component is the module and the P component is the phase in the UV plane. Each object (e.g., the ball, the blue goal, etc.) is searched independently of the other objects. If known, the last position of the object is used as the starting point for its search. If not known, the center of the frame is used. The objects are found using region-growing techniques. Basically, two queues of pixels are maintained, one used for candidate pixels, the other used for expanding the object. Several validations can be associated to each object, such as minimum and maximum sizes, surrounding colors, etc.

Two different Linux processes, Frontvision and Omnivision, handle the image frames associated with each camera. These processes are very similar except for the specific objects that are tracked. Figure 2 illustrates the actions carried out by the

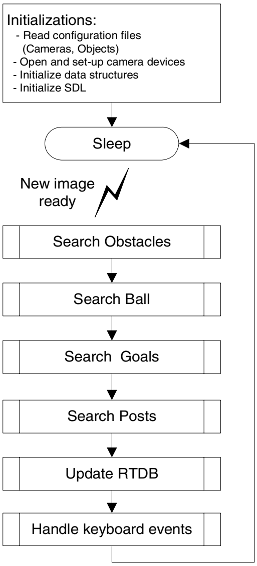


Fig. 2. Flowchart of the Frontvision process

Frontvision process. Upon system start-up, the process reads the configuration files from disk to collect data regarding the camera configuration (e.g. white balance, frames-per-second, resolution) as well as object characterization (e.g. color, size, validation method). This information is then used to initialize the camera and other data structures, including buffer memory. Afterwards the process enters in the processing loop. Each new image is sequentially scanned for the presence of the ball, obstacles, goals and posts. At the end of the loop, information regarding the diverse objects is placed in a real-time database.

Keyboard, mouse and the video framebuffer are accessed via the Simple DirectMedia Layer library (SDL, [12]). At the end of each loop the keyboard is pooled for the presence of events, which allows e.g. to quit or dynamically change some operational parameters.

4 A Modular Architecture for Image Processing: Why and How

As referred to in the previous sections, the CAMBADA robotic soccer players operate in a dynamic and passive information space, depending mostly on visual information to perceive and interact with the environment. However, gathering information from such type of environments is an extremely processing-demanding activity [4], with hard to predict execution times. Regarding the algorithms described in Section 3, above, it could be intuitively expected to observe a considerable variance in process execution times since in some cases the objects may be found almost immediately, when their position between successive images does not change significantly, or it may be necessary to explore the whole image and expand a substantial amount of regions of interest, e.g. when the object disappears from the robot field of vision. This expectation is in fact confirmed in reality, as depicted in Figure 3, which presents a histogram of the execution time of the ball tracking alone. Frequently the ball is located almost immediately, taking less than 2.5ms to complete. However, a significant amount of instances require between 17.5ms and 32.5ms to complete and, sometimes, the process requires over 50ms, which is the inter-frame period used by the cameras.

As described in Section 3, the CAMBADA vision subsystem architecture is monolithic with respect to each camera, with all the image-processing carried out

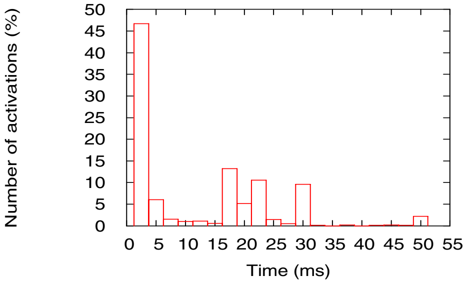


Fig. 3. Ball tracking execution time histogram captured at run-time

within two processes, the Frontvision and the Omnivision, respectively. Each of these processes tracks several objects sequentially. Thus, the following frame is acquired and analysed only after tracking all objects, which may take, in the worst case, hundreds of milliseconds, causing a certain number of consecutive frames to be skipped. These are vacant samples for the robot controllers that degrade the respective performance and, worse, correspond to black-out periods in which the robot does not react to the environment. Considering that, as discussed in Section 1, some activities, like obstacle detection, have hard deadlines this situation becomes clearly unacceptable. Increasing the available processing power could, to some extent, alleviate the situation (although not completely solve it). However, the robots are autonomous and operate from batteries, and thus energy consumption aspects are highly relevant, which renders brut-force approaches undesirable.

4.1 Using Real-Time Techniques to Manage the Image Processing

As remarked is Section 1, some of the activities carried out by the robots exhibit real-time characteristics with different levels of criticality, importance and dynamics. For example, the latency of obstacle detection limits the robots maximum speed in order to avoid collisions with people, walls or other robots. Thus, the obstacle detection process should be executed as soon as possible, in every image frame, to allow the robot to move as fast as possible in a safe way. On the other hand, detecting the corner poles for localization is less demanding and can span across several frames. However, this activity should not block the more frequent obstacle detection. This calls for the encapsulation of each object tracking in different processes as well as for the use of preemption and appropriate scheduling policies, giving higher priority to most stringent processes. These are basically the techniques that were applied to the CAMBADA vision subsystem as described in the following section.

4.2 A Modular Software Architecture

Figure 4 describes the software modular architecture adopted for the CAMBADA vision subsystem, for each of the two cameras used. Standard Linux services are used to implement priority scheduling, preemption and data sharing.

Each camera uses one process (*ReadXC*) to create a shared memory region where the images are buffered. The processes are periodically triggered by the cameras, whenever a new image frame is available. Each object tracking process (e.g. obstacle,

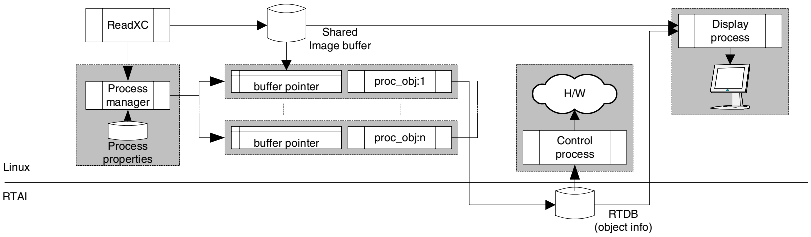


Fig. 4. Modular software architecture for the CAMBADA vision subsystem

ball), generically designated by proc_obj:x, x={1,2,…n}, is triggered by a process manager module, according to the relevant process attributes (period, phase) stored in a process database. Once started, each process gets a link to the most recent image frame available and starts tracking the respective object. Once finished, the resulting information (e.g. object detected or not, position, degree of confidence, etc.) is placed in a real-time database [6] (Object info), similarly located in a shared memory region. This database may be accessed by any other processes on the system, e.g. to carry out control actions. A display process may also be executed, which is useful mainly for debugging purposes.

Scheduling of vision related processes relies on the real-time features of the Linux kernel, namely the FIFO scheduler and priorities in the range 15-50. At this level Linux executes each process to completion, unless the process blocks or is preempted by other process with higher real-time priority. This ensures that the processes are executed strictly according to their priority (i.e., importance) with full preemption, relying solely on operating system services, in a non intrusive and transparent way. The real-time features of Linux, despite limited, are sufficient at this time-scale (periods multiple of 50ms) as long as memory swapping and disk access are avoided.

The buffer management system keeps track of the number of processes that are connected to each buffer. Buffers may be updated only when there are no processes attached to them, thus ensuring that processes have consistent data independently of the time required to complete the image analysis. This approach is adequate to situations where different objects have different degrees of dynamism, e.g., the ball is highly dynamic and needs being tracked in every frame but the relative goal position is less dynamic and can be tracked every four frames. Moreover, in the latter case prediction methods can be effectively used, as suggested in [10], to allow obtaining estimates of object positions based on past data.

The process activation is carried out by a process manager that keeps, in a database, the process properties, e.g. priority, period and phase. For each activation, the process manager scans the database, identifies which processes should be activated and sends them appropriate signals. The cameras are set to grab images periodically, and the arrival of each new image frame is used to activate the process manager. This framework allows reducing the image processing latency, since processes are activated immediately upon the arrival of new images. Another relevant feature that is supported is the possibility of de-phasing the process activations in the time domain, to minimize mutual interference and thus reducing their response time.

5 Experimental Results

In order to assess the benefits of the modular approach with respect to the initial monolithic one, several experiments were conducted, using a PC with an Intel Pentium III CPU, running at 833MHz, with 256MB of RAM. The PC runs a Linux 2.4.21 kernel, patched with RTAI 24.1 for the experiments involving the real-time architecture. The image-capture devices are Logitech Quickcams, with a Philips chipset. The cameras were set-up to produce 320*240 images at a rate of 20 frames-per-second. The time instants were measured accessing the Pentium TSC.

5.1 Monolithic Architecture

The code of the Frontvision and Omnivision processes (Section 3) was instrumented to measure the start and finishing instants of each instance. Figure 5 presents a histogram of the inter-activation intervals of these processes, while Table 1 presents a summary of some relevant statistical figures regarding their response-time.

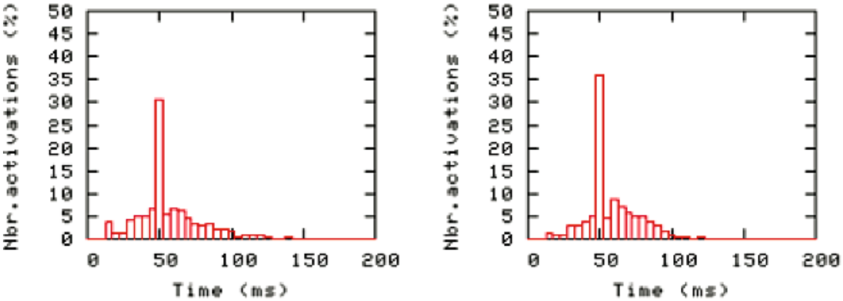


Fig. 5. Activation interval for Frontvision (left) and Omnivision (right)

The response time of both processes exhibits a substantial variance, with execution times ranging from 10ms to over 600ms and an average execution time around 44ms. Noting that the processing of a frame can only start after the previous one being fully processed, this response-time variance leads to the activation time dispersion observed in Figure 5.

Table 1. Frontvision and Omnivision response–time statistical figures

Process	Max (ms)	Min (ms)	Average (ms)	Stand. deviation
Frontvision	427.7	9.9	43.6	30.6
Omnivision	635.5	10.8	44.0	30.4

The combination of such an irregular activation pattern with highly variant response-times results in a high jitter value and number of skipped frames. Remembering that the image-processing is part of the robot control loop, this leads to an irregular and poor robot performance.

5.2 Modular Architecture

The different image-processing activities have been separated and wrapped in different Linux processes, as described in Section 4. The periods, offsets and priorities assigned to each one of the processes are summarized in Table 2.

The obstacle avoidance processes are the most critical ones since they are responsible for alerting the control software of the presence of any obstacles in the vicinity of the robot, allowing it to take appropriate measures (e.g. evasive maneuvers or immobilization). Therefore these processes are triggered at a rate equal to the camera frame rate and receive the highest priority, ensuring a response-time as short as possible. It should be remarked that these processes scan delimited image regions, only, looking for specific features, thus their execution time is bounded and relatively short. In the experiments the measured execution time upper bounds were 7ms and 9ms for *Avoid_Om* and *Avoid_Fr*, respectively. Therefore, this architecture allows ensuring that every frame will be scanned for the presence of obstacles.

Table 2. Image-processing processes properties

Process	Period (ms)	Priority	Offset (ms)	Purpose
Ball_Fr	50	35	0	Ball tracking (front camera)
BGoal / YGoal	200	25	50/150	Blue / Yellow Goal tracking
BPost / YPost	800	15	100/200	Blue / Yellow Post tracking
Avoid_Fr	50	45	0	Obstacle avoidance (front cam.)
Ball_Om	50	40	0	Ball tracking (omni camera)
Avoid_Om	50	45	0	Obstacle avoidance (omni camera)
Line	400	20	0	Line tracking and identification
KGoal	100	30	0	Goal line tracking

The second level of priority is granted to the *Ball_Om* process, which tracks the ball in the omni-directional camera. This information is used when approaching, dribbling and kicking the ball, activities that require a low latency and high update rate to be successful. Therefore this process should, if possible, be executed on every image frame, thus its period was also set to 50ms.

The third level of priority is assigned to the *Ball_Fr* process, responsible for locating the ball in the front camera. This information is used mainly to approach the ball when it is at medium to far distance from the robot. Being able to approach the ball quickly and smoothly is important for the robot performance but this process is more delay tolerant than the *Ball_Om* process, thus it is assigned a lower priority.

The *KGoal* process detects the goal lines, being mainly used by the goal keeper. Contrarily to the previous processes, the dynamics of the lines depend only on the robot movement. Furthermore, the localization of the robot within small regions is complemented with an odometry subsystem, which updates the robot position. This allows having a lower activation rate and priority of the vision-based goal lines detection without incurring in significant performance degradation.

A similar reasoning was applied to decide the attributes of the *BGoal* and *YGoal* processes, which track the position of the blue and yellow goals, respectively. Since the goals are stationary with respect to the play fields, and due to the availability of odometry data, updates can be made more sparsely and are not particularly sensitive to jitter. For this reason these processes were assigned a priority of 25 and a period of 200ms (every 4 frames).

The field line detection process (*Line*) detects and classifies the lines that delimit the play field, pointing specific places in it. This information is used only for calibration of the localization information and thus may be run sparsely (400ms). Post detection processes (*BPost* and *YPost*) have a similar purpose. However, since the information extracted from them is coarser than from the line detection (i.e., it is affected by a bigger uncertainty degree), it may be run at even a lower rate (800ms) without a relevant performance degradation.

The offsets of the different processes have been set-up to de-phase the process activations as much as possible. With the offsets presented in Table 2, besides the obstacle and ball detection processes, which are executed for every frame, no more than two other processes are triggered simultaneously. This allows minimizing mutual interference and thus reducing the response-time of lower priority processes.

Figures 6, 7 and 8 show the inter-activation intervals of selected processes (obstacle, ball, goal line and yellow post tracking), which clearly illustrate the differences between the modular and the monolithic architectures regarding the processes temporal behavior. The processes that receive higher priority (obstacle detection, Fig. 6) exhibit a narrow inter-activation variance, since they are not blocked and preempt other processes that may be running. Figure 7 regards the inter-activation intervals of the ball tracking processes. As stated above, the ball tracking process on

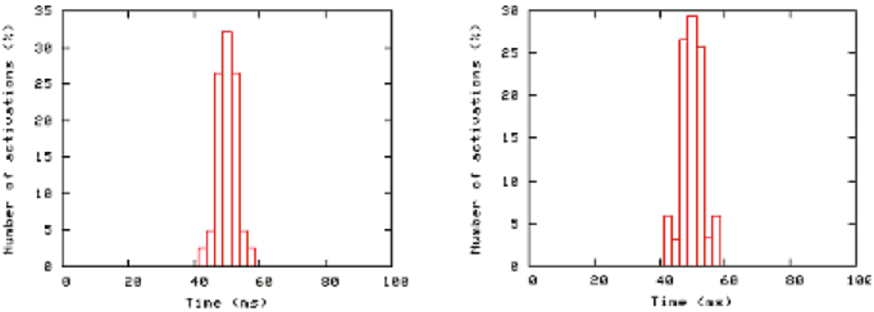


Fig. 6. Front (left) and omni-directional (right) obstacle detection processes inter-activation intervals

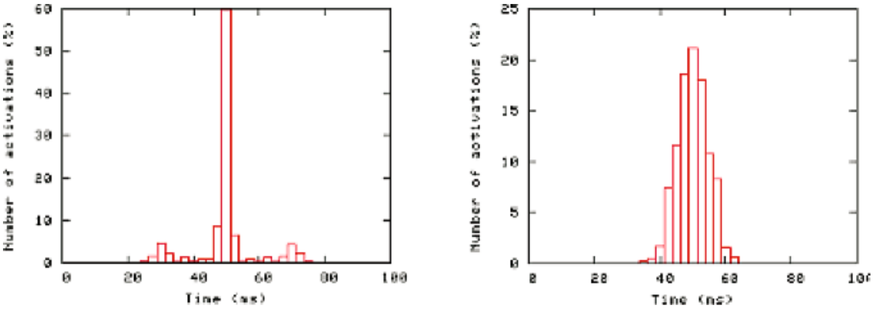


Fig. 7. Front (left) and omni-directional (right) ball tracking processes inter-activation intervals

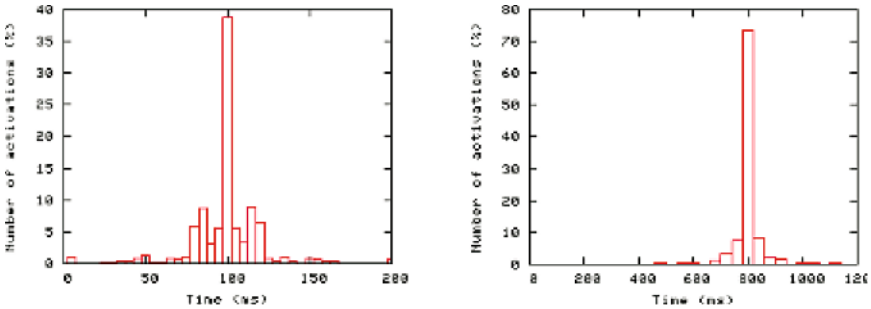


Fig. 8. Goal line (left) and yellow post (right) tracking processes inter-activation intervals

the omni-directional camera received higher priority since its data is used by more time sensitive activities. For this reason its inter-activation interval is narrower than the ball tracking process related to the front camera. Ball-tracking processes exhibit a significantly higher response-time jitter than obstacle detection because, in the worst case, they must scan the whole image. For this reason the lower-priority ball-tracking process exhibits a much higher inter-activation jitter than the higher-priority one. The same behavior is observed for the remaining processes, which see their inter-activation jitter increase as their relative priorities diminish.

Table 3 shows statistical data regarding the inter-activation intervals of these processes, which confirm, in a more rigorous way, the behavior observed above. The processes are sorted by decreasing priorities exhibiting, from top to bottom, a steady increase in the gap between maximum and minimum values observed as well as in the standard deviation. This is expected since higher priority processes, if necessary, preempt lower priority ones increasing their response-time.

Comparing the data in Tables 1 and 3, a major improvement can be observed with respect to the activation jitter of the most time-sensitive processes, which was reduced from around 30ms to 2ms-3ms (object avoidance) and 3ms-9ms (ball tracking). Furthermore, since the obstacle avoidance processes have a relatively short execution

Table 3. Modular architecture: statistical data of process inter-activation intervals

Process	Max (ms)	Min (ms)	Standard deviation (ms)
Avoid_Om	56.9	43.0	1.9
Avoid_Fr	58.1	41.9	2.5
Ball_Om	63.6	35.8	4.7
Ball_Fr	75.9	25.4	8.9
KGoal	506.6	0.5	27.4
Bgoal	547.9	0.6	39.1
YGoal	654.6	0.6	42.0
Line	711.0	38.8	53.3
BPost	1159.7	541.1	60.8
YPost	1101.0	485.6	53.3

time (up to 8.5ms in this architecture) it becomes possible to guarantee their execution in every image frame allowing the robots to run at higher speed without compromising safety.

6 Conclusion

Computer vision applied to guidance of autonomous robots has been generating large interest in the research community as a natural and rich way to sense the environment and extract the necessary features. However, due to the robots motion, vision-based sensing becomes a real-time activity that must meet deadlines in order to support adequate control performance and avoid collisions. Unfortunately, most vision-based systems do not rely on real-time techniques and exhibit very poor temporal behavior, with large variations in execution time that may lead to control performance degradation and even sensing black-out periods (skipped image frames).

In this paper, the referred problem is identified in the scope of the CAMBADA middle-size robotic soccer team, being developed at the University of Aveiro, Portugal. Then, a new architectural solution for the vision subsystem is presented that substantially improves its reactivity, reducing jitter and frame skipping.

The proposed architecture separates the vision-based object-tracking activities in several independent processes. This separation allows, transparently and relying solely on operative system services, to avoid the blocking of higher priority processes by lower priority ones as well as to set independent activation rates, related with the dynamics of the objects being tracked, together with offsets that de-phase the activation instants of the processes to further reduce mutual interference.

As a consequence, it became possible to guarantee the execution of critical activities (e.g., obstacle avoidance) and privilege the execution of others that, although not critical, have greater impact on the robot performance (e.g., ball tracking). This result and approach are relevant for a class of robots in which the vision subsystem is part of their control loop, leading to a better control performance.

References

1. Controller Area Network - CAN2.0 (1992). Technical Specification, Robert Bosch, 1992.
2. J. Gibson (1979), “The Ecological Approach to Visual Perception”, Houghton Mifflin, Boston, MA, 1979.
3. F. Santos, L. Almeida, P. Pedreiras, L.S.Lopes, T. Facchinnetti (2004), "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication Among Mobile Computing Agents". WACERTS 2004, Workshop on Architectures for Cooperative Embedded Real-Time Systems (satellite of RTSS 2004). Lisboa, Portugal, 5-8 Dec. 2004.
4. Guilherme N. DeSouza and Avinash C. Kak (2004) "A Subsumptive, Hierarchical, and Distributed Vision-Based Architecture for Smart Robotics," IEEE Transactions on Systems, Man, and Cybernetics -- Part B: Cybernetics, Vol. 34, pp. 1988-2002, October 2004.
5. RTAI for Linux, available at http://www.aero.polimi.it/~rtai/

6. L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, L.S.Lopes (2004). "Coordinating distributed autonomous agents with a real-time database: The CAMBADA project". ISCIS'04, 19th International Symposium on Computer and Information Sciences. 27-29 October 2004, Kemer - Antalya, Turkey.
7. K. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa (1995) "RoboCup: The Robot World Cup Initiative", *Proc. of IJCAI-95 Workshop on Entertainment and AI/Alife,* Montreal.
8. H. Kopetz (1997). "Real-Time Systems Design Principles for Distributed Embedded Applications", Kluwer.
9. Proc. of the NASA Workshop on Biomorphic Robotics, (2000), Jet Propulsion Laboratory, California Institute of Technology. USA.
10. G. Iannizzotto, F. La Rosa, L. Lo Bello (2004). "Real-time issues in vision-based Human-Computer Interaction". Technical Report, VisiLab, University of Messina , Italy.
11. G. Weiss (2000), "Multiagent systems. A Modern Approach to Distributed Artificial Intelligence" MIT Press, 2000.
12. Simple DirectMedia Layer, available at http://www.libsdl.org/index.php
13. Andrew Blake, R. Curwen, A. Zisserman (1993), "A framework for spatio-temporal control in the tracking of visual contours". Int. Journal of Computer Vision, 11, 2, 127—145, 1993.

Illumination Independent Object Recognition

Nathan Lovell

School of CIT, Griffith University, Nathan 4111 QLD, Australia

Abstract. Object recognition under uncontrolled illumination conditions remains one of hardest problems in machine vision. Under known lighting parameters, it is a simple task to calculate a transformation that maps sensed values to the expected colors in objects (and minimize the problems of reflections and/or texture). However, RoboCup aims to develop vision systems for natural lighting conditions in which the conditions are not only unknown but also dynamic. This makes fixed color-based image segmentation infeasible. We present a method for color determination under varying illumination conditions that succeeds in tracking the objects of interest in the RoboCup legged league.

1 Introduction

Vision systems that perform colour based object recognition (as in the SONY Aibo Four-Legged RoboCup League) rely heavily on a static lighting environment where objects of interest have known colors (for example, the ball is orange). This simplifies the vision problem significantly and has resulted in standard vision modules consisting of three steps [1]:

Color segmentation: Each pixel in the image is classified as being either one of the important colours or an unknown colour.
Blob formation: Identify groups of same coloured pixels (blobs).
Object recognition: Blobs are analysed to determine objects.

Of course, if pixels are incorrectly classified, then the blobs inaccurately reflect the features of the object resulting in identification error. Even in known and static lighting environments, specular reflection can make (for example) white pixels look pink and orange pixels look yellow. The Aibo camera is very unsophisticated compared to other digital cameras that have very complex mechanisms for dealing with changing lighting conditions including variable shutter speeds and automatic focusing. Work in other leagues shows how these features can be used effectively [2]. The Aibo camera, however, lacks many of these advantages and it provides only the most simple of lighting modes that must be manually set. It does not even provide a consistent colour sampling across single images [3].

As we move toward using natural lighting conditions, with unknown and dynamic lighting temperature and intensity, and including shadows and indirect and reflected light, the problems are compounded significantly. If the lighting conditions are known, then the change in perceived colours for different lighting conditions is well modelled by a computable mathematical transformation [3].

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 384–395, 2006.

In unknown lighting conditions the problem of correctly classifying a pixel is very difficult on a pixel by pixel basis.

RoboCup has a strong interest in uncontrolled illumination due to its 2050 target. There has been a stream of papers on lighting conditions over the 2003 and 2004 symposiums [4, 5, 6, 7, 8, 9, 10, 11, 12], however, the implementations in the actual robots has been disappointing. Consider as an illustration the technical challenge regarding variable lighting conditions that has been present in the 4-legged league for some years now. All 24 teams attempted a solution to this problem in 2004. However, the result demonstrated not only the difficulty of the problem but also that current research is far from a solution. While most teams did well in the period when the lights were close to normal levels, as soon as the lights were dimmer no team was able to correctly identify both the ball and the goal, let alone score a goal. Research efforts for uncontrolled illumination for RoboCup have also been reflected elsewhere [13, 14, 15, 16, 17, 18]. Still the results have not convincingly produced a working system.

The main problem with previous approaches has been the concern with creating pixel classifiers for color segmentation of high accuracy and high coverage. We maintain that under changing illumination conditions, this is in essence an impossible task and pixel classification must be determined in the context of the image and, at least partly, by what colour we *expect* that pixel to be. The human eye does this unconsciously. For example, tree leaves at sunset still appear green though in a digital picture very few of the pixels that make up the leaves will be in any way green.

We introduce an efficient object recognition system that is robust to changes in colour intensity and temperature because it does not rely solely on colour classification in order to form blobs. We propose a classifier that is very accurate but only on pixels that are at the core of each color class and essentially refuses to make a decision for most other pixels labeling them as unknown. Our approach works while providing a color label for less than 10% of the pixels in each frame. Instead of using the colour of each pixel to form blobs, we first form the blobs and then use the colours of each pixel within it to tell us the colour of the blob. We can now use the context of the entire image rather than a simple pixel-by-pixel analysis in our object recognition process. Our process then is:

Edge detection: Detect efficiently the edges within the image.

Sparse classification: Classify each pixel that is not determined to be an edge according to the sparse classifier.

Object recognition: Detect the blobs within the image and use the classified pixels within them to determine colour.

We now show details that make each step very efficient. Thus, the entire process has insignificant overhead compared to previous object recognition systems.

There have been other attempts at combining edge detection and colour segmentation to improve object recognition [19] however these have been done with the aim of improving vision system performance and accuracy, not with the aim of overcoming problems associated with variable illumination.

Table 1. Comparison of our edge detection algorithm with well-known Sobel edge detection. The average runtime for the algorithm improves more than 75% for a window size of five. The pixels labeled incorrectly are less than 7% overall. The times in the table are measured in AVW2 profile clicks. AVW2 is our vision testbed program [20].

	Min Runtime	Max Runtime	Avg Runtime	False Positives	False Negatives	Total Incorrect
Sobel	75	129	84.2	-	-	-
Our Algorithm	14	21	20.9	4.1%	2.3%	6.4%

2 Our Fast Edge Detection

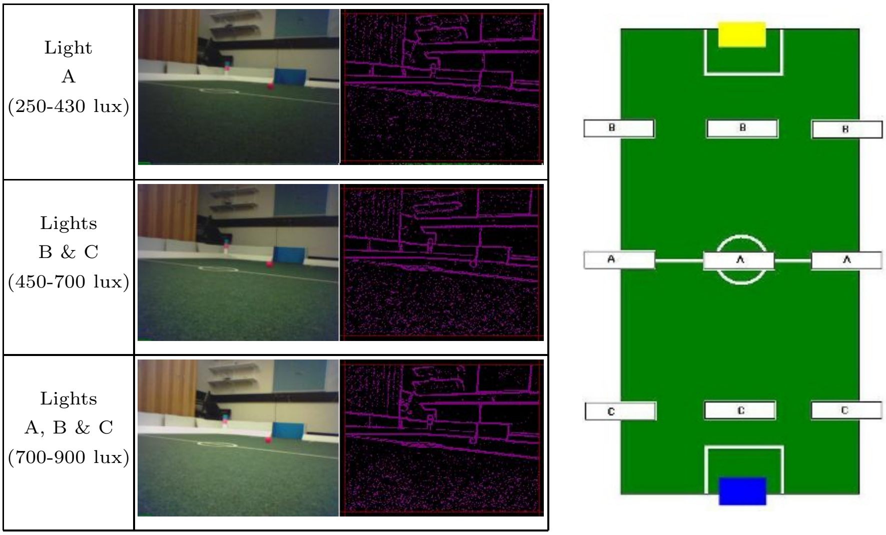
While we are confident that a robust edge detection algorithm would be sufficient for the first step in our process, it must also be very fast. We found that the standard edge detection routines (such as Sobel, Robert's Cross and Canny [21]) are simply too slow to be useful in our real-time processing environment. Therefore, we introduce here a new edge detection routine that executes much faster than traditional edge detection algorithms.

Edge detection routines slide a regular "window" over pixels and determine if the colour between the pixels at the centre of the window varies significantly from the average colour difference over such window. There are variations on this, e.g., Sobel tests hypothesis of edges that could exist within the window, checking each of them.

Our edge detection uses a similar idea however we only consider pixels in the row and column of the centre, thus the window we use is not a square but a cross. The difference in pixels at the centre of the window is computed and compared to the average difference between other pixels in both the row and column respectively. If the difference is much higher (either compared to the horizontal difference, or the vertical one), then that this pixel lies on an edge.

Although both our routine and the traditional edge detection algorithms are linear time under Big-O notation[1], there is a big difference in the order of the hidden cost. In traditional edge detection algorithms, each pixel in the window must be compared to each of its neighbors. For a window size w the constant in such algorithms is $2w(w-1) = O(w^2)$. Thus, quadratic on w, and usually very large. For example, a window size of five leads to 40 colour comparisons for each window evaluation and therefore 40 colour comparisons per pixel in the image. Contrast this to our technique where, with size w, the constant is $2w = O(w)$, i.e. linear in w. This means that in the above example, our algorithm will only do 10 colour comparisons per pixel, thus running in a quarter of the time of a traditional edge detection algorithm. There is a very minor trade off in terms of the algorithm quality. Table 1 compares the runtime cost of our algorithm with

[1] Big-O notation expresses the number of operations an algorithm performs, $t(n)$, as a function of the number, n, of input data items. An algorithm is $O(f(n))$ if there is $c > 0$ and $n_1 > 0$ such that $t(n) < cf(n), \forall n > n_1$. Thus, $t(n)$ is $O(ct(n)), \forall c > 0$ and the constant cost, c, is hidden.



(a) (b)

Fig. 1. (a) Our edge detection routines are very robust to varying illumination conditions. Edge detection remains of high quality. (b) Our experimental setting. Three rows of independently controllable fluorescent lights.

that of a traditional algorithm (Sobel's) a set of 100 images with a (reasonable) window size of five. It also compares the quality of our algorithm against Sobel's by comparing the output of the two algorithms and measuring the percentage of false positives and false negatives.

Despite the use of a smaller window, our edge detection algorithm is very robust (less than 7% error) and remains a solid foundation for the rest of the process as it is particularly resistant to changing lighting conditions. Fig. 1 (a) illustrates the stability. Under the changing illumination conditions of each subsequent image the edge analysis remains remarkably stable. The lighting conditions are described in Fig. 1 (b). Our lab lighting consists of three rows of independently controlled fluorescent lights. The row of lights labeled A runs across the middle of the field. Rows B and C are positioned at the yellow and blue ends of the field respectively. When only row A is on, the field has an inconsistent illumination ranging from 250 to 430 lux. With rows B and C there is a slightly better illumination ranging from 450 to 700 lux. RoboCup conditions are approached only with all three rows of lights on (700 to 900 lux).

Any choice of edge detection algorithm must be implemented carefully to make it feasible to run in real-time on an Aibo. We emphasize an important optimisation that must be performed to make even our improved algorithm suitable. As the window slides over the image, we store and re-use color differences that will occur in comparisons more than once (in fact, proportional to the window size). Consider the pixels at $(10, 10)$ and $(11, 10)$. The difference in colour between these two pixels is calculated for the first time when the window (of size five) is centred on $(5, 10)$ and used in the average to contrast with the difference between pixels at $(5, 10)$ and $(6, 10)$. As the window moves along the tenth row, this difference must be used 10 times. The cost of each comparison is expensive because it involves a three dimensional distance calculation (with a square root). We minimise this cost by keeping a circular buffer of the calculated differences between both the last five pixels and, looking ahead, the next five. Since we iterate across rows first, and then columns, it is also necessary to maintain a circular buffer for each column in the image, in addition to the one for the current row. By using this technique the difference between each two pixels is only calculated once per pixel.

3 Our Sparse Classification

We overcome the problems associated with illumination independence by shifting the focus from colour-based segmentation to edge detection. Colour segmentation in the legged league consists of labeling with one of the environment's colour classes each pixel in the image. We represent this as a colour-segmentation function $c\_class : Y \times U \times V \rightarrow C$ that given a triplet (y, u, v) produces the colour class. For the last few years, as many as 10 colours are used for objects of interest in the field (including, the field itself, beacons, goals, opponents, teammates, markings and of course the ball). A representation of the function *c_class* as a complete colour-lookup table that maps every possible YUV triplet to a particular colour class would be extremely large (2^{24} bits). RoboCup researchers have used many techniques to learn the *c_class* function and to represent it. Among these, machine learning and statistical discrimination techniques like linear discriminant, decision trees [22], Artificial Neural Networks [23], Support Vector Machines [24] and instance-based classifiers (k-nearest neighbors and other non-parametric statistics [22]) have been used. However, the state of the art (in terms of accuracy vs speed) for this environment remains a look-up table of characteristic vectors for each of the 3 dimensions [1]. Look-up tables are faster than machine learning artifacts by several orders of magnitude [25].

While all of these methods can learn and represent the 2^{24} bit colour table off-line (colour calibration), they have trouble adapting smoothly and reliably on-line. The problem is that a change in illumination conditions implies a change in the colour table in a non-simple way [24]. Thus, all of these approaches will eventually fail as the colour table increasingly becomes inaccurate in changing illumination conditions.

Whatever the machine learning techniques used and representation of the segmentation function $c\_class : Y \times U \times V \rightarrow C$ is, the aim had been to have high accuracy on the entire space of (y, u, v) values and then segment an entire image. Thus, classifiers were built with the requirement to label all pixels which are perceived from a known colour. But, under another set of illumination conditions, the function $c\_class : Y \times U \times V \rightarrow C$ is rather different. In fact, one could say the c_class_{IC} depends on the illumination conditions IC. One approach is to detect what IC is frequently during play and switch the colour-segmentation function on board of the robot dynamically [9]. This assumes that one can reliably identify

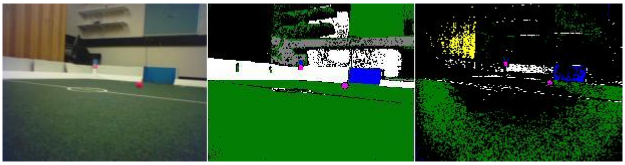


Fig. 2. *Sparse* classification. The image on the left is segmented by a colour classifier under known lighting conditions. The sparse classified image on the right only labels pixels that are in the same color class in a variety of illumination conditions.

IC and have a reasonable rich set of pre-loaded segmentation functions. However, the possible universe of values for IC implies a second stage of classification, namely the reliable identification of the suitable function c_class_{IC}.

In contrast to previous work, we create a classifier that is as *sparse* as possible. That is, we only want to classify a pixel belonging to a colour class if we are sure that it will remain in that class in a wide range of illumination conditions. Our approach is to consider identifying first that region of the YUV space for which given a colour class (say blue) c_class_{IC} remains constant and equal to blue for most values of IC. That is, our experience suggests that there is core region in YUV space where most illumination conditions will regard this as blue. We want our classifier to provide no label for those values where a (y, u, v) triplet fluctuates between green and blue along many illumination conditions.

Fig 2 illustrates this difference. The left segmented image shows the result of a traditional classifier trained with corresponding lighting conditions. The right segmented image results from our idea of a sparse classifier that is trained to classify only those pixels that it can surely labeled correctly across a variety of illumination conditions. We call this “sparse colour segmentation”. Standard object recognition systems would find a sparse colour segmentation useless because large blobs of a common colour are non-existent within the image. Our system will not use the colour-labeled pixels for that purpose so, as we will see in the Section 4, a sparse classification is actually preferable.

In supervised learning, a training and test sets are used to produce a classifier. Namely, the function $c\_class : Y \times U \times V \to C$ can be learned from many pairs of the form $[(y, u, v),$ C_CLASS]. Earlier work [25] has shown how to learn a very efficient and succinct representation encodes the classifier as a decision list to take advantage of the parallelism of bytes. This representation decomposes the function *c_class* into simpler functions and observes that the corresponding fast C++ implementation essentially tests conditions as decision lists [26]. The learning algorithm PART [27] as implemented in WEKA [26] has been very successful in learning a representation of *c_class* that is no longer than $256 \times 4 bytes = 1$ KB. The accuracy is above 98% for operating on the same illumination conditions as where the training and test sets were collected.

We describe how to obtain a new classifier that would produce a class label for those regions of the YUV space where c_class_{IC} remains constant across a large number of values for IC. We use a training set T_{IC} to learn a classifier $decision\_list_{IC}$. We repeat this process for 5 different illumination conditions of the same field (additional base classifiers are required if we are moving the field into a different room). We call the resulting classifiers $decision\_list_{IC_i}$, for $i = 1, \ldots, 5$. We then process all source training sets T_{IC_i} used to learn the base-classifiers. For each example $[(y, u, v),$ C_CLASS] we test if all 5 classifiers $decision\_list_{IC_i}$ agree on a classification (and if that classification is C_CLASS). If that is the case, this example is appended into a new global training set T. Otherwise, the example is placed in the training set with the label replaced by the value unknown. Once all training examples for the case classifiers have been processed, the new training set T is used to learn the sparse classifier. All learning stages use the PART algorithm and are represented as decision lists. We should indicate that we were able to obtain a sparse classifiers that required 256 × 14 bits (that is, 448 bytes) and had a precision of 99%.

4 Our Blob Forming

The final step before object recognition is blob forming. We take advantage of the edges we have found in the first step. The edges of our blobs result from a border-following algorithm that creates a list of pixels that represents the boundary of each region of interest in the image. This avoids the need for complex union-find methods to build blobs from colour labeled pixels. Union-find algorithms are quadratic in complexity on the average diameter of the blob because they must inspect every pixel in the blob. Border tracing is a linear operation on the average diameter of the blob so is clearly a preferable method.

The edges themselves contain insufficient information to locate regions of the image which are interesting so we use the sparse colour information we have obtained by classification. We are fairly sure that anything labeled as, say, blue, is actually blue in any lighting condition. Therefore a blob that contains many more blue pixels than any other colour is likely to be a blue object. We use our colour labeled pixels as a set of sample points and use Algorithm 1 to establish the blob that each sample point lies in. As we build the list of pixels describing the blob in Line 4, it is trivial to compute other properties of the blob such as its bounding box or relation to the horizon of the image. A frequency histogram of colour classes found within the blob is also built by Line 7.

In a naive implementation of this algorithm, the test in Line 3 could be computationally time consuming. Since there will be many blobs, and each blob will contain many pixels, there will be very large list of pixels to search to test if our current pixel is already assigned to a blob. We recommend several optimisations that make this algorithm much more efficient. Instead of searching the existing boundary lists, we label (on the segmented image) each pixel in the boundary of a blob with the blob ID of that pixel. This process is performed in Line 4 of the algorithm as the boundary list is being constructed. Since we have 8-bits per pixel in a segmented image, and only 15 or so colour labels, we are free to use the other 240 values as identifiers for blobs. In this way the test in Line 3 is a single memory access to determine a blob ID.

Algorithm 1. Blob Form

```
Input: A set, $P$, of seed points. An image where pixels on edges have been identified.
Output: A set, $B$, of blobs where each blob is represented by the pixel list that forms
    its boundary.
for all Points, $p \in P$ do
    Find the pixel, $e$, in the edge above $p$.
    if $e$ is not in a blob $b \in B$ then
        Trace boundary to find pixel list describing a new blob $b'$.
        Insert $b'$ into $B$.
    else
        Update colour information on $b$.
    end if
end for
```

5 Our Object Recognition

Fig. 3 illustrates how our object recognition system, based on fast edge detection, is preferable in changing illumination conditions to a traditional object recognition system based on colour segmentation. The images in the left column are the source images in several illumination conditions. Refer again to Fig. 1 for a description of the illumination conditions. We illustrate our illumination independent object recognition by a series of images that represents different stages in our vision processing pipeline. Even though the pixel classification changes in each image (column two), the blob forming stage (column three) remains fairly stable because it is based on our fast edge detection routine. Therefore the objects on the field (a blue/pink beacon, the blue goal and a pink[2] ball) are recognised correctly in each image despite the illumination changes.

The task of object recognition is very similar to traditional systems because the blobs are recognised accurately in a wide variety of lighting conditions. We have several interesting blobs that require further investigation to determine if they are actual objects such as the goals, beacons or the ball. One advantage of our particular system is that blobs inherently are composed of a list of edge pixels. In prior work we have demonstrated a linear time algorithm for straight edge vectorisation [28]. This means that it requires very little processing overhead to vectorise the blob's boundary. This is especially useful when processing the field blobs as it allows us to identify field edges and corners. There are several other situations where having the boundary of the blob already identified is very helpful. For example the centre of the ball is often found by using the mathematical technique of perpendicular bisectors [29]. This relies on knowledge of the boundary of the ball. There are methods now for linear time detection of circles which also require the boundary to be identified [30]. Aside from these, there

[2] The ball is pink in these images rather than the league standard of orange because at the time of writing there was a proposed rule change to use a pink ball. This rule change has since been rejected.

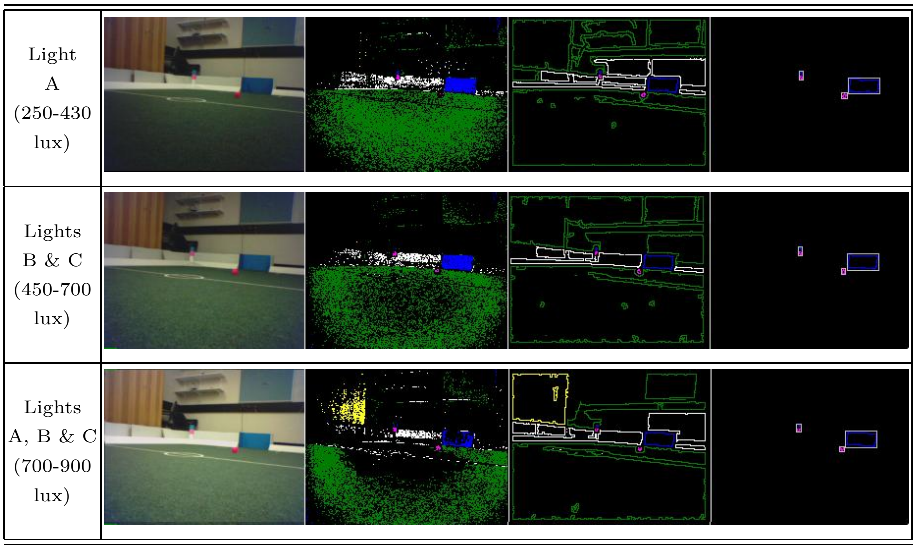


Fig. 3. Illustration of our object recognition process under a variety of illumination conditions. Source images are shown before their sparse classification. The result of edge detection can be seen in Fig. 1. The third column shows the result of our blob forming algorithm and the final column illustrates all objects properly identified.

are many techniques in both RoboCup and other literature about mechanisms for analysing blobs to determine the real-world objects properties [31, 32, 3].

We have made available on our website[3] a video that demonstrates our ability to track a standard RoboCup ball in variable lighting conditions. There are four test cases illustrated on the video:

Standard Illumination: Light rows A, B, and C (refer again to Fig. 1 (a)).
Medium Illumination: Light rows A and C.
Low Illumination: Light row B.
Dynamic Shadows in Low Illumination: A solid sheet is passed between the light source and the field in random motion.

The same sparse classifier is used in each of the test cases and all four cases were filmed without rebooting the Aibo. The automatic camera setting on the Aibo (auto white balance) is *off* and the camera is fixed to medium gain, medium shutter speed and fluorescent lighting mode.

This video demonstrates that Aibo tracking the ball in each lighting environment. The lights on the head of the Aibo are indicative of how well it is tracking the ball - bright white lights indicate that it sees the ball clearly and red lights indicate that it is unsure of the ball but it has not lost it completely[4]. No lights

[3] http://www.griffith.edu.au/mipal

[4] This can happen, for example, when the edge detection contains a flaw which allows the border trace algorithm to "escape" the ball at some point around it's border. In this case we end up with a very large, unlikely ball and we wait until the next vision frame to confirm its location.

Table 2. Comparing our old object recognition system to our new one. The old system had no edge detection component and edge detection and classification are merged in the new algorithm. All times are measured in AVW2 profiling clicks described in [20].

	Edge Detection	Classification	Blob Forming	Object Recognition	Total
Standard Algorithm	-	4.2	21.8	12.9	38.9
Our Algorithm	20.9		18.6	12.9	52.4

would indicate a complete loss of the ball although this does not happen in this video. Notice that our object recognition system copes with all of the lighting conditions sufficiently well. Although we see some minor degradation of performance in the final test case, even in this extremely complex lighting environment we illustrate sufficient competence to track a ball. Notice also that our object recognition system does not confuse skin colour with the ball, even in the final test case with low light and dynamic shadows.

Finally, we compare the overall performance of our object recognition algorithm with the performance of our prior one ([25, 31, 29]). Table 2 shows the running time of each stage of the two object recognition algorithms. The tests were done on a set of 100 images. Our new object recognition process is approximately 35% slower than our old one. This is mostly due to the time required for the edge detection step. The algorithm is, however, still well within the performance requirements for running as a real-time vision system on a SONY Aibo.

6 Conclusion

We have been able to overcome the problems associated with illumination independent object recognition by shifting the focus in the process from the step of colour segmentation to that of edge detection. We have developed a very fast edge detection algorithm to make this feasible even on platforms, such as the SONY Aibo, with limited computational capacity. By relying on edges in the real image to form blobs, rather than boundaries in classified images, we have removed the problem of requiring a highly accurate classification. We instead label only those pixels that we are fairly certain will be classified correctly over a range of various lighting conditions.

Traditional colour segmentation algorithms, by nature, do not work in dynamic illumination conditions. They are specifically trained for one particular condition and they become increasingly inaccurate as they are moved further from their initial training condition. Thus the traditional approach of forming blobs based on the results of colour segmentation is severely hampered in any dynamic lighting condition. On the other hand, a suitable classifier can be built for a wide range of lighting conditions provided it is freed from the restriction of being as complete as possible. Thus it is possible to use the colour of pixels, even in highly variable lighting conditions, once the blob is identified in some other way.

We have presented a fast enough edge-detection algorithm to run on the Aibo in RoboCup game conditions. We have used this algorithm as a basis for a new object recognition system that uses colour information but does not rely on training a classifier in a way specific to any single lighting condition. We have been able to produce good results in object recognition over a variety of illumination conditions using this technique.

References

1. Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color segmentation for interactive robots. In: Int. Conference on Intelligent Robots and Systems, IEEE Computer Society Press (2000)
2. Grillo, E., Matteucci, M., Sorrenti, D.: Getting the most from your color camera in a color-coded world. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
3. Röfer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jünge, M., Laue, T., M. Lötzsch, W.N., Spranger, M.: German team 2004 - the german national robocup team. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
4. Dahm, I.: Fully autonomous robot color classification. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
5. Cameron, D., Barnes, N.: Knowledge-based autonomous dynamic colour calibration. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
6. Dahm, I., Deutsch, S., Hebbel, M.: Robust color classification for robot soccer. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
7. Jüngel, M., Hoffman, J.: A real-time auto-adjusting vision system for robotic soccer. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
8. Meyer, G., Utz, H., Kraetzchmar, G.: Playing robot soccer under natural light: A case study. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
9. Sridharan, M., Stone, P.: Towards illumination invariance in the legged league. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
10. Steinbauer, G., Bischof, H.: Illumination insensitive robot self-localization using panoramic eigenspaces. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
11. Gönner, C., Rous, M., Kraiss, K.: Real-time adaptive colour segmentation for the robocup middle size league. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
12. Lange, S., Riedmiller, M.: Evolution of computer vision subsystems tasks. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
13. Austin, D., Barnes, N.: Red is the new black - or is it? In: Proc. of the 2003 Australasian Conference on Robotics and Automation. (2003)
14. Finlayson, G., Hordley, S., Hubel, P.: Colour by correlation: A simple, unifying framework for colour constancy. IEEE Transactions on Pattern Analysis and Machine Intelligence **23(11)** (2001)
15. Finlayson, G., Hordley, S.: Improving gamut mapping color constancy. IEEE Tranactions on Image Processing **9(10)** (2000)
16. Finlayson, G.: Color in perspective. IEEE Transactions on Pattern Analysis and Machine Intelligence **18(10)** (1996) 1034–1038
17. Brainhard, D., Freeman, W.: Bayesian colour consistency. Journal of Optical Society of America **14(7)** (1986) 1393–1441

18. Buchsbaum, G.: A spatial processor model for objecct color perception. Journal of Franklin Institute **310** (1980) 1–26
19. Murch, C., Chalup, S.: Combining edge detection and colour segmentation in the four-legged league. In: Proc. of the 2003 Australian Robotics and Automation Association. (2003)
20. Lovell, N.: Real-time embedded vision system development using aibo vision workshop 2. In: Proc. of the Mexican Int. Conference on Computer Science. (2004)
21. Gonzalez, R., Woods, R.: Digital Image Processing. Addison Wesley (1992)
22. Chen, J., Chung, E., Edwards, R., Wong, N., Mak, E., Sheh, R., Kim, M., Tang, A., Sutanto, N., Hengst, B., Sammut, C., Uther, W.: rUNSWift team description. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2003)
23. Kaufmann, U., Mayer, G., Kraetzschmar, G., Palm, G.: Visual robot detection in robocup using neural networks. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
24. Quinlan, M., Chalup, S., Middleton, R.: Techniques for improving vision and locomotion on the sony aibo robot. In: Proc. of the 2003 Australasian Conference on Robotics and Automation. (2003)
25. Estivill-Castro, V., Lovell, N.: Improved object recognition - the robocup 4-legged league. In Liu, J., Cheung, Y., Yin, H., eds.: Intelligent Data Engineering and Automated Learning, Springer (2003) 1123–1130
26. Witten, I., Frank, E.: Data Mining — Practical Machine Learning Tools and Technologies with JAVA implementations. Morgan Kaufmann, California (2000)
27. Frank, E., Witten, I.: Generating accurate rule sets without global optimization. In: Machine Learning: Proc. of the Fifteenth Int. Conference, Morgan Kaufmann (1998)
28. Lovell, N., Fenwick, J.: Linear time construction of vectorial object boundaries. In Hamza, M., ed.: 6th IASTED Int. Conference on Signal and Image Processing, ACTA Press (2004)
29. Estivill-Castro, V., Fenwick, J., Lovell, N., McKenzie, B., Seymon, S.: Mipal team griffith - team description paper. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
30. Sauer, P.: On the recognition of digital circles in linear time. Computational Geometry: Theory and Applications **2** (1993) 287–302
31. Lovell, N., Estivill-Castro, V.: A descriptive language for flexible and robust object recognition, Springer-Verlag (2004)
32. Seysener, C., Murch, C., Middleton, R.: Extensions to object recognition in the four-legged league. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)

On-Line Color Calibration in Non-stationary Environments*

Federico Anzani[1], Daniele Bosisio[1]
Matteo Matteucci[1], and Domenico G. Sorrenti[2]

[1] Politecnico di Milano
matteucci@elet.polimi.it
[2] Università di Milano - Bicocca
sorrenti@disco.unimib.it

Abstract. In this paper we propose an approach to color classification and image segmentation in non-stationary environments. Our goal is to cope with changing illumination condition by on-line adapting both the parametric color model and its structure/complexity. Other authors used parametric statistics to model color distribution in segmentation and tracking problems, but with a fixed complexity model. Our approach is able to on-line adapt also the complexity of the model, to cope with large variations in the scene illumination and color temperature.

1 Introduction

Color is a strong clue for object recognition, and for this reason it is used in many industrial and research applications. The real-robot leagues of Robocup, one of which is the mid-size league, are challenging applications conceived to use information about color. Here we have color codes for each relevant object in the robot world, altogether to changing lightning conditions in a dynamic scene with many objects to recognize. These varying conditions represent a non-stationarity in the environment and they are the main issues that prevents a reliable object recognition in natural light conditions, as clearly stated in [1].

The literature in this field is huge, we just mention here some works from Robocup literature. Cameron and Barnes [2] approach the problem relying on the a priori known scene geometry for identifying some regions of the image as corresponding to some features of the environment; this is done without the use of color-classification, but on domain-specific knowledge instead. Then the color information of such regions is used for building the color classifier(s); in order to track the dynamics of the lightning they associate the would-be-current classifier(s) with the previous, so to keep them tied to the symbolic label(s).

In Jüngel et al. [3] colors are defined by simple partitions parallel to the color space axes; in order to attain adaptation the colors are defined relative to a reference color which, in the Robocup legged league, the example application of

* This work has been partially supported by the MADSys project P.R.I.N. 2003 "Software technologies and knowledge models for design, implementation and testing of multi-agent robotic system in real environments", funded by M.I.U.R.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 396–407, 2006.

the paper, is the green. No explanation is given about this choice. Our opinion is that the reason is the (domain-specific) observation of its nearly-granted presence in the image (as mentioned e.g., by [4]). The setup for the classification of each color is executed off-line, and then a domain-specific heuristic is used to track the reference color, while other colors are classified by displacing their region in the color space by the same amount the reference color has moved. This is risky, as noticed by [1], because distances in the color space can be stretched by changes in the illumination.

Color constancy [5] approaches have been also used to reconstruct the incident light and adjust perceived colors for automatic color calibration [6]. However these approaches have unpractical computational requirements and thus cannot be applied on-line for real-time tracking of color changes. Different issues prevent the effective use of histogram based approaches [7]; they require a large number of data-points (pixels) and a coarsely quantized color space. In the absence of an accurate model for the apparent color, which changes over time, these models for density estimation cannot be obtained and a different approach is required. The subsequent sections briefly presents the parametric approach to color modeling which is at the basis of our contribution and the algorithm used for adaptation; experimental results on real images and conclusions are presented in the final sections.

2 Mixture of Gaussians for Classification and Modeling

In order to attain adaptivity, we use a generative approach, which implements a probabilistic color modeling and classification scheme based on mixture models. This approach has been originally proposed to implement object tracking under non-stationary illuminating conditions [8, 9]. In this context, the color of each pixel is supposed being determined according to a given probability distribution, which depends on the colors in the observed object and is characterized by some parameters.

Let us begin by temporary forgetting the color labels and limiting to model the probability distribution of color values in the color space. This distribution will be multi-modal not only because we expect more than one color, but also because more than one tone will be present for some color; this can be due, e.g., to different lightning conditions, in different places, for the same object. This probability distribution allows a probabilistic classification of the image pixels and in our case is approximated by the use of a parametric model, i.e., a mixture probability distribution.

In the scheme introduced by McKenna in [9], we have pixel values $\mathbf{x}$ in a color space of D dimensions (i.e., 2 dimensions in McKenna's word), an object O, and $j = 1, ..., m$ components, which live in the color space too. The model is trained on the pixels known to belong to the object; the whole object is represented with a mixture of Gaussians, i.e., components. Our problem requires to deviate a little from this object-based scheme, because our aim is color modeling and not direct object recognition. We still have pixel values in the color space (who's dimensions are also called color components, but we shall not use this term component with this semantic any more, to avoid confusion with mixture model components), but we have not a single object to recognize. We have many, i.e. much more than one, color labels which we would like to recognize. Each color label could require more than one component in the color space, for an adequate modeling. This trained model will be used for classification of online data.

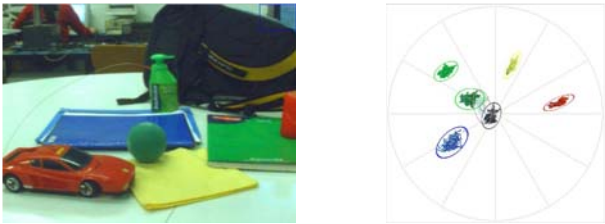
To gain a better understanding of the parametric model involved by such a mixture distribution, suppose we have J different colors in the scene and let $\mathbf{x}$ be the vector of numbers representing the pixel values in the color space (e.g., the RGB or HSV coordinates for that pixel). The probability of observing this vector depends on the probability of the vector of features given the real color label of the object in the scene C_j. However, suppose we do not know in advance the real color of the observed object, but only the probability of the observed object being of a certain color $p(C_j)$, by the total probability theorem, we can write the probability of each pixel observed as

$$p(\mathbf{x}|\Theta) = \sum_{j=1}^{J} p(\mathbf{x}|\theta_j, C_j)p(C_j) \tag{1}$$

where $\sum_{j=1}^{J} p(C_j) = 1$, $p(C_j)$ is the so called *mixing probability* and corresponds to the prior probability that $\mathbf{x}$ was generated by the j^{th} component; $p(\mathbf{x}|\theta_j, C_j)$ is the conditional density for the pixel color features given that this pixel belongs to color component C_j (i.e., our generative model). The term Θ, in Equation 1, represents the set of all parameters describing the pixel probability distribution including conditional probability parameters θ_js and color label probabilities $p(C_j)$s. We can extend this mixture model to have more components than the J colors in the scene since we can associate two or more components to the same color label. Moreover, if we suppose to have M components and the conditional probability of the feature color vector being a Gaussian density, the probability of a pixel feature vector becomes a Gaussian mixture model [10, 9]

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{M} \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}[\mathbf{x}-\mu_k]^T \Sigma_k^{-1}[\mathbf{x}-\mu_k]} \cdot p(C_k) \tag{2}$$

In this model, each mixture component is modeled by a Gaussian distribution with mean μ_k and covariance matrix Σ_k. This generative model is somehow independent from the logical color label associated to each component and its parameters can be obtained by a maximum likelihood estimation of mixing probabilities $p(C_j)$s and Gaussians parameters θ_js. The association between each mixture component and the appropriate color label can be obtained by a (human) supervised labeling of components. Actually, in our method, this is implemented in a different way by using a (human) labeled dataset to initialize mixture parameters (i.e., initial number of components, mixing probabilities, and Gaussian parameters) and associate from the very beginning the appropriate labels to the components of the mixture.



(a) (b)

Fig. 1. An example of color density model obtained using a mixture of Gaussians, on the left the original image, on the right the probability distribution of image pixels; the lines represent the curve where the cumulative probability reaches the 0.9 value

Modeling the color distribution with a mixture of Gaussians can be easily explained: in Figure 1(b) patches of colors from image in Figure 1(a) are modeled by a mixture of 2D Gaussian in the Hue-Saturation (HS) color space. Black and white pixels are also projected on the $V = 255$ plane just for easing the visualization; please notice how the white pixel of the image have a greenish hue as well as the presence of two different green components due to spatial variation of color.

At classification time, each pixel is classified according to the conditional probability of being generated by a certain color component to which is associated a label. This is implemented by the maximum a-posteriori criterion for the selection of the color component, and label, by attributing to the pixel the color component, and label, of the component which is the most likely:

$$C(\mathbf{x}) = \arg\max_{C_j} p(C_j|\mathbf{x}, \Theta) = \arg\max_{C_j} p(\mathbf{x}|\theta_j, C_j)p(C_j). \quad (3)$$

In really dynamic environments, the mixing probability might change almost at any frame due to occlusions and changes in the field of view. This would interfere with the maximum a-posteriori criterion since the prior probability for each class $p(C_j)$ would change with the observed scene. For instance, suppose we miss the red ball for a few frames because of occlusions, using a null mixing probability as prior will prevent to detect it as it will come back in the field of view. This can be faced by adopting, at classification time, an improper uniform prior for color labels (i.e., $p(C_j) = 1/J$) turning the maximum a-posteriori approach into maximum likelihood classification:

$$C(\mathbf{x}) = \arg\max_{C_j} p(C_j|\mathbf{x}, \Theta) = \arg\max_{C_j} p(\mathbf{x}|\theta_j, C_j). \quad (4)$$

3 On-Line Adaptation with Expectation-Maximization

We can now come to the main issue of the work, i.e. adaptation of the model to changing light conditions. To overcome issues due to non stationarity, we use *Expectation-Maximization* (EM) algorithm [11] to adapt model parameters over time. EM is an iterative algorithm that computes maximum likelihood estimation, in a very efficient and fast way, by optimally coping with missing data (i.e., we do not know mixing probabilities in future frames). Let N be the number of data-points in the data-set. The Estimation step of EM computes the "expected" classes at time t for all data points $\mathbf{x}_n$ using the current Gaussian parameters:

$$p(C_j|\mathbf{x}_n, \theta_j^{(t)}) = \frac{p(\mathbf{x}_n|C_j, \mu_j^{(t)}, \Sigma_j^{(t)})p(C_j)^{(t)}}{\sum_{i=1}^{M} p(\mathbf{x}_n|C_i, \mu_i^{(t)}, \Sigma_i^{(t)})p(C_i)^{(t)}} \tag{5}$$

The Maximization step of EM computes the maximum likelihood estimates of the mixture distribution given the data class membership distributions [12]:

$$\mu_j^{(t+1)} = \frac{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})\mathbf{x}_n}{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})} \tag{6}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})[\mathbf{x}_n - \mu_j^{(t+1)}][\mathbf{x}_n - \mu_j^{(t+1)}]^T}{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})} \tag{7}$$

$$p(C_j)^{(t+1)} = \frac{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})}{N} \tag{8}$$

EM is proved to reach a local maximum, and the optimality of the extremum depends on the initial estimate of mixture parameters, especially when these parameters are randomly selected. In our application, the components means and variances are initialized by using hand-classified sets of pixels from the first frame; the mixing probability is taken proportional to the density of the corresponding subset of data used in the initialization and the number of components is initializated to the number of classified sets.

From each subsequent frame, a new set of pixels is sampled and inserted in the data-set without a label. One step of EM is computed on this augmented data to provide model adaptation. A soft and stable adaptation is granted by weighting data points, increasingly from the older to the newer, by using a forgetting factor. To avoid excessive computations with reduced marginal gains, data points older than a fixed threshold are removed from the data-set.

To understand how EM is able to track non-stationary distributions and thus adapt the color model to illumination changes, refer to Figure 2. In this case we present a (simulated) data-set where a cloud of points, referring to the same color, moves in the color space during time because of non-stationary light conditions. From the image sequence it can be noticed how the algorithm performs

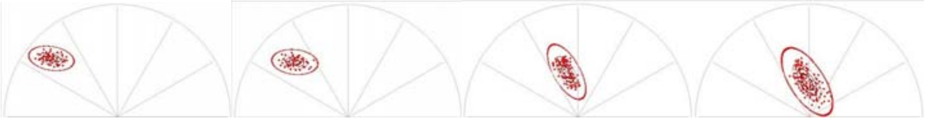


Fig. 2. Distribution tracking using EM (upper part of HS diagram reported)

an unsupervised probabilistic clustering of the data while adapting to the distribution changes.

4 Adaptation of Model Complexity

In modeling with mixture of Gaussians, the main problem is to choose the right structure for the model, i.e, the *right* number of components of the Gaussian mixture. A common approach in stationary situation (e.g., data mining) is to try every possible structure (up to a given threshold) and select the best one. However, this method cannot be used in real-time dynamic contexts because of both non-stationarity and excessive computational cost. In the following subsections we extend the mixture of Gaussian model previously described by introducing an automated procedures to on-line adapt the model complexity.

4.1 Reducing Mixture Components

Since data distribution varies over time, the number of Gaussian mixtures could become higher than the optimal value. This strongly affects the performances of the algorithm and could lead to overfitting the noise in the data. This unrequired complexity of the model can be reduced by merging clusters that could be represented by only one Gaussian without losing precision in the model and preserving the correct classification with respect to color labels. This situation happens quite often when the scene become darker (e.g., clouds in natural light context or light bulbs switched off in offices) and the color distribution, as represented in the HSV space, concentrates on the bottom of the cone.

To perform cluster merging we use the greedy approach; at every frame, before computing EM, the two nearest components, for every label, are selected as candidate for the merging. In our current implementation we use the Euclidean distance although better and more sound results could be obtained using the Mahalanobis distance, given that our components are Gaussian. To decide if they have to be merged or not, a new Gaussian component is created with mean and variance computed using all the points belonging to the two candidate-to-the-merge components.

In order to estimate which of the two models (i.e. the merged single component or the original two separate components) is the best model, the Bayesian Information Criterion (BIC) [13] is used. Assuming we are given the data $\mathcal{D}$ and a family of alternative models M_j, this criterion uses the posterior probabilities $p(M_j|\mathcal{D})$ to give a score to the models, using the following formula, from Kass and Wasserman [14]:

$$BIC(M_j) = \hat{\mathcal{L}}_j(\mathcal{D}) - \frac{P_j}{2} \log R \tag{9}$$

where $\hat{\mathcal{L}}_j(\mathcal{D})$ is the log-likelihood of the data according to the j-th model, taken at its maximum-likelihood point, P_j is the number of free parameters in M_j and R is the number of data used to fit the model. In order to increase the speed only two components are examined at a time. The algorithm iterates until no more components can be merged, according to the BIC scoring.

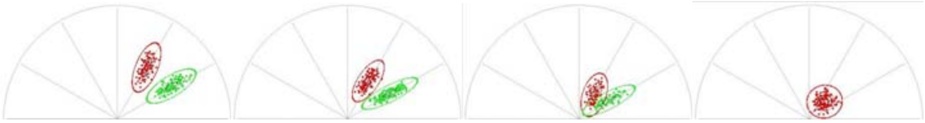


Fig. 3. An example of merging of components, in the HS plane. The components can merge only if they have the same label. This is a synthetic example, built after real cases, which, for an easier understanding, presents components referring to just one label. This behaviour usually shows up when the intensity of the light on the scene decreases.

4.2 Increasing Mixture Components

In this section we present our proposal for overriding the opposite problem of merging components, i.e. when one cluster of pixel values in the color space divides in two. If this situation goes undetected the model will loose one of the two, by classifying it as noise; or, alternatively, the two clusters of pixel values will be modeled as a single component with large variance. This situation usually occurs when the observed scene becomes brighter and the different tones of the same color begin to separate in the color space.

We propose a very simple, but effective, approach in order to detect potential splitting cases based on a measure of data-points density. Let $N_{(p)}$ denote the number of data-points contained in a given confidence interval $V(\overline{p})$, i.e., the hyper-ellipsoid centered on the Gaussian mean and containing in its volume a probability mass $\overline{p}$. The average density $D(\overline{p})$, as a function of the confidence $\overline{p}$, can be computed as: $D(\overline{p}) = N_{(\overline{p})}/V(\overline{p})$.

The approach is based on the observation that when we would like to split, we have a much lower density around the mean of the component than the average values across the component. $D(0.95)$ could be considered a reasonable approximation of the average density for the whole component while $D(0.25)$ can be taken as a reasonable approximation of the density about the mean. Using a K coefficient, empirically we found that a good value for K is between 1.5 and 1.8, if $D(0.95) > K \cdot D(0.25)$ we decide the current model does not fit the data distribution well enough. We then split the cluster along the direction of maximum variance. The two resulting components will be initialized with mean at $\frac{1}{4}\sigma$ and $\frac{3}{4}\sigma$ respectively, and a diagonal covariance matrix with diagonal elements set to $\frac{1}{4}\sigma^2$. The subsequent EM steps will fine tune the parameters of the components, Figure 4.

It is quite easy to find counter-examples, e.g. in data-mining, where this heuristic does not work properly, but we found in our experiments that in the color domain, it is highly reliable. Moreover, it involves a small amount of computation and does not slow-down the algorithm significantly. Another solution could be to split all the clusters, calculate BIC and then decide which model is the best as we do for merging. The different approaches used in increasing and reducing model complexity are due to the cost of computing the parameters of the Gaussians. In a merging operation calculating the parameters of the new

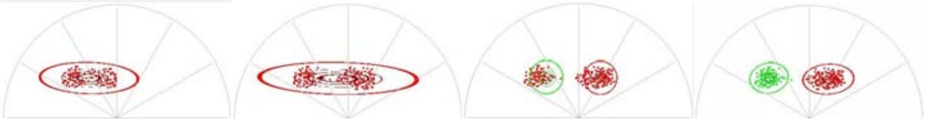


Fig. 4. An example of splitting of components, in the HS plane (left to right). Note in the third frame that the two new components are initialized to symmetrical Gaussians; in the next frame the EM algorithm adjusts their parameters.

cluster is easy and fast; in the opposite operation, finding the two new centroids is more difficult and expensive because some EM steps are needed.

4.3 Stopping the Adaptation

Although adapting the model structure and its parameters seems to be the definitive solution to color model tracking, using an adaptive color model could cause different problems, first of all due to the lack of ground-truth. Citing from [8]: "any color-based tracker can loose the object it is tracking, due, for example, to occlusion or object disappearing from the field of view".

As an object disappears its color component(s) are removed from the color space. The color model, i.e. its component(s), will try thus to fit image regions which do not correspond to the object. A method to detect such data associations problem and then to stop the adaptation of that component is therefore needed. This idea of stopping adaptation is adapted from the selective adaptation of McKenna et al. [8] (the pun is intentional). Moreover, it will not suffice to stop adaptation at all, but it will be necessary to detect this situation, and eventually to stop adaptation, selectively for each component.

Our proposal to circumvent this problem is based on the observed log-likelihood measurements. The EM algorithm maximizes the log-likelihood of the color data over time. Let $\mathcal{L}_i$ be the normalized log-likelihood for the i^{th} component, and $X^{(t)}$ be the data-set at frame t we have:

$$\mathcal{L}_i^{(t)} = \frac{1}{N^{(t)}} \sum_{\mathbf{x} \in X^{(t)}} \log p(\mathbf{x}|\theta_i) \tag{10}$$

At each frame t, $\mathcal{L}_i^{(t)}$ is evaluated, and this is done for each component. If the tracker looses the object(s) connected to the i^{th}-component, a large drop in the $\mathcal{L}_i^{(t)}$ will occur. The component adaptation is then suspended until the model gains again a sufficient support from the data. In practice we have a threshold $T_i^{(t)}$ and adaptation is performed only if $\mathcal{L}_i^{(t)} > T_i^{(t)}$. The mean, $\nu_i^{(t)}$, and standard deviation, $\sigma_i^{(t)}$, of $\mathcal{L}_i^{(t)}$ are computed, in our experiments, for the n most recent above-threshold frames. The threshold is $T_i^{(t)} = \nu_i^{(t)} - k\sigma_i^{(t)}$, where k is a constant whose value has been experimentally set at about 1.5. The whole set of components, under online adaptation or not, are used for classification purposes instead.

5 Experiments on Real Data-Sets

In the following we present some experiments in order to validate our proposal. These are on real data-sets, one source is the set of images[1] provided by the authors of [1], and the other source are video sequences grabbed in our laboratory. In some cases we use 2D plots to explain the results, but notice that the adaptation always took place in the 3D HSV color space. In this color space, data-points can translate or rotate. Translations are usually due to changes of the intensity of light, while rotations are due to changes in the color temperature. In all the experiments, the algorithm has been initialized by selecting a certain number of patches from the first frame in the sequence; the initial number of compents has been set to the number of patches selected and the initial mixture parameters has been estimated from the pixel patches.

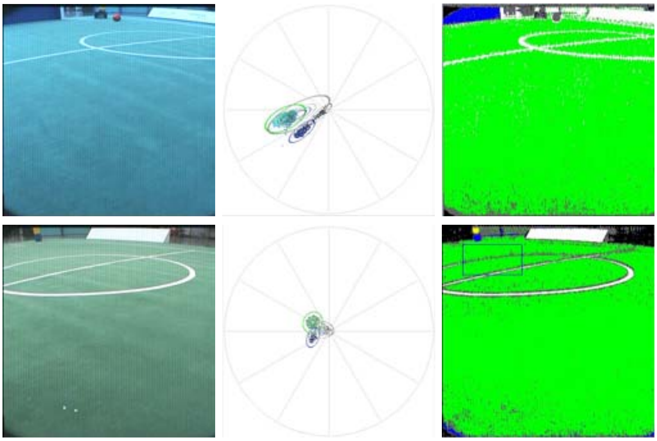


Fig. 5. The different color distribution obtained by adapting to the two images on the left, treated like they were consecutive. The upper image has been grabbed during natural day lights, the one underneath has been taken during evening with neon lights.

The first example describes how the algorithm works with respect to changes of light conditions. Normally this is due to day/night cycle. Assuming slowly varying conditions, the model adapts and the variance of the components grows if the scene becomes brighter and decreases if becomes darker. Components with different labels will not be merged, so the real data distribution is well described. Hereafter we present the results obtained in a really extreme situation: an abrupt change of the light on the scene, from natural day light in one image, to artificial (neon) light at night in the other; the two were put in sequence and the results show the capability of the algorithm to adapt, see Figure 5. We think that such an abrupt change is a realistic estimate of what can be perceived by a generic indoor mobile robot. Therefore this confirms the capabilities of our system to make really more robust the perception system.

[1] Available at `http://smart.informatik.uni-ulm.de/DataSets/RoboCup/natLight/`

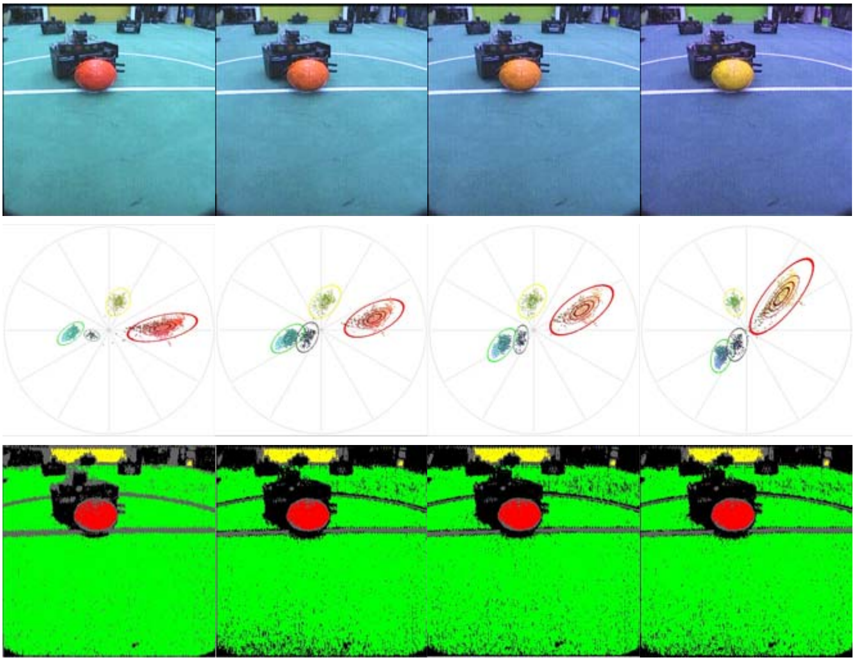


Fig. 6. The algorithm adapts the components to Hue variations, see in the central row how the color distribution rotates in HS plane, and how the image classification remains good. The images have been concatenated as if they were a sequence.

The second experiments validates the adaptation of the algorithm with respect to changes of the color temperature. We hand-manipulated some images, taken from the web-site mentioned before, changing the hue channel, see Figure 6, in order to mimic situations of natural light in outdoor environments. In such situations the light is not always white and changes with time; for example, in the morning it usually tends to move toward the blue, while in the evening tends to move toward the red. In such situations adaptation on the Hue component is really relevant. As it can be checked in the pictures, the algorithm allows a correct and robust image classification over time by adjusting the components to the real data distribution, rotating in the HS plane.

The last experiment presents a bright scene that becomes dark. The model adapts over time diminishing the number of cluster during the darkening phase, when the data-points concentrate in the bottom of the HSV cone. The experiment demonstrate a very good performance and only very few errors are due to mis-classification of noise, moreover this happened in very difficult, i.e. dark, conditions (Figure 7). When the scene will be bright again the split algorithm will divide the components to keep-up with the number of clusters.

The algorithm for on-line color calibration presented has been developed to work in parallel with a complete 15fps color tracking system. The color look-up

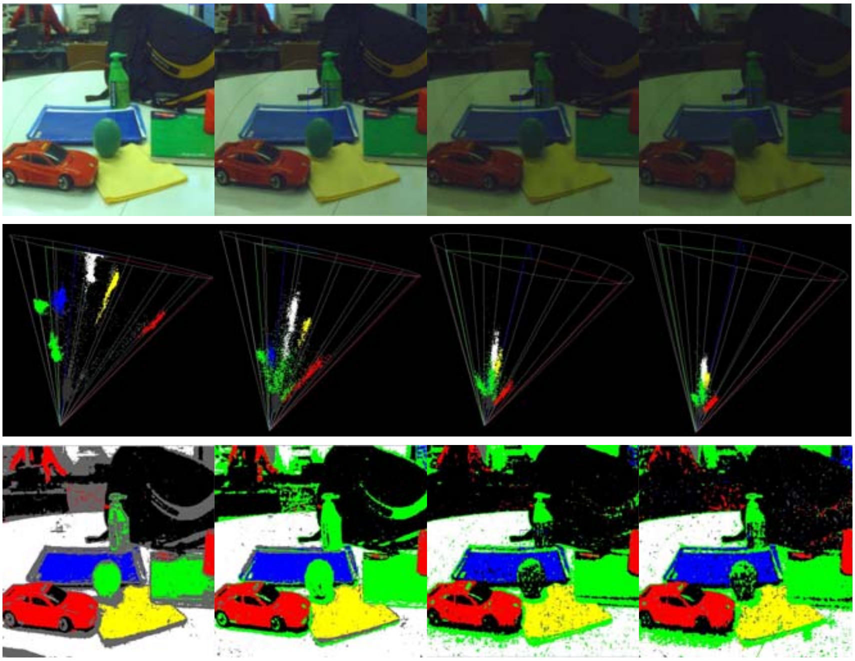


Fig. 7. From top to bottom: original images, HSV space, classified image. Images are taken from a 45 frames sequence grabbed in our lab.

table used for classification is updated by the current (unoptimized) algorithm at 5-8 Hz being the only thread on a P4 at 2.4Ghz using a subsample (80×60) of the original image.

6 Conclusions

Soccer robots must be able to play under natural illumination that can change over time both in intensity and color temperature. To obtain good color classifications, two approaches are available: to adapt the color model or to use algorithms enforcing color constancy. The first approach is today the only one allowing performances compatible with an on-line real-time use. We started with a known approach to color modeling, based on EM maximum-likelihood iterative estimation. This known algorithm includes the possibility of stopping the EM-based adaptation; we adapted this algorithm to the multi-target domain required by Robocup real-robots league, especially for what concerns the stopping adaptation functionality; more important is the addition of on-line model order adaptation. This is a relevant issue in color modeling, especially for enabling its use in dynamical and difficult domains like the real-robots Robocup as well as indoor robotics. Our proposal produced quite good results, and results in quite extreme experiments validate such claim.

References

1. Mayer, G., Utz, H., Kraetzschmar, G.K.: Playing robot soccer under natural light: A case study. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 238 – 249
2. Cameron, D., Barnes, N.: Knowledge-based autonomous dynamic colour calibration. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 226 – 237
3. Jüngel, M., Hoffmann, J., Lötzsch, M.: A real-time auto-adjusting vision system for robotic soccer. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 214 – 225
4. Jamzad, M., Lamjiri, A.K.: An efficient need-based vision system in variable illumination environment of middle-size robocup. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 654 – 661
5. Forsyth, D.A.: A novel algorithm for color constancy. International Journal of Computer Vision **5** (1990) 5 – 36
6. Mayer, G., Utz, H., Kraetzschmar, G.K.: Towards autonomous vision self-calibration for soccer robots. In: Intl. Conference on Intelligent Robots and Systems (IROS02), Lausanne, Switzerland (October 2002) pp. 214 – 219
7. Swain, M.J., Ballard, D.H.: Color indexing. International Journal of Computer Vision **7** (1991) 11–32
8. McKenna, S.J., Raja, Y., Gong, S.: Tracking colour objects using adaptive mixture models. Image and Vision Computing **17** (1999) 225 – 231
9. Raja, Y., McKenna, S., Gong, S.: Tracking and segmenting people in varying lighting conditions using colour. In: Proceedings of FG'98, Nara,Japan (1998)
10. Wren, C., Azarbayejani, A., Darrel, T., Pentland, A.: Pfinder: Real-time tracking of the human body. IEEE Transaction on Pattern Anaysis and Machine Intelligence **9** (1997) 780–785
11. Hartley, H.: Maximum likelihood estimation from incomplete data. Biometrics **14** (1958) 174–194
12. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via em algorithm. Journal of the Royal Statistical Society, Series B **39** (1977) 1–38
13. Schwarz, G.: Estimating the dimension of a model. Ann. of Stat. **6** (1978) 461–464
14. Kass, R., Raftery, A.: Bayes factors. Journal of American. Stat. Ass. **90** (1995) 773–795

Robust and Accurate Detection of Object Orientation and ID Without Color Segmentation

Shoichi Shimizu, Tomoyuki Nagahashi, and Hironobu Fujiyoshi

Dept. of Computer Science, Chubu University, Japan
shiyou@vision.cs.chubu.ac.jp, kida@vision.cs.chubu.ac.jp,
hf@cs.chubu.ac.jp
http://www.vision.cs.chubu.ac.jp/

Abstract. This paper describes a novel approach to detecting orientation and identity of robots without color segmentation. The continuous DP matching calculates the similarity between the reference pattern and the input pattern by matching the intensity changes of the robot markers. After the continuous DP matching, a similarity value is used for object identification. Correspondences of the optimal route obtained by back tracing are used for estimating the robot's orientation. This method archives orientation estimations of less than 1 degree and robustness with respect to varying light conditions.

1 Introduction

To give optimal visual-feedback, in order to control a robot, it is important to raise the robustness and accuracy of the vision system. Especially, in the RoboCup Small Sized League(F180), a global vision system that is robust to unknown and varying lighting condition, is needed. The vision system which has been generally used, processes an image to identify and locate robots and the ball. For low-level vision, the color segmentation library, called CMVision [1], has been used to perform color segmentation and to connect components analysis to return colored regions in real time without special hardware. After the color segmentation, the process of object identification is employed based on the results of the color segmentation which is then followed by the process of the pose estimation of the robot. To raise the robustness with respect to varying light condition, color calibration [2] need to be done in advance, but requires minimal set up time.

In this paper, we propose a robust and accurate pattern matching method for identifying robots and estimating their orientations simultaneously without color segmentation. Our approach uses continuous DP matching to search for similar pattern, which is obtained by scanning at a constant radius from the center of the robot. The DP similarity value is used for object identification, and for obtaining the optimal route by back tracing to estimate its orientation. We realized robustness of object identification with respect to varying light conditions by taking advantage of the changes in intensity only.

In the following, the related work and our approach are described in section 2. Section 3 describes the method for robust and accurate object identification.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 408–419, 2006.

The experimental results are shown in section 4. Section 5 discusses some of the advantages of the proposed method. Lastly, section 6 concludes the paper.

2 Related Work

In the Small Size League, one team must have a 50 mm blue colored circle centered on the top of its robot while the other team must have a 50 mm yellow patch. To detect the robot's orientation and to identify it, teams are allowed to add extra patches using up to three colors. Figure 1 shows examples of patterns found on the top of the robot.

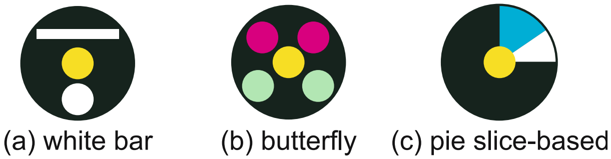


Fig. 1. Example of general identification patterns

Type (a), called "white bar", is used to calculate the pose of robot precisely. The robot's orientation is calculated using a white stripe and the least-squares method [3] or second-moment [4]. For identification, other sub patches are used.

Type (b), called "butterfly", has been reported in [5]. Geometric asymmetry can be used to find the rotational correspondence for orientation estimation.

Type (c), called "pie slice-based", is unique and is described in [6]. This method scans the circular pattern from markers on the robot. The angle resolution is not sufficient (8 degree) due to low resolution.

These methods use information from color segmentation to determine a robot's identity. Such colors have problems with brightness changes and non-uniform color intensity over the field, including sharp shadows.

2.1 Proposed Patch Pattern

Our approach uses only the changes in intensity obtained by scanning at a constant radius from the center of the robot and not by using the results of color segmentation. Therefore, we can paste suitably-colored patches on the top of the robot as shown in Figure 2. This makes a large number of different patterns for identification, and it's easy to modify patch patterns. Moreover, preparing the rule-based reference table by user for object identification is no longer necessary.

3 Object Identification

The DP matching calculates the similarity between reference pattern and input pattern by matching the intensity changes of the robot markers. After the DP

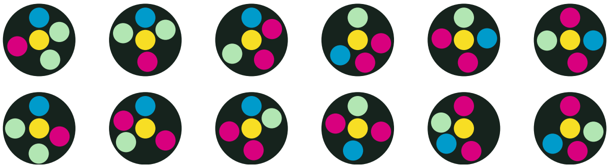


Fig. 2. Example of our ID plate

matching, a similarity value is used for identification. Correspondence of optimal route obtained by back tracing is used for estimating its orientation. The flow of the proposed method is as follows and shown in Figure 3:

1. Color conversion(RGB to YUV)
2. Detection of the center of blue/yellow colored circles
3. Converting to 1-dimensional signal by scanning at some constant radius from the center of the robot
4. Identifying our robots by continuous DP matching
5. Finding the robot's orientation by back tracing.

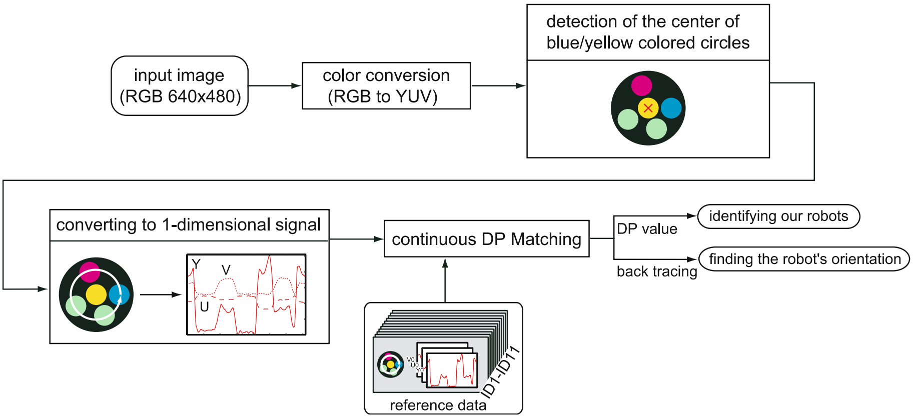


Fig. 3. Overview of our vision system

3.1 Detection of the Center of Blue/Yellow Colored Circle

It is important to detect the center of the blue/yellow colored circle because our approach uses this center position to convert to 1-dimensional signals for the object identification. The followings describe an algorithm for determining the center position of a circle given three points on a plane.

The three points determine a unique circle if, and only if, they are not on the same line. The relationship of these three points is expressed as:

$$(x_c - x_i)^2 + (y_c - y_i)^2 = (x_c - x_j)^2 + (y_c - y_j)^2 = (x_c - x_k)^2 + (y_c - y_k)^2 . (1)$$

where (x_c, y_c) is a center coordinate, three points on image are (x_i, y_i) (x_j, y_j) (x_k, y_k). Equation (1) is a linear simultaneous equation. Thus, (x_c, y_c) is determined by Gaussian elimination using the following steps:

Step1. Detect blue/yellow colored circle.
Step2. Extract contour points of the circle.
Step3. Select three points from contour points randomly, calculate center position (x_c, y_c) by equation (1).
Step4. Increment a count in the accumulator at point (x_c, y_c).
Step5. Step 3 and 4 are repeated 100 times.

Finally, the maximum number of votes is determined as the center of the main marker.

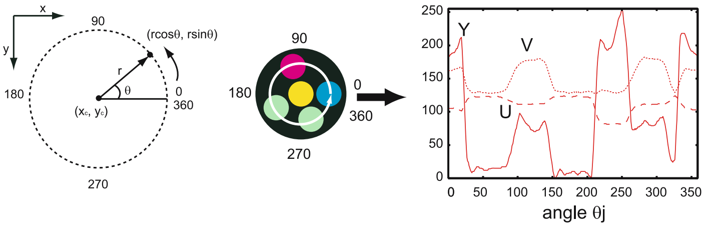
3.2 Converting to 1-Dimensional Signal

The intensity values of YUV on the top of the robot are obtained by scanning at a constant radius (r=10 pixel) from the detected center of the circle as shown in Figure 4. It is impossible to obtain the 359 points (1 degree each) on the circle's perimeter because of the low resolution image. To solve this problem, we apply the bilinear interpolation to estimate the robot's orientation with sub-pixel accuracy.

Image coordinate (x, y) for an angle θ is obtained by

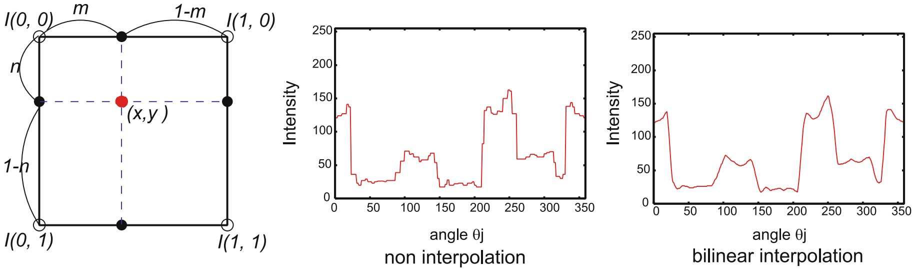
$$x = r\cos\theta + x_c, \quad y = r\sin\theta + y_c \tag{2}$$

where (x_c, y_c) is the center position on the image coordinate. Since the values of (x, y) are real numbers, the intensity value $I(x, y)$ is interpolated by the bilinear interpolation method used for 2-dimensional operations, for instance



(a) scanning at a constant radius (b) converting to 1-dimensional signal

Fig. 4. Converting to 1-dimensional signal



(a) bilinear interpolation (b) example of interpolation

Fig. 5. Bilinear interpolation

magnifying an image. The interpolated value in intensity is calculated as shown in Figure 5(a)

$$\begin{aligned} I(x,y) = (1-n)((1-m)I(0,0) + mI(1,0)) \\ + n((1-m)I(0,1) + mI(1,1)) \end{aligned} \tag{3}$$

Figure 5(b) shows the interpolated intensity values of Y. This can be useful to estimate the orientation angle with sub-pixel accuracy. Finally, the intensity values of Y normalized to 0∼255, U and V are obtained as a 1-dimensional signal from the circle patches on the robot as shown in Figure 4(b), and these values are expressed as:

$$I(\theta_j) = I(r\cos\theta_j, r\sin\theta_j) \;\; j = 0, \cdots, 359. \tag{4}$$

3.3 Identifying Our Robots by the Continuous DP Matching

To uniquely distinguish a robot, the intensity values $I(\theta_j)$ as a reference pattern for each robots are registered initially by clicking with the mouse of points in the direction of the robot's front help to assign an ID to each robot. The continuous DP matching is performed to calculate a similarity between the reference patterns and the input pattern of the current image.

Continuous DP matching. The DP matching has been used in various area such as speech-recognition [7]. DP matching is a pattern matching algorithm with a nonlinear time-normalization effect. Timing differences between two signal patterns are eliminated by warping the axis of one, so that the maximum coincidence is attached as the minimized residual distance between them. A starting point of input pattern provided by scanning described in section 3.2 is not at the same position as the reference pattern. Therefore, continuous DP matching can be useful in computing the similarity distance by considering the lag of each starting point. The input pattern is repeated twice as $(1 < i < 2I)$ and this handling is shown in Figure 6.

In this implementation, the symmetrical DP path, shown in Figure 7(a), is used. Minimum accumulated distance is calculated by the following equations.

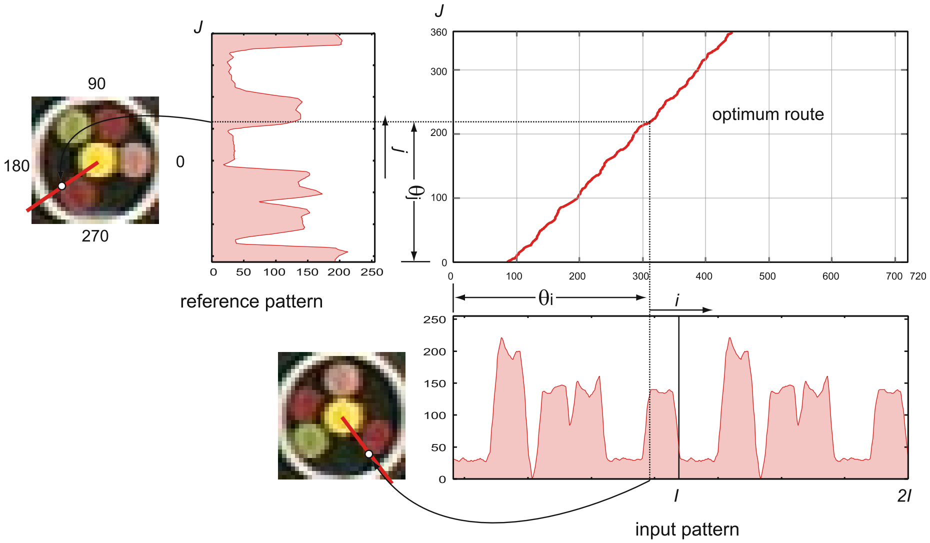


Fig. 6. Example of back tracing

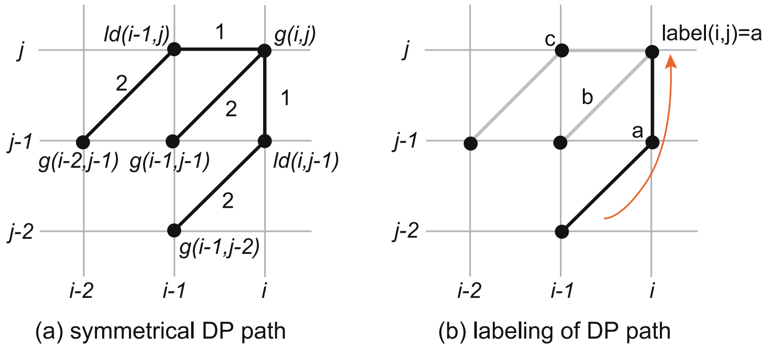


Fig. 7. Symmetrical DP path

Let the vertical axis represents reference pattern frame j, and the horizontal axis as input pattern frame i. Initial conditions are given as:

$$\begin{cases} g(i,0) = 0 & (i = 0, 1, \dots, I) \\ g(0,j) = \infty & (j = 1, 2 \dots, J) \end{cases} \tag{5}$$

where I and J are length of each patterns. The minimum accumulated distance $g(i,j)$ on the i frame and j frame are calculated by:

$$g(i,j) = \min \left\{ \begin{array}{ll} g(i-1,j-2) + 2 \cdot ld(i,j-1) & : (a) \\ g(i-1,j-1) + ld(i,j) & : (b) \\ g(i-2,j-1) + 2 \cdot ld(i-1,j) & : (c) \end{array} \right\} + ld(i,j). \tag{6}$$

Local distance $ld(i,j)$ on the point of (i,j) is computed as:

$$ld(i,j) = (I_t(\theta_i) - I_{t-1}(\theta_j))^2. \tag{7}$$

The length for the optimal route:

$$c(i,j) = \begin{cases} c(i-1,j-2)+3 \middle| if(a) \\ c(i-1,j-1)+2 \middle| if(b) \\ c(i-2,j-1)+3 \middle| if(c) \end{cases} \tag{8}$$

is used to obtain the normalized accumulated distance by:

$$G(i) = \frac{g(i,J)}{c(i,J)}. \tag{9}$$

Object ID recognition. The continuous DP matching is performed to calculate similarity distances for each reference pattern, when a blue/yellow circle of the robot is detected. The identity of the robot is determined by selecting the reference pattern which is given the minimum value of G.

3.4 Object Orientation Estimation by Back Tracing

To detect the robot's orientation, back tracing, which computes local corresponding points of input and reference patterns by referring the selected DP path, is performed as follows:

1. DP matching and labeling of the selected DP path
   While computing the minimum accumulated distance, the path selected by equation (6) is memorized with label a/b/c as shown in Figure 7(b).
2. Back tracing
   After normalizing minimum accumulated distance, the minimum value of $G(i,J)$ is selected as a starting point for the back tracing.

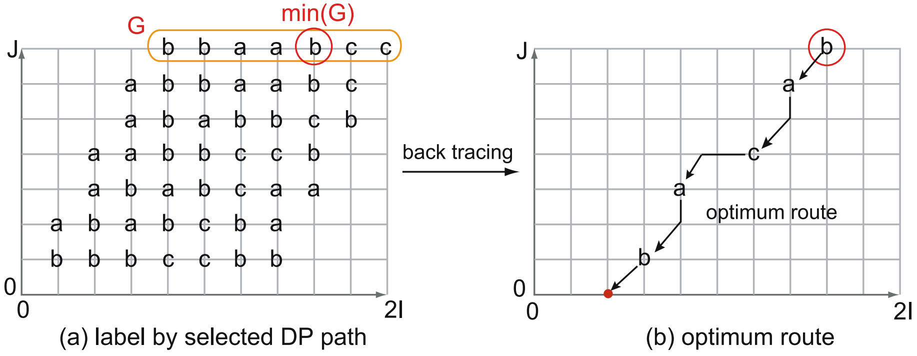
   $$i' = argmin_{(J/2 \leq i \leq 2I)} G(i,J) \tag{10}$$

   The optimum route is tracked by referring to the label, either 'a', 'b', or 'c' at each node. The DP path labeled 'a' means insert, and 'c' means delete. The path 'b' means that frame i and j are a pair of corresponding point. When path 'b' appears on the optimum route, the orientation of the current robot θ is estimated by:

   $$\theta = \theta_i - \theta_j \tag{11}$$

   where θ_i is the orientation angle of input pattern, and θ_j is reference pattern. This process is finished when the route, by back tracing, reaches the end point($j = 0$), and then the average of the angle θ points at the robot's orientation(front direction).

As we mentioned above, object orientation and ID are determined by the continuous DP matching and not color segmentation.



(a) label by selected DP path

(b) optimum route

Fig. 8. Back tracing

4 Experimental Results

The performance of their proposed method was computed in simulation as well as real experiments with regard to robustness and accuracy in varying light conditions.

4.1 Experimental Results of Orientation Estimation

Results in simulation experiments. To determine the accuracy of the orientation estimation, the estimated angle using the proposed method is compared to ground truth. Table 1 shows the results in simulation experiments evaluating 360 patterns (1 degree each). In comparison to the performance of general methods based on the least-squares method [3] and the second-moment method [4] using "white bar" ID plate, our method has better accuracy in orientation estimation.

The accurate center position of the blue/yellow colored circle for main marker can not be obtained, when the circle's perimeter has noise. In this case, we evaluate the robustness of our method using the pattern in which the center position of the circle translate to its neighbors. The noise 1 in Table 1 is an area of 3x3 pixels except for the center. The noise 2 is an area of 5x5 pixels except for the center and the noise 1. Five pixels represent 25 millimeters. The SSD in Table 1 means linear matching using the sum of squared difference to estimate the orientation. The SSD is better than proposed method when a very accurate center position (noise 0) is obtained. However, our method is effective with respect to errors in the center position of the circle because the DP warping function can obtain the optimum correspondence against the gap.

Results in real experiments. Table 2 shows results in experiments using the real vision system, in which a camera is mounted at a height of 4,000 mm. We can see that our method has almost the same performance as the general method, and it works well with respect to the "white bar". This shows that our method can obtain the direction of opponent robot's front, and this information is useful to intercept the passing ball.

Table 1. Average of absolute errors of orientation estimation in simulation experiments [degree]

	proposed method		general method	
noise	SSD	DP	least-squares method	second-moment
0	0.30	0.76	0.85	1.08
1	1.71	1.10	-	-
2	4.20	1.75	-	-

Table 2. Average of absolute errors of orientation estimation in real experiments [degree]

	proposed method	least-squares method	second-moment
white bar	0.85	1.17	0.96
patch pattern	0.95	-	-

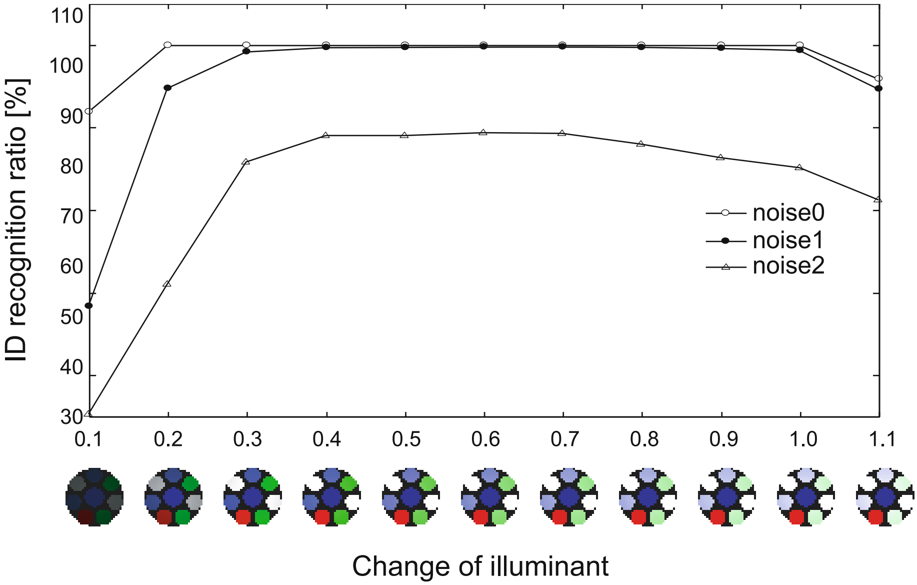


Fig. 9. Result of ID recognition in simulation experiments

4.2 Experimental Results of Object Identity

Results in simulation experiments. To determine the robustness with respect to varying the light condition. A model of illuminant and the marker are created by CG, the pixel intensity of the input image is created by changing the illuminant. Figure 9 shows ID patterns under illuminant changes in the simulation and identification performance against the 11 unique robots. Our system is performance is stable against the change in lighting conditions. However, the recognition ratio decreases at the noise 2. When the error of the center is two

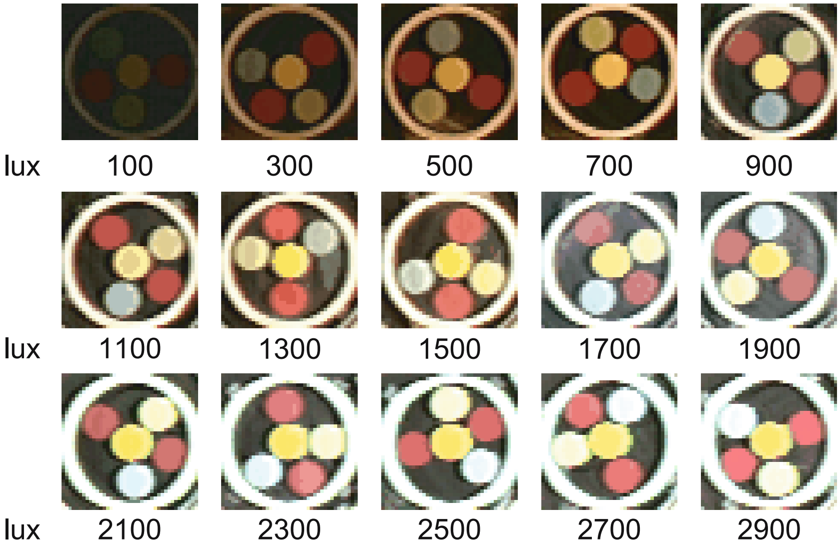


Fig. 10. Images captured under the illuminance of ranging from 100 to 2,900 lux

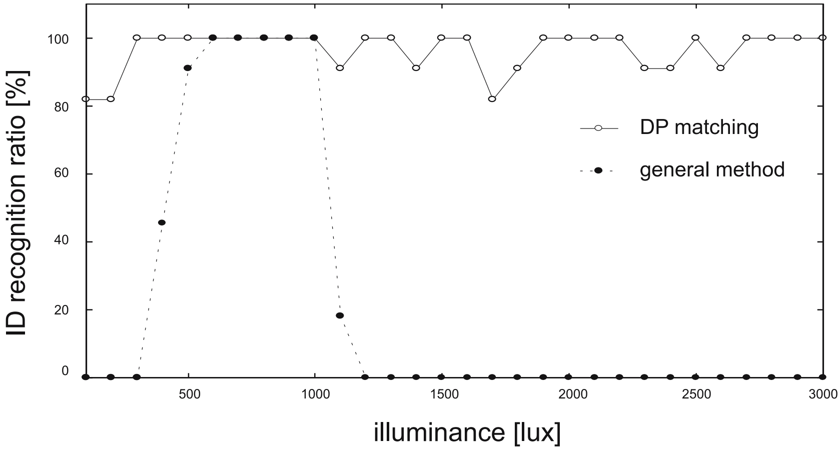


Fig. 11. Result of ID recognition in real experiments

pixels, the center is near the edge of the main marker. Therefore, it is difficult to calculate the 1-dimensional signal, and the recognition ratio decreases.

Results in real experiments. Figure 10 shows images captured under the illuminance ranging from 100-to-2,900 lux. In the experiment, we evaluate 330 images for 11 unique IDs with varing light condition(100∼3,000 lux). Figure 11 shows object identification ratios for 330 images. Note that the general method means color segmentation based object identification adjusting the threshold to obtain high performance for lighting condition between 600 to 1,000 lux. On the other hand, for reference patterns of our method, only the images captured under the light of 1,000 lux are registered. It is clear that our method has a better performance with respect to varying light conditions, because our approach is not based on color segmentation rather it is based on matching using changes in intensity obtained by scanning at a constant radius from the center of the robot.

5 Discussion

This section describes some of the benefits of the proposed method.

- **Easy handling for set up**
  In order to register reference patterns for each robot's ID, the orientation is obtained by clicking the front of the robot to assign an ID for each of our robots. There is no need for making rule-based reference table for object identification.
- **Easy to modify the patches**
  Since the white bar is used to estimate the robot's orientation in general method, the area for pasting more patches of sub-markers is restricted. However, our method allows for more space on the top of the robot. Moreover, it is very easy to modify the patch pattern because of its easy set up.
- **Robustness with respect to varying light conditions**
  There is no perfect color segmentation. Even if the lighting conditions are changed by meteorological effects, our method can work well because the changes in intensity are used for detecting a robot's orientation and identity.
- **Obtaining direction of opponent robot**
  Our method for estimating the robot's orientation works well to any shaped-patch patterns such as “white bar”. Therefore, it is possible to know the direction of the opponent robot's front. This means our robot can intercept the ball passing between the opponent robots.

The demerit of the proposed method is as follows. Our method is converted to a 1-dimensional signal. Therefore, if the center of circle can't be calculate accurately, it is difficult to convert to it to a 1-dimensional signal accurately. To calculate accurately, the object identity and orientation is necessary to suppress the error of the center position within two pixels. Moreover, in the case of estimating the ID, it is necessary to compare between input pattern and all reference patterns. Therefore, as the number of robots increase the computational cost is increases.

6 Conclusion

This paper describes a novel approach for detecting orientation and identity of robots without color segmentations. We show that the proposed method achieves accurate performance in orientation estimation, in comparison to the general method, as well as its robustness with respect to varying light conditions. The system using the proposed method runs in real time on a Xeon 3.0GHz, PC, as such the system can be completely setup in a short amount of time by a single operator.

Future works will focus on more automation in the registration procedure of reference patterns.

Acknowledgement

This research was partly supported by agrant from the High-Tech Research Center Establishment Project from Ministry of Education, Culture, Sports, Science and Technology.

References

1. J. Bruce, T. Balch, M. Veloso: "Fast and Inexpensive Color Image Segmentation for Interactive Robots", In Proceedings of IROS-2000, Japan, 2000.
2. A.Egorova, M.Simon, F.Wiesel, A.Gloye, and R.Rojas: "Plug & Play: Fast Automatic Geometry and Color Calibration for Cameras Tracking Robots", Proceedings of ROBOCUP2004 SYMPOSIUM, 2004.
3. K.Murakami, S.Hibino, Y.Kodama, T.Iida, K.Kato, S.Kondo, and T.Naruse: "Cooperative Soccer Play by Real Small-Size Robot", RoboCup 2003: Robot Soccer World Cup VII, Springer, pp. 410-421, 2003.
4. Ball D., Wyeth G., Nuske S: "A Global Vision System for a Robot Soccer Team", Proceedings of the 2004 Australasian Conference on Robotics and Automation (ACRA), 2004.
5. J.Bruce, M.Veloso: "Fast and Accurate Vision-Based Pattern Detection and Identification", Proceedings of ICRA-03, the 2003 IEEE International Conference on Robotics and Automation (ICRA'03), May, 2003.
6. S.Hibino, Y.Kodama, Y.Nagasaka, T.Takahashi, K.Murakami and T.Naruse: "Fast Image Processing and Flexible Path Generation System for RoboCup Small Size League", RoboCup 2002: Robot Soccer World Cup VI, Springer, pp. 53-64, 2002.
7. H. Sakoe, S. Chiba: "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", IEEE Trans. Acoust., Speech, and Signal Process., Vol.ASSP-26, pp. 43-49, 1978.

A Fuzzy Touch to R-MCL Localization Algorithm

Hatice Köse and H. Levent Akın

Boğaziçi University
Department of Computer Engineering
34342 Bebek, Istanbul, Turkey
{kose, akin}@boun.edu.tr

Abstract. In this work, a novel method called Fuzzy Reverse Monte Carlo Localization (Fuzzy R-MCL) for global localization of autonomous mobile agents in the robotic soccer domain is proposed to overcome the uncertainty in the sensors, environment and the motion model. R-MCL is a hybrid method based on both Markov Localization(ML) and Monte Carlo Localization(MCL) where the ML module finds the region where the robot should be and MCL predicts the geometrical location with high precision by selecting samples in this region. In this work, a fuzzy approach is embedded in this method, to improve flexibility, accuracy and robustness. In addition to using Fuzzy membership functions in modeling the uncertainty of the grid cells and samples, different heuristics are used to enable the adaptation of the method to different levels of noise and sparsity. The method is very robust and fast and requires less computational power and memory compared to similar approaches and is accurate enough for high level decision making which is vital for robot soccer.

Keywords: Global localization, ML, MCL, Fuzzy logic, Robot soccer.

1 Introduction

The localization problem is the estimation of the position of a robot relative to the environment, using its actions and sensor readings. Unfortunately the sensors and the environment are uncertain, so the results are typically erroneous and inaccurate. From the simplest geometric calculations which do not consider uncertainty at all, to statistical solutions which cope with uncertainty by applying sophisticated models, many solutions have been proposed [1, 2, 3]. Although some of these approaches produce remarkable results, due to the nature of the typical environments they are not satisfactory. Generally, solutions producing precise results suffer from slowness, and high memory usage. Whereas a fast solution in practice typically produces only coarse results. Even when they produce precise local results, some approaches like Kalman filters, fail to find the global position. Consequently, localization still remains as a nontrivial and challenging problem.

In robot soccer, a robot is typically expected to find its own location using the distinguishable unique landmarks in the field, and then use this information

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 420–427, 2006.

to find the location of the ball and goal. For such a real-time application with robots limited by on board computational resources, fast solutions with less memory and computational resources are especially demanded. Consequently, localization is a vital problem for robot soccer. This work is a part of the Cerberus Team Robot soccer project [4], and aims to localize the legged robots in the soccer field globally, while solving the above mentioned problems. There are a several limitations and assumptions related to the rules of the Four Legged League of Robocup [5]. In this work, the previously developed hybrid approach called *Reverse Monte Carlo Localization*(R-MCL) [6, 7] combining the ML and MCL methods is extended by using fuzzy sets to improve success in case of high sparsity and noise.

The organization of the paper is as follows: In the second section, a brief survey of localization methods is presented. In the third section detailed information R-MCL algorithm can be found. In the fourth section, the fuzzy extension to R-MCL is presented. The results of the application of proposed approach are given in section five. In the sixth section, conclusions and suggestions for future work are given.

2 Localization Methods

The simplest localization method depending on the range and bearing data is triangulation, which uses geometry to compute a single point that is closest to the current location. But in real world applications a robot can never know where it is exactly because of the uncertainty in its sensors, and the environment. Consequently, several different approaches which estimate the position of robot probabilistically were introduced to integrate this uncertainty into the solutions.

Kalman filter (Kalman-Bucy filter) is a well-known approach for this problem. This filter integrates uncertainty into computations by making the assumption of Gaussian distributions to represent all densities including positions, odometric and sensory measurements. Since only one pose hypothesis can be represented, the method is unable to make global localization, and can not recover from total localization failures [8, 9, 3].

Many works consider Markov localization (ML) [1, 10]. ML is similar to the Kalman filter approach, but it does not make a Gaussian distribution assumption and allows any kind of distribution to be used. Although this feature makes this approach flexible, it adds a computational overhead.

Monte Carlo Localization (MCL) is a version of Markov localization that relies on sample-based representation and the sampling/importance re-sampling algorithm for belief propagation [2, 11]. Odometric and sensory updates are similar to ML. Most of the MCL based works suffer from the kidnapping problem, since this approach collapses when the current estimate does not fit observations. There are several extensions to MCL that solve this problem by adding random samples at each iteration. Some of these methods are Sensor Resetting Localization (SRL), Mixture MCL (Mix-MCL), and Adaptive MCL (A-MCL). In SRL, when the likelihood of the current observation is below a threshold, a small fraction of uniformly distributed random samples is added [12]. Mix-MCL additionally weights these samples with current probability density. This method has been developed for extremely accurate sensor information [3]. Adaptive MCL only adds samples when the difference between short-term estimate (slow changing noise level in the environment and the sensors) and the long-term estimate (rapid changes in the likelihood due to a position failure) is above a threshold. [3].The MHL method discussed in [13] aims to avoid caused by using a single Gaussian, by using a mixture of Gaussians enabling the representation of any given probability distribution of the robot pose.

ML-EKF method is a hybrid method aiming to make use of the advantages of both methods, taking into consideration the fact that ML is more robust and EKF is more accurate [3].

Although there have been only a few fuzzy logic based approaches, they appear to be promising [14, 15]. In these approaches, the uncertainty in sensor readings (distance and heading to beacons) is represented by fuzzy sets.

3 R-MCL Method

As mentioned in section 2, ML is robust and converges fast, but is coarse and computationally complex. On the other hand, sample based MCL is not as computationally complex as ML, and gives accurate results. However, it can not converge to a position as fast as ML, especially in the case of an external impact on the position of the robot (such as kidnapping). In addition, the number of samples to be used is generally kept very high to cover all the space and converge to the right position. Several extensions have been made for adaptive sample size usage, but these still do not solve the slow coverage problem. The Reverse Monte Carlo Localization algorithm [6, 7] was developed to benefit from the advantages of these two methods while avoiding their disadvantages. The idea is to converge to several cells by ML or another grid based method, then produce a limited number of samples inside these bulk of grids to find the final position. The average of these samples would give the final position and the standard deviation might give the uncertainty of the final position as in the MCL based methods. In the original MCL, the number of samples is increased to decrease the bias in the result. In R-MCL since we converge by selecting the cells with maximum probability, so the bias is already decreased. The R-MCL algorithm is given in Figure 1.

In this work, some improvements were done on the R-MCL method and in particular the ML module. In the modified ML, not only the distance but also the bearing information is used to find the best grids, so the number of chosen grids decrease and confidence increases. When the samples are drawn, also the best samples are selected using distance and the bearing from these very good cells, and their average is returned as the current pose. Note that samples are taken into consideration only when the position is above a certainty level, in other words the number of chosen cells are below a limit(e. g. 50), and there is at least one very good cell which is below or equal to the minimum error in

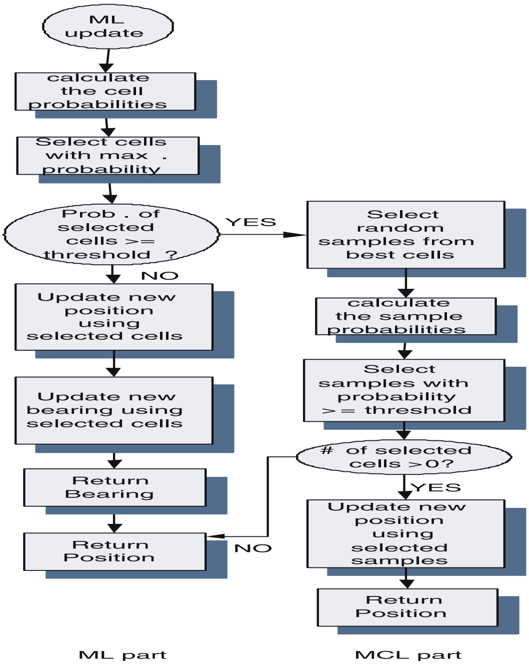

ML update
calculate the cell probabilities
Select cells with max . probability
Prob . of selected cells >= threshold ?
YES
NO
Update new position using selected cells
Update new bearing using selected cells
Return Bearing
Return Position
Select random samples from best cells
calculate the sample probabilities
Select samples with probability >= threshold
# of selected cells >0?
YES
NO
Update new position using selected samples
Return Position
ML part
MCL part

Fig. 1. The R-MCL flowchart

both distance and bearing limitations. Also if there are no samples which satisfy the minimum bearing and distance error condition then the results of ML are used instead. The bearing of the new pose is found by the ML module inside the R-MCL because it is more accurate and robust.

4 Fuzzy R-MCL

In this work, after improving R-MCL, a fuzzy approach was embedded in it, to improve flexibility, accuracy and robustness. Fuzzy membership functions are used in modeling the uncertainty of the grid cells and samples. Here, the uncertainty model μ_1 which is used in both ML and R-MCL is replaced by the fuzzy model μ_2 represented by the fuzzy membership function given in Figure 2(b). The previous model was simple and fast but it was not flexible enough to improve success when sparsity and noise is high. Especially if the cell size is kept high (30 cm) as in [6, 7] compared to 5 cm used in [3] a more flexible model is needed to weight the probability of being in that cell. It is not preferable to give the same weight to every point when the cells sizes are so big, and to the samples inside these cells.

In both of the models given in Figure 2, d_i represents the difference between the observed relative distance from robot to the currently observed landmark, and the calculated distance from the current cell center to the currently observed

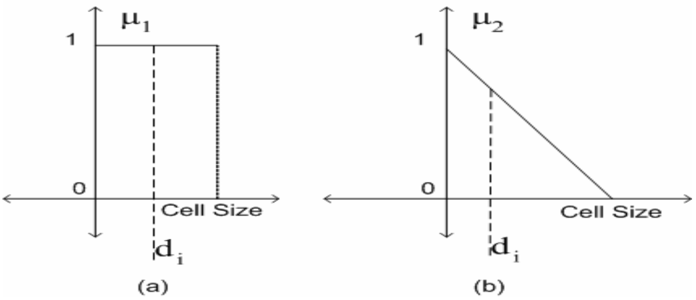


Fig. 2. Fuzzy membership functions used

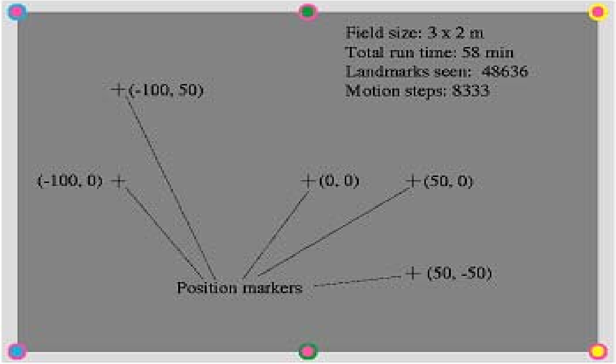


Fig. 3. The test field

landmark. This enables us to weight the samples according to their fitness to the observation and odometry.

5 Tests and Results

In the testing phase, a standard set of test data is used. These data are based on the records of the test runs of Sony's ERS 210 quadruped robots (AIBO) on the Robocup soccer field, used in [3] for comparison of several well-known localization methods in literature. These are produced by running the robot on the field as shown in Figure 3 on an eight like path for almost an hour, stopping the robot on several predefined points called markers, and recording the observations and odometry readings during this run. The tests aim to analyze accuracy and robustness in case of noisy and sparse data, and the ability to handle kidnapping problem. In the noisy data tests, randomly chosen data are replaced by noisy ones. The number of noisy samples is increased to see the robustness and accuracy of observed methods in case of high noise levels. In sparse data tests, samples are deleted from the tests, in a predefined sequence, (beyond the robots awareness). As the frequency increases, the behavior of the selected methods is

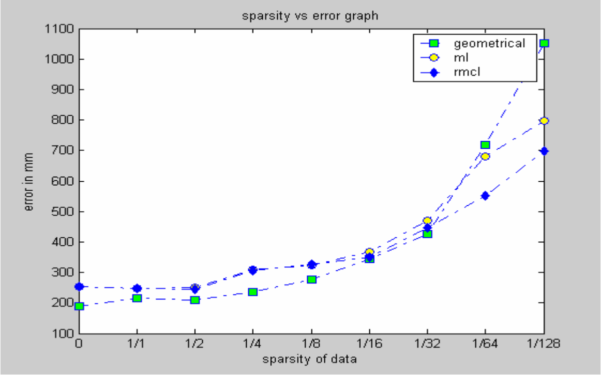


Fig. 4. The results for the sparse data case

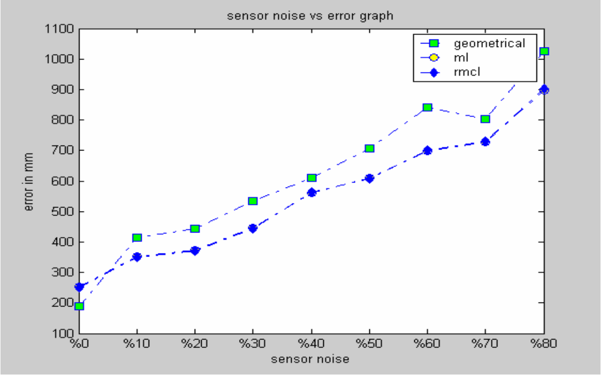


Fig. 5. The results for the noisy data case

observed. Lastly, the kidnapping problem is tested, by changing the position of the robot (beyond the robots awareness), and the time for recovery is recorded.

Several different kinds of membership functions (e. g. trapezoidal) and different sizes (e. g. twice the cell size) were tested and the best model found is the model μ_2 presented in the Figure 2(b). In Figure 4 and Figure 5, the accuracy of Fuzzy-RMCL and other proposed methods are presented, in case of sparse and noisy data, respectively. The results of the previous versions of the proposed methods can be found in [6, 7]. The error rates of the tests are calculated from the expected location of robot, when it reaches a marker, and its exact location. Note that, there are also unavoidable errors in the exact locations of robot due to experimental problems reported by the data providers. When the results of the tests are compared to the similar tests in [3, 13], the R-MCL method shows similar performance in the sparse and noisy data tests. The cell size is chosen as 5 cm in the referenced works, but it is taken as 30 cm in the current work, to increase the speed, and triangulation method which is used in the case of observing two or more landmarks are not used in the implementations. These facts would decrease the error rate very much, but the current case is more realistic and similar to real world case. The parameter set is chosen after detailed tests and comparisons.

Note that in the case of sparse data, Fuzzy R-MCL outperforms ML especially when sparsity is more since it is logical to gain more information to locate the robot by throwing more samples as the sparsity increases. However, as the noise increases it is not logical to throw more samples, but to keep the number of cells as small as possible to cope with noise.

During the tests, it was realized that choosing the cell size very big decreases the effect of odometry, and can cause a *temporary kidnapping* effect when the robot moves from one cell to other, which also decreases the success rate.

6 Conclusions

Localization in a totally unknown area is a very hard task for autonomous mobile robots. This work aims to propose a fast, reliable, computationally and resource efficient solution to the global localization problem. The solution should be successful in environments like the Robocup Games and the challenges which require very high accuracy and speed. For this reason previously a hybrid method called R-MCL method was developed. In this work, a fuzzy approach is embedded in this method, to improve flexibility, accuracy and robustness. In addition to using Fuzzy membership functions in modeling the uncertainty of the grid cells and samples, different heuristics are used to enable the adaptation of the method to different levels of noise and sparsity. The method is very robust and fast and requires less computational power and memory compared to similar approaches and is accurate enough for high level decision making which is vital for robot soccer. There is an ongoing research for improving the success by solving the temporary kidnapping problem due to large cell size.

Acknowledgments

This work was supported by Boğaziçi University Research Fund project 03A101D. We thank Stephen Gutman and Dieter Fox for kindly providing the test data.

References

1. W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids," *Institut fr Informatik III. , Universtiat Bonn, Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.
2. S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo Localization for Mobile Robots," *Artificial Intelligence,* 128, pp. 99-141, 2001.
3. J. S. Gutmann, and D. Fox, "An Experimental Comparison of Localization Methods Continued," *In Proc. of the 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'02),* pp. 454-459, 2002.

4. Cerberus, Team Web Site *http://robot. cmpe. boun. edu. tr/aibo/home.php3,* 2005.
5. Sony Four-Legged Robot League, "Soccer Rules", *http://www.tzi.de/4legged/pub/Website/History/Rules2005.pdf,* 2005.
6. H. Kose, and H. L. Akin, "Robots from nowhere", *RoboCup 2004: Robot Soccer World Cup VIII*, Daniele Nardi, Martin Riedmiller, Claude Sammut, et al. (Eds.), LNCS, Vol. 3276, pp.594-601, 2005.
7. H. Kose, and H. L. Akin, "Experimental Analysis and Comparison of Reverse-Monte Carlo Self-Localization Method", *Proceedings of CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby,* Vienna, Austria, December 2 - 4, 2004,
8. A. W. Stroupe, T. Balch, "Collaborative Probabilistic Constraint Based Landmark Localization," *Proceedings of the 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems IROS'02),* pp. 447-452, 2002.
9. A. W. Stroupe, K. Sikorski, and T. Balch, "Constraint-Based Landmark Localization," *RoboCup 2002: Robot Soccer World Cup VI.* LNCS 2752, pp.8-24, 2002.
10. D. Fox, W. Burgard, and S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments" *Journal of Artificial Intelligence Research*, 11, pp. 391-427, 1999.
11. D. Schulz, and W. Burgard, "Probabilistic State Estimation of Dynamic Objects with a Moving Mobile Robot," *Robotics and Autonomous Systems*, 34, pp. 107-115. 2001.
12. S. Lenser, and M. Veloso, "Sensor Resetting Localization for Poorly Modelled Mobile Robots" *Proceedings of ICRA 2000, IEEE,* pp. 1225-1232 2000.
13. S. Kristensen and P. Jensfelt, "An Experimental Comparison of Localisation Methods, the MHL Sessions", *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03),* pp.992-997 , 2003.
14. P. Buschka, A. Saffiotti, and Z. Wasik, "Fuzzy Landmark-Based Localization for a Legged Robot" *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS),* Takamatsu, Japan, Page: 1205-1210, 2000.
15. H. Kose, S. Bayhan, and H. L. Akin, "A Fuzzy Approach to Global Localization in the Robot Soccer Domain" *IJCI Proceedings of International XII Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2003),* ISSN 1304-2386 Vol:1, No:1, pp.1-7, July 2003.

A New Practice Course for Freshmen Using RoboCup Based Small Robots

Yasunori Nagasaka[1], Morihiko Saeki[2], Shoichi Shibata[3], Hironobu Fujiyoshi[3], Takashi Fujii[3], and Toshiyuki Sakata[2]

[1] Dept. of Electronics and Information Engineering
[2] Dept. of Mechanical Engineering
[3] Dept. of Computer Science,
Chubu University College of Engineering, Japan
any@nn.solan.chubu.ac.jp, saeki@isc.chubu.ac.jp, shibata@cs.chubu.ac.jp, hf@cs.chubu.ac.jp, fujii@cs.chubu.ac.jp, tosakata@isc.chubu.ac.jp

1 Introduction

Contemporary engineers need to have the ability not only to freely make use of their professional knowledge and skills, but also to integrate and combine a wide range of knowledge and skills and to build a complex system to solve a problem. But the current educational programs of individual departments (mechanical engineering, electrical engineering, electronic engineering, computer science) are usually designed and performed independently. Therefore it is hard for students to understand how knowledge and technologies of each field are integrated and combined in the objects of the real world. In order to increase student understanding in this area, we propose a new practice course dealing with a completely functional object: a robot.

There are several experiments and practice courses dealing with robots as educational materials. LEGO MINDSTORMS is sometimes used for an introductory course and is acknowledged as efficient educational material [1],[2]. There are also some trials to introduce RoboCup based robots to education in practice [3]–[7], and RoboCup soccer simulation [8],[9]. The first steps in the process of building a robot are more difficult for students than using LEGO MINDSTORMS, because students need to master such tasks as the mechanical process, soldering, and programming. But once they have mastered such tasks, they are better able to try more advanced tasks like adding different type of sensors later.

The Owaribito-CU robot soccer team has competed in RoboCup since 1999. Along the competitions, robots for RoboCup small size league have been developed in one of the extracurricular activities. This year faculty of the team have introduced the construction of a simplified robot to the regular curriculum of the Chubu University college of engineering.

This paper describes the outline of a newly designed practice course for undergraduate engineering students, especially for freshmen. The process and review of the course, which was carried out as five day intensive class in summer vacation, and the analysis of the questionnaire survey for the students after the course are mentioned. Our new course is close to Baltes [4], Anderson [5], and

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 428–435, 2006.

DME project in MIT [7] in previous trials. Distinctive features of our course are the realization of inter-departmental education for freshmen without prior knowledge and skills, and compact course work for a short term intensive class.

The course was open to all undergraduate students. It covers not only themes dealing with hardware and software, but also topics in system engineering and the wide range of knowledge and technologies related to several fields. By introducing such a course into the early stages of undergraduate education, we hope to stimulate students to become interested in other fields besides their own specialized fields.

2 A Basic Robot for the Course

Currently we Owaribito-CU team have three wheeled omni directional robots, ball handling devices, and a multi camera vision system. Fig 1 (left) shows our omni-directional robot for the 2004 competition.

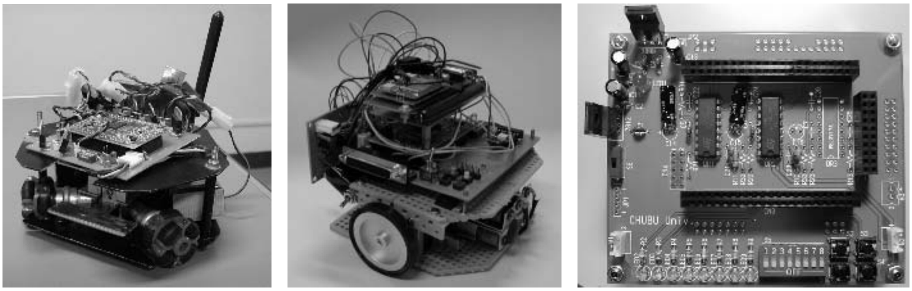


Fig. 1. 2004 omni-directional moving robot for competition (left), prototype of simplified robot (center) and main board (right)

This robot contains expensive motors and a wireless communication board. And it is therefore not suitable for the practice course. The robot which will be constructed by students in the course is a simplified robot. Fig 1 (center) shows the trial piece of the simplified robot. It is easy to construct, even for beginners. Another merit of the basic robot is that it can be constructed at low cost. Already there is a trial to introduce a low cost robot to education in CMU [10].

Our simplified robot for educational use can be constructed for about 330 US dollars per robot including wireless communication function. Required parts (and their cost) are the main board (30 US dollars), wireless communication (90), H8 micro computer (30), motor drive circuit parts (30), DC motor (60), chassis parts (20), tire and wheel (30), battery pack (30), other miscellaneous parts (10). We think total cost is an important factor as educational material, because 30 or more robots are constructed in a course. Keeping costs as low as possible is important in order to continue offering the course in the future.

All parts except the main board are ready-made. Only the main board was newly designed for the course. Fig 1(right) shows the main board. It includes the power supply circuit, motor driver IC, interface to H8 MPU board, switches and LEDs.

Since the construction work is carefully chosen and combined, the robot can be constructed easily within 30 hours by an undergraduate student without any prior knowledge of the robot design and architecture. Once the students experience constructing the robot by themselves, they will have a sense of achievement through the process.

3 A New Practice Course Using RoboCup Based Small Robots

3.1 Aims and Merits of the Course

The aim of the new course is to introduce the fundamental technologies of various engineering fields to freshmen. The course will be a typical example to show how the knowledge and the technologies of many fields are actually applied to build a complex system like a robot. It is expected that students are encouraged to be interested in their field and even non-specialized fields. As another feature of the course, inter-departmental education is realized. Teams are organized by the students from different departments. Making such team produces the chance to communicate with those from other departments, and encourages students' interest in other fields.

Some of the achievements of this course are as follows.

(1) Students can understand the basics, such as the architecture of the robot, driving mechanism, control circuits and programs.
(2) Students can understand how to use various tools.
(3) Students can devise some part of the robot by themselves with their own ideas.
(4) Students have fun constructing a robot and develop their interest in topics in engineering. Students encourage their will to study related subjects.
(5) Students can gain confidence in constructing a robot by themselves.

3.2 Syllabus

Course plan
The course is completed in a five day intensive class with an orientation where a prior explanation of the syllabus is given. We plan to divide the class into teams of students belonging to different departments, and each team constructs its own soccer robot. In order to give an incentive to learn with enjoyment and to construct the robots with interest and enthusiasm, a competition using produced soccer robots is held on the last day of the course.

Schedule
The course is carried out as an appropriate combination of lectures and practices. In order to quickly put into practice the knowledge and information provided

during lectures, the lecture class is assigned in the first half of the day and the practice class is assigned in the latter half. Concerning the first 3 days, the order of contents described below might be re-arranged in rotation for some student teams, because of the size of a class and the limitation of the work place.

the first day
(lecture) Dynamics of Mechanical parts (such as motors, wheels, gears). Behavior of the driving part. How to use machine tools.

(practice) Machine design (shape of chassis, position of electric motor). Evaluation of driving part (regarding speed or torque). Practice using machine tools (cutting out chassis, drilling, bending, smoothing). Test run.

the second day
(lecture) Motor driver IC and circuit. PWM method. Wireless communication. H8 micro processor (A/D converter, D/A converter, peripheral interface). Battery and power supply. How to use tools. Soldering. How to test the circuit.

(practice) Construction of controller circuit. Test operation.

the third day
(lecture) Control program. Programming for wireless communication. USB communication. Process of software development.

(practice) Understanding and tuning the control program. Improvement of the user interface (USB controller). Trial operation.

the 4th day
Construction of robots. Preparation of presentation.

the 5th day
Construction of robots. Final tuning. Team presentation. Competition.

Besides the themes of lecture and practice in the course, it will be possible to include related themes like experiments in logic circuit, how to use an oscilloscope, assembler programming, and so on. In those themes, image processing and AI are especially important research and development themes of RoboCup. Though these themes should be included in the course, they are omitted at present due to time limitations.

Evaluation
Students will be evaluated by writing a paper concerning the skills and knowledge they have learned in the construction process, by completeness and originality of the robots (body, mechanism, circuits, program, and design) they have constructed, by expressiveness of the presentations they have given, and the results they have achieved at the competition. All points mentioned above are individually graded, and the total of them will be used for the evaluation. Students' attitude in the class will be also evaluated in some cases.

3.3 From Preparation to Final Competition

The course textbook was written by faculty and teaching assistants. While the contents of the textbook were limited to only related topics, it had more than

120 pages finally. The textbook was printed and bound prior to the beginning of the course and distributed to participants.

Orientation was held prior to the course to explain the contents of the course and to invite students to the course. All students in the college of engineering can participate this course, but the themes of each day listed in the syllabus are designed mainly for students of the department of mechanical engineering, electrical engineering, electronics and information engineering, and computer science. 81 students registered, while upper limit of participants was 90. Only a few students from outside the above four departments registered.

The course started on August 2nd. To have enough time, the class started at 9:30 and finished at 18:20 everyday. The final day of the course was also the University's "Open Campus" day. Therefore we opened the work room in the morning and the presentation and the competition in the afternoon to the public. Many high school students showed up, and we could make an appeal to them. Fig 2 shows the presentation and competition. Fig 3 shows the examples of robots which students constructed. The parts they used were the same, but the appearance of the robots was different for each teams.

There were some minor problems during construction, but finally all constructed robots worked well. All participants and robots attended the final competition. After the competition, the members of the champion team received a commendation and were given "The Dean of College Cup". They also earned the right to enter "The President Cup" competition in autumn.

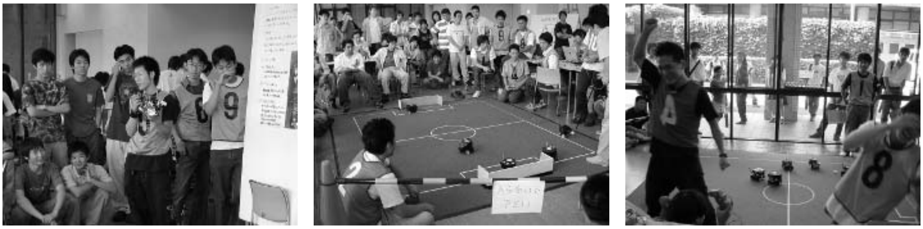


Fig. 2. Presentation and competition

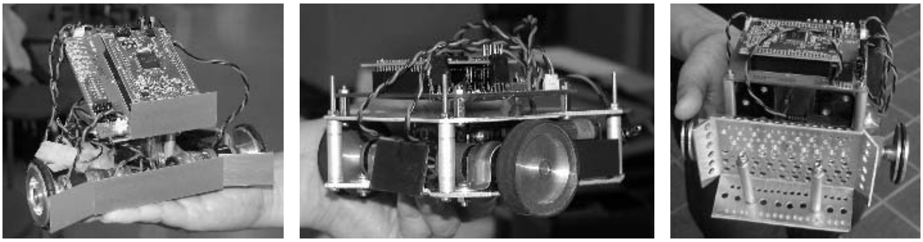


Fig. 3. Examples of robots which students constructed

3.4 Support for Completed Students

After the course, an opportunity to improve their robots is provided for students completing the course. In order to motivate their will to study, we also have "The President Cup" at the university festival in autumn. To the teams who have achieved excellent results in the President Cup, financial assistance for competing in the domestic RoboCup game, "The Japan Open", will be given by the university. Furthermore, if a team wins the Japan Open championship, they will be provided financial assistance to participate in the international RoboCup competition.

4 Questionnaire Survey

We asked course participants to evaluate the course by completing a survey questionnaire on the final day. 74 students (73 male, 1 female) completed the whole course. There were 73 freshmen and one junior. All of them answered the questionnaire. The survey results are found below:

Q1. Part of work in a team: 55 students took charge of work in their own field; for example, students from the mechanical engineering department mainly took charge of mechanical work, and so on.
Q2. Interest in other fields: 71 students answered "Yes". This the result we had hoped for, and it was good that the participants had interest in other fields. The team arrangement policy placing students from different departments on each team may have contributed to the increase in student interest in other fields.
Q3. Team arrangement: 2 students answered that a team should consist of the students from a single department. 72 students approved of the inter-departmental team arrangement. We had hoped that students would make good use of the chance to communicate with students from other fields. and they responded positively to this.
Q4. Knowledge and skills you obtained: Answers were mechanical process, making electronic circuit, software, wireless communication, and so on. Many students answered that they gained knowledge and skills from other fields other than their own specialization, something they do not usually experience in their department course. The confidence which come from such experiences might have influenced the answers. Actually it is hard to understand and make good use of the knowledge and skills of other fields in only five days, but having the confidence based on the experiences will be beneficial for students' future study.
Q5. Was the course worthwhile?: 65 students answered "Yes". 41 students answered "Fully worthwhile". For some students who already have prior experience in building robots or electronic circuits, the contents of the course might be too easy, and they might feel bored. This course is basically designed for freshmen who don't have such experience. For experienced students, a more advanced course should be offered.

Q6. Good or Bad points of the course: There were many kinds of answers, both good and bad. The good points included exchange between students from different fields, fun building a robot, learning a wide variety of fields, and so on. These are also our hopes, and it was good to be evaluated positively by students.

On the other hand, many bad points were also pointed out. They are poor preparation, too busy, classes were too long, end of work of a day was vague, poor tools, and so on. We can understand most of these. Many problems resulted from the fact that this was the first year that the course was offered, and we realize that improvements must be done.

Q7. Work time: 45 students answered “enough”, while 28 answered “not enough”. We believe the difference in answers is a result of differences in their experiences.

Q8. Independent activity after course: 38 students expressed their hope of independent activity after course. Actually 10 students continued to refine their robots and participated in “The President Cup” competition in autumn. We must continue to support such hopeful students. The problem is keeping the work room reserved for independent activity and also providing parts and tools.

Q9. Do you recommend this course to underclassmen?: 66 students answered “Yes”. Many points to improve were pointed out in the previous question, but this answer indicates that many students evaluated the course as worthwhile though not quite satisfactorily.

Q10. Achievement: 67 students answered that they had a sense of achievement.

Q11. What did you make in past?: 60 students answered that they had experience building a plastic model. 20 participants had built a radio controlled car, while 12 participants had made robots. Since the participants are all engineering students, these answers are consistent with expectation.

Q12. Soldering skill: 34 students didn’t have ecperience in soldering before they attended this course. 33 of them answered that they obtained soldering skills. Actual experiences were valuable for them.

Q13. Programming in C language: 47 students answered that they understood the program used for robot control, while 26 students answered that they couldn’t understand it. Understanding the concept of computer programming requires time and training. Furthermore most participants had not yet taken a class of programming. Therefore if they can’t understand the programs it is unavoidable, even though the contents were limited to simple concepts.

Summary of questuonnaire survey

Many participants answered positively to most questions. It is concluded from the answers that the participants evaluated the course as worthwhile though not quite satisfactorily.

Positive points of the course were the exchange between students from different fields , the fun of making a robot, learning a wide variety of fields, and so on. But many points to improve were also pointed out by students. They included poor preparation, too busy, long classes, unclear quitting time, poor tools, and so on. These are areas for improvement. Most participants thought that they obtained knowledge and skills of various fields. They had a sense of achievement.

5 Conclusion

In this paper we have described the new practice course which introduces simplified soccer robots to the undergraduate education. We asked course participants to evaluate the course on the final day. Most participants thought that they obtained knowledge and skills from various fields and that they had a sense of achievement.

This trial is expected to increase the number of students who are interested in the science and technology related to robots. It will be an example of the effectiveness of the RoboCup activity in education.

References

1. F. Martin : “Kids Learning Engineering Science Using LEGO and the Programmable Brick”, Proc. AERA-96, 1996.
2. Maja J Mataric: “Robotics Education for All Ages”, Proc. AAAI Spring Symposium on Accessible, Hands-on AI and Robotics Education, 2004.
3. R.D’Andrea : “Robot Soccer: A Platform for Systems Engineering”, Computers in Education Journal, 10(1) : pp.57–61, 2000.
4. Jacky Baltes, Elizabeth Sklar, and John Anderson : “Teaching with robocup”, Accessible Hands-on Artificial Intelligence and Robotics Education, SS-04-01 in AAAI Spring Symposium, pp.146–152, 2004.
5. John Anderson, Jacky Baltes, David Livingston, and Elizabeth Sklar : “Toward an undergraduate league for robocup”, Proc. the RoboCup Symposium, 2003.
6. Elizabeth Sklar, Simon Parsons, and Peter Stone : “RoboCup in Higher Education: A Preliminary Report”, RoboCup-2003: Robot Soccer World Cup VII, Springer Verlag, Berlin, 2004.
7. http://web.mit.edu/mecheng/dme/
8. Peter Stone : “RoboCup as an Introduction to CS Research”, RoboCup-2003: Robot Soccer World Cup VII, Springer Verlag, Berlin, 2004.
9. Coradeschi S. and Malec J. : “How to make a challenging AI course enjoyable using the RoboCup soccer simulation system”, RoboCup-98: The Second Robot World Cup Soccer Games and Conferences, Springer Verlag Lecture Notes in Artificial Intelligence (LNAI), pp.120–124, 1999.
10. http://www-2.cs.cmu.edu/˜pprk/
11. T.Fujii, H.Fujiyoshi, Y.Nagasaka, T.Takahashi : “RoboCup Small Robots as Education Platform for Undergraduate Students”, SICE System Integration Division Annual Conference, Dec. 2002, pp.205–206 (in Japanese).

A New Video Rate Region Color Segmentation and Classification for Sony Legged RoboCup Application

Aymeric de Cabrol[1], Patrick Bonnin[2,1], Thomas Costis[2], Vincent Hugel[2], Pierre Blazevic[2], and Kamel Bouchefra[2,⋆]

[1] Laboratoire de Transport et de Traitement de l'Information L2TI
Institut Galilée, Av JB Clément, 93430 Villetaneuse, France
mrik@l2ti.univ-paris13.fr
[2] Laboratoire de Mécatronique et Robotique de Versailles
10-12 Av de l'Europe, 78 140 Vélizy, France
{bonnin, thcostis, hugel, pierre}@lrv.uvsq.fr

Abstract. Whereas numerous methods are used for vision systems embedded on robots, only a few use colored region segmentation mainly because of the processing time. In this paper, we propose a real-time (i.e. video rate) color region segmentation followed by a robust color classification and region merging dedicated to various applications such as RoboCup four-legged league or an industrial conveyor wheeled robot. Performances of this algorithm and confrontation with other existing methods are provided.

1 Introduction: Motivations and Constraints

Our motivation is to find a segmentation method that is robust to changes of lighting conditions as well for the RoboCup challenges in the four-legged league, as for other applications such as the vision system of the wheeled industrial conveyor robot of the CLÉOPATRE project [1]. RoboCup challenges we want to deal with are:

- the Vision Challenge,
- the ability to tune vision parameters quickly.

These robotics applications require to get several kinds of information: on the one hand they need to identify colored areas of the image that are supposed to be the ball, the goals, the landmarks, the players and the soccer field for the RoboCup application. On the other hand they have to extract the edges that are supposed to be the white lines on the field.

The constraints are stringent in term of available computing power. The proposed segmentation will be performed at video rate on AIBO ERS-7 for the

[⋆] Thanks to the RNTL. This work is partly supported by the French research office, grant number 01 K 0742, under the name of CLÉOPATRE project.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 436–443, 2006.

RoboCup Challenge, and on the laptop of the vision system embedded on the wheeled robot of the CLÉOPATRE project.

As we want the algorithms to be usable for both applications and others, they must not be dedicated to a specific application.

2 RoboCup Related Works

2.1 Different Directions

As most teams (UNSW [2], CMU [3], UChile [4], etc ...), we use since our first participation in Paris in 1998 [5] a color classification based on a look-up table, followed by a blob detection as low level vision processes. The look-up table is generated by taking a large number of color samples from images of the game field.

Unfortunately color labeling is sensitive to changes in illumination, and manual calibration is time consuming. So autonomous and dynamic color calibration methods have been proposed [6, 7]. The latter paper underlines that the effects of changing lighting conditions are the displacement of the position of the colors in the color space, but the relative position of colors to each other does not change. So the green of the soccer field is taken as a reference color.

Another direction is to use a specific segmentation [8]. Vision processing speed can be increased by avoiding processing all pixels of the image. Even though this specific algorithm is very efficient, we do not want to go that way because we would like to implement a more general purpose segmentation.

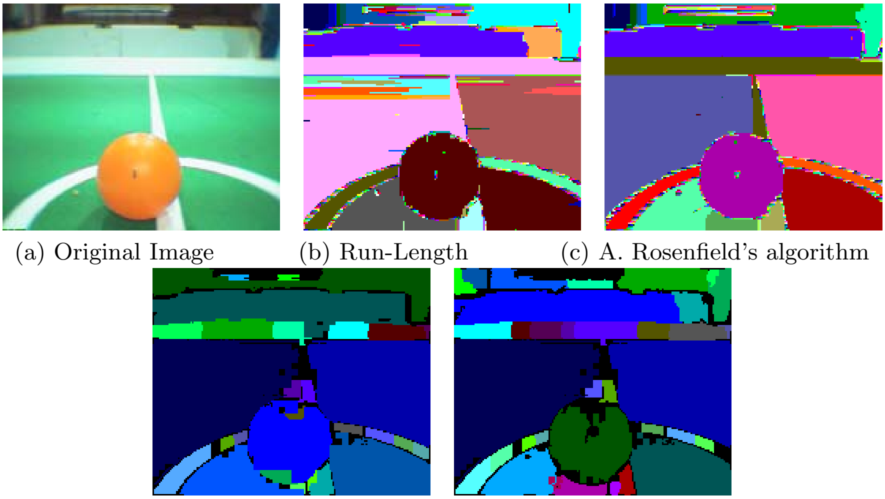
2.2 Proposed Direction

Our key idea is to use a color region segmentation on the whole image, step which is widely independent of lighting conditions, followed by a region color classification. Such a classification, based on the average intensity values of the pixels belonging to the regions, is more robust than color pixel classification. Unlike [9], the growing procedure must be fast enough to guarantee the independent whole segmentation of every image.

Outside of the RoboCup framework, many mobile and autonomous robots integrate vision systems for navigation, obstacle avoidance, and for other robotics tasks [10, 11, 12]. Vision systems are generally based on edge or optical flow segmentations, or color classification of pixels, or neural networks. But few are based on region segmentation [13, 14], even if the pieces of information extracted by region image features are useful for the accomplishment of the robotic task. In addition Priese and Rehrmann implemented their algorithm on a dedicated hardware (Transputer Image Processing System) [14].

3 Region Segmentation Algorithms

Firstly, we adapted two *blob detection* methods (the A. Rosenfield, JL. Pflaz algorithm [17], and a run-length algorithm [3]) to color region growing. The performances are slightly better with A. Rosenfield's algorithm. Temporal results



(a) Original Image (b) Run-Length (c) A. Rosenfield's algorithm

(d) Parallel Isotropic Growing (e) Cooperative Segmentation

Fig. 1. Segmentation results with different algorithms

will be given in §. 5. But in both cases the quality of the segmentation and the processing time are too dependent of the image and of the values of the control parameters.

Secondly, we adapted our *Gray Level Cooperative Segmentation* [16] to color segmentation. The first step is a splitting implementation adapted from the first step of the Horowicz and Pavlidis's algorithm. A region is settled if no edge point is found in it and if an homogeneity criterion is verified. The second step is a parallel and isotropic growing from embryo regions. The quality of this segmentation is quite independent of the image, and does not vary in dramatic proportions with the values of the control parameters. But the processing time is too long! In removing the cooperation with the edge detection, the quality of the results decreases without increasing time performances! It can be noticed from the segmentation results of figure 1 that an isotropic and parallel growing procedure (see (d) and (e)) gives better quality results than a line scan procedure (see (b) and (c)).

4 Proposed Segmentation

Our color region segmentation method is composed of three main steps:

- a *Hierarchical and Pyramidal Merging*, initialized from the pixels,
- a *'Video Scan' (or 'Data Flow') Merging*, adapted for the pyramidal regions,
- a *Color Merging*, merging step based on a color classification of regions.

Each of these two previous main steps considers the operation of merging of each kind of regions separately as a sub-step. Regions are 3×3, 9×9 and 27×27 pixels. Be aware that all the pixels of these regions are not gathered in the square region.

For the first step, as in [13], we use a hierarchical and pyramidal merging method which takes advantage of the connexity of the current pixel neighbourhood, except that we work with an orthogonal topology, and not an hexagonal one. This merging is more efficient than using the quad tree structure. The order of the merging is the following: 3×3 regions from image pixels, then 9×9 regions from 3×3 and finally 27×27 regions from 9×9. The germ is the central pixel or the central region. For the initial step of this fusion, each pixel belonging to the 3×3 neighbourhood is merged to the central pixel (germ) if their intensity values on the three image planes (RGB or YUV) are not too different (see Fig.2 (b)). This sub-step requires a first control merging parameter based on the difference of adjacent pixel intensity. Then, successively, 9×9 (see Fig.2 (c)) and 27×27 (see Fig.2 (d)) regions are obtained in quite a same manner. A neighbour 3×3 region (resp. 9×9) is merged into the 3×3 (resp 9×9) central region (germ) if they verify the connexity criterion, and if the intensities of the adjacent 3×3 regions (in both cases) are not too different. The connexity criterion is the following: the two adjacent pixels must belong to the regions, one for each.

Since the extraction of edge point information is also needed for the localization of the robot, and as shown by the German Team [8] the localization does not need the computation on each pixel of the high resolution image, we combine an adaption of Kirsh 4 gradient operator [18] and an edge thinning step with the initial 3×3 pyramidal gathering. The Kirsh 4 operator is applied on the Y plane only for a pixel every 3 lines and every 3 columns. The thinning step is applied simultaneously on the reduced image.

Associated with the initial 3×3 pyramidal gathering, this processing takes 0.11 ms more than the gathering alone. This additional time is only the computation time. Alone, this edge point detection takes 0.55 ms.

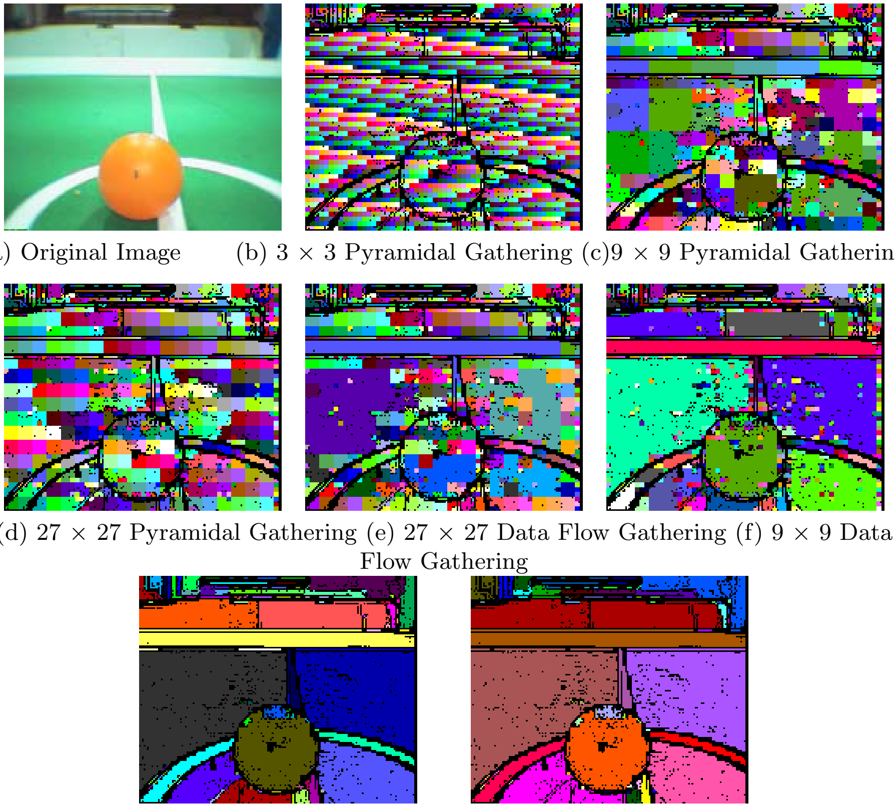
For the second main step, the regions are considered in the opposite order: 27×27, then 9×9 and finally 3×3. The principle of gathering of the 27×27 (see Fig.2 (e)) regions is exactly the same as those used for the pixels in the A. Rosenfield & JL. Pflatz's algorithm, except that it is applied on 27×27 regions rather than on pixels, and therefore the connexity must be verified. Only the two past regions (relatively to the video scan) are taken into account: the left and the upper ones. The gathering of the 9×9 (see Fig.2 (f))and 3×3 (see Fig.2 (g)) region is slightly more complex, because it takes into account the two previous past regions and two next regions: the right and the bottom ones. The merging criteria are the same as for the first step: related to the connexity and to the difference of intensity of 3×3 regions. The information related to the topology of regions (e.g. gathered points and then 3 × 3 and 9 × 9 regions) is stored in the data structure of regions. So our algorithm requires only one examination of each image pixel value during the initial 3×3 gathering. All previous steps are quite independent of the changes of the lighting conditions.

The third main step is constituted by two sub-steps:

- *Color classification of regions*
- *Region merging based on Color Classification*

For the moment, the *color classification of regions* consists of finding the best color classification in the YUV space for each region, and to verify the coherence of this classification in the UV space. The results seem to be stable, because it is more robust to classify mean intensity values on a given region rather than pixel intensity values as for a pixel classification. But we are looking for another color space as HSV. As for the 9×9 pyramidal gathering, the *region merging based on color classification* consists of considering one 3×3 every 3 according to the lines, and every 3 according to the columns, and to look at its upper, lower, right and left 3×3 neighbour regions. The regions are merged if they are classified with the same colour. The good quality of the segmentation results can be noticed. In fact the pyramidal gathering main step followed by the data flow gathering procedure simulates a parallel and isotropic growing (see Fig. 2). Two different kinds of parallelism may be underlined:

- all adjacent regions are *simultaneously* merged into the growing region,
- several regions are growing simultaneously.



(a) Original Image (b) 3 × 3 Pyramidal Gathering (c)9 × 9 Pyramidal Gathering

(d) 27 × 27 Pyramidal Gathering (e) 27 × 27 Data Flow Gathering (f) 9 × 9 Data Flow Gathering

(g) 3 × 3 Pyramidal Gathering (h) Color Identification Gathering

Fig. 2. Segmentation Results of the Different Steps of our Algorithm

Table 1. Detailed processing times of algorithms steps. *Processing time* column contains duration of each step, whereas *total* shows the total duration up to this step. *Percentage* is the contribution of the step to the whole process duration. Steps are in bold font, while sub-steps are in normal one. The first sub-step is shown as the more time consuming, that underlines the efficiency of the following gatherings.

Algorithm	Processing Time	Total	Percentage
Pyramidal Gathering	**4.44 ms**	**4.44 ms**	**64%**
3 × 3	3.95 ms	3.95 ms	57 %
Edge Point Detection	0.11 ms	4.06 ms	1.6 %
9 × 9	0.31 ms	4.37 ms	4.4 %
27 × 27	0.07 ms	4.44 ms	1 %
Data Flow Gathering	**1.45 ms**	**5.89 ms**	**20.8 %**
27 × 27	0.01 ms	4.45 ms	0.15 %
9 × 9	0.31 ms	4.76 ms	4.4 %
3 × 3	1.13 ms	5.89 ms	16.5 %
Color Classification Gathering	**1.06 ms**	**6.95 ms**	**15.2 %**
Color Classification	0.06 ms	5.95 ms	0.9 %
Color Gathering	1.0 ms	6.95 ms	14.3 %

This last property is due to the fact that the merging criterion between two given regions is based on the mean intensity values (on Y, U and V planes) of the initial and connected 3×3 regions (one for each given region). These parameters do not vary during the growing of regions and are independent of the order of the merging between regions.

5 Results and Comments

The temporal results are obtained with an ultra light notebook DELL X300 (Intel Pentium M 378, 1.4 GHz, 2 MB L2 cache, 400 MHz FSB) dedicated to the embedded vision system of a robot. Approximately 10 images are used for testing. The image size is 176×144, with 3 bytes per pixel. The performances of the Sobel's and Kirsh4's operators for edge detection are given for comparison. The processing time of a given algorithm must be at most twice the one of Kirsh's operator to be of interest for the following of our studies. Results are presented in table 2.

Though the cooperative segmentation extracts edge points also, this algorithm is faster than its version without the cooperation. In fact, the edge points make the isotropic and parallel region growing faster. But we are surprised at the bad performances of the adaptation of the run-length algorithm. The explanation is that too many segments are generated, and merging them takes a long time. We are also surprised at the good performances of our method, compared to Kirsh's operator and to the adaptation of A. Rosenfield's algorithm. The explanation is the reduced number of pixel access: 3 pixels for the adaptation of Rosenfield's algorithm, 9 for the Kirsh's, and 1 for our algorithm and the pyramidal approach.

Table 2. Comparison between several algorithms duration

Algorithm	Processing Time	Speed Up
Adapt. of A. Rosenfield's Algo	13.6 ms	2.3
Adapt. of Run-Length	28 ms	4.75
Cooperative Segmentation	72.5 ms	12.3
Parallel and Isotropic Growing	106 ms	18
Proposed Method	5.89 ms	1 : reference
Color Kirsh 4 operator	12.8 ms	2.17
Color Sobel operator	17.6 ms	2.99
Kirsh 4 on Y plane	4.45 ms	0.755
Color Classification and Blob Extraction	2.57 ms	0.436

All source, binaries and tested images will be available on the web site of the CLÉOPATRE project. The quality of the resulting segmentation is suitable for robotics applications. The different regions of the color image are correctly separated. Some points are missing inside regions. Though this is penalizing for a nice looking segmentation, all needed region attributes (gravity center, including boxes etc..) are correct.

Taking into account the processing times of table 2, the proposed segmentation will run at video rate on the robot of the CLÉOPATRE project. For the moment, the *Color Classification and Blob Extraction* (2.57 ms on our test computer) and the *Kirsh4 on Y plane* (4.45 ms) are still used on the AIBO ERS-7 for low level vision processing. The processing time of our new segmentation is similar (6.95 ms compared with 7.02 ms). During RoboCup 2005 we implemented the new segmentation algorithm on AIBO ERS-7. It ran at 15 Hz together with all other modules such as locomotion, localization and behaviours.

6 Conclusion

We have proposed a general-purpose robust real-time region color segmentation and classification, and shown that this was more efficient than pre-existing methods.

The swiftness of this algorithm is mainly due to the reduced number of pixel access, to the bottom-up and then top-down hierarchical merging. Its robustness is the consequence of the region color classification based on mean value for the area.

References

1. M. Silly-Chetto, T.Garcia, F.Grellier, Open source components for embedded real time applications, IEEE Int Conf on Control and Automation, June 2002, Xiamen, China.
2. A.Olave, D.Wang, J.Wong, T.Tam, B.Leung, MS.Kim, J.Brooks, A.Chang, N.Von Huben, C.Sammut, B.Hengst, The UNSW RoboCup 2002 Legged League Team, 6th Int. Workshop on RoboCup 2002 Fukuoka, Japan.

3. J.Bruce, T.Balch, M.Veloso, Fast and inexpensive color image segmentation for interactive robots. Proceedings of the 2000 IEEE / RSJ Int Conf on Intelligent Robots and Systems (IROS'00) 3 (2000) 2061-2066.
4. JC.Zagal, R del Solar, P.Guerrero, R.Palma, Evolving Visual Object Recognition for Legged Robots, 7th Int. Workshop on RoboCup 2003 Padova, Italy.
5. V.Hugel, P.Bonnin, JC.Bouramoué, D.Solheid, P.Blazevic, D. Duhaut, Quadruped Robot Guided by Enhanced Vision System and Supervision Modules, RoboCup 1998, Springer Verlag, M. Asada and H. Kitano ed.
6. D.Cameron, N.Barnes, Knowledge based autonomous dynamic colour calibration, 7th Int. Workshop on RoboCup 2003 Padova, Italy.
7. M.Jüngel, J.Hoffmann, M.Lötzch, Real Time Auto Adjusting Vision System for robotic soccer 7th Int. Workshop on RoboCup 2003 Padova, Italy.
8. T.Röfer, M.Jüngel, Fast and Robust Edge - Based Localization in the Sony four-legged robot league, 7th Int. Workshop on RoboCup 2003 Padova, Italy.
9. F.Hundelshausen, R.Rojas, Tracking Regions, 7th Int. Workshop on RoboCup 2003 Padova, Italy.
10. G.N.DeSouza, A.C.Kak, Vision for Mobile Robot Navigation: A Survey, IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI, vol.24 n2, February 2002.
11. Proceedings of IROS 2001, IEEE Int Conf on Intelligent Robots and Systems, October 2001, Hawai USA.
12. Proceedings of ROMAN 2001, IEEE Int Workshop on Robot and Human Communication, September 2001, Bordeaux - Paris FRANCE.
13. L. Priese, V. Rehrmann, A Fast Hybrid Color Segmentation Method, Proceedings Mustererkennung 1993, pp 297-304, DAGM Symposium, 1993 Lübeck.
14. L. Priese, V. Rehrmann, R. Schian, R. Lakmann, Traffic Sign Recognition Based on Color Image Evaluation, Proceedings IEEE Intelligent Vehicles Symposium '93, pp. 95-100, July 1993, Tokyo, Japan.
15. S.Horowitz, T.Pavlidis, Picture segmentation by a direct split-and-merge procedure, Sec Int Joit Conf on Pattern Recognition, pp 424-433, 1974.
16. P.Bonnin, J.Blanc Talon, JC.Hayot, B.Zavidovique A new edge point / region cooperative segmentation deduced from a 3D scene reconstruction application, SPIE 33rd Ann Int Symp on Optical & Optoelectronic Applied Science & Engineering, vol Application of digital image processing XII, August 1989, San-Diego California.
17. A. Rosenfeld, JL Pfalz, Sequential operations in digital picture processing, Journal of ACM, vol 13, no 4 1966.
18. Kirsh, Computer determination of the constituent structure of biological images, Computer & biomedical research, USA, 1971 vol 4 no 3 pp 315-328.

A Novel and Practical Approach Towards Color Constancy for Mobile Robots Using Overlapping Color Space Signatures

Christopher Stanton and Mary-Anne Williams

Innovation and Technology Research Laboratory
Faculty of Information Technology
University of Technology Sydney
{cstanton, Mary-Anne}@it.uts.edu.au

Abstract. Color constancy is the ability to correctly perceive an object's color regardless of illumination. Within the controlled, color-coded environments in which many robots operate (such as RoboCup), engineers have been able to avoid the color constancy problem by using straightforward mappings of pixel values to symbolic colors. However, for robots to perform color vision tasks under natural light the color constancy problem must be addressed. We have developed a color vision system which allows for the color space signatures of different symbolic colors to overlap. This raises the question: if a specific pixel value can be mapped to multiple symbolic colors, how does the robot determine which color is the "correct" one? Context plays an important role. We adopt a knowledge driven approach which allows the robot to reason about uncertain color values. The system is fully implemented on a Sony AIBO.

1 Introduction

Within the color-coded world of RoboCup[1], most teams use color image segmentation techniques to assist with the identification of relevant objects, such as the ball, goals, landmarks and other robots. Generally, the vision systems developed for RoboCup take advantage of the engineered RoboCup environment – where the lighting is bright, evenly dispersed and constant, and the colors of important objects highly distinct – by providing a one-to-one mapping between pixel values and symbolic colors, i.e. for any given raw pixel value there is a maximum of one corresponding symbolic color class.

Such systems ignore the reality that even within the controlled RoboCup environment the colors of important objects do indeed overlap within the color space. For example, many teams within the legged-league are familiar with the problem that when in shadow or dim light, the orange of the soccer ball can appear to be the same color as the red uniforms worn by one team of robots. Such color misclassifications can have dire consequences for system performance, as witnessed by many a robot

[1] http://www.robocup.org

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 444 – 451, 2006.

chasing the red uniform worn by a fellow robot, guided by the misconstrued belief that the red robot is in fact the orange soccer ball.

Color constancy is the ability to perceive an object's color correctly regardless of illumination [1]. To help overcome this problem, we have developed a color vision system which allows for a pixel's raw data value to be mapped to a set of many possible symbolic color classes. However, introducing such a relationship raises the question: if a specific pixel value can belong to multiple symbolic colors, how does the robot determine which color is the "correct" one? In this paper we detail our approach which allows the robot to reason about uncertain color values. The system is completely implemented on a Sony AIBO[2], and an initial version was used with great success at RoboCup 2004.

2 Color Constancy and Mobile Robotics

The appearance of a surface's color is dependent upon the complex interaction of a number of factors, including the reflectance properties of the surface, the camera, and the lighting conditions. A change in any of these factors can affect an object's apparent color. Illumination is rarely constant under natural light. Even when lighting is relatively constant, the viewing geometry for mobile robots is not. If we consider the legged-league of RoboCup, in which teams of Sony AIBOs play soccer, the viewing geometry is consistently shifting as the robot's camera is located in, and moves with, the robot's head. For example, the robot's own head often casts shadows over the ball.

Mobile robots must also compensate for imperfect and noisy sensors. For example, the camera on the AIBO ERS7 has a "fisheye" effect which produces a blue discoloration around the edge of the camera. The camera also discriminates dark colors poorly, making it difficult to distinguish between colors such as the dark grey skin of an AIBO ERS210, the black pants of a referee, field green in shadow, or the blue team's uniform. Also, the fact that a mobile robot is indeed mobile can affect camera performance. In robotic soccer robots frequently collide, and for legged robots there is an element of "bounce" when the robots walk. Motions such as these can cause color distortion and blur within the images captured by the robot's camera. Finally, any solution must be capable of operating in real-time within the limited computational resources provided by the robot's hardware.

3 Prior Work

There is an enormous body of literature regarding computational color constancy. However, the vast majority of this research focuses on static images, database image retrieval, and off-board processing. The general aim of computational color constancy is to determine the effect of the unknown illuminant(s) in a scene, and then to correct the image by either mapping to an illumination invariant representation, or by correcting for the illuminant.

[2] http://www.sony.net/Products/aibo/

In terms of color constancy applied to mobile robots there is a much smaller body of knowledge. Many approaches use color insensitive algorithms to assist with object or color recognition, so that once an object or color is recognized, the robot can survey the pixel values within the image, and then use these values to update color tables dynamically, e.g. [2], [3], [4], [5]. Another method is to use image statistics, either from a single image or a series of images, to determine a global scene illuminant - the rationale being if lighting conditions can be accurately classified, then an appropriate color mapping between raw pixel values and symbolic colors can be selected by the robot, e.g. [6] [7]. Lastly, an alternative approach is to pre-process an image to improve the separation of colors within the color space so that symbolic classes are more tightly clustered around a central point, e.g. [8].

The area of specific concern in this paper is determining symbolic color class membership in robotic color vision tasks when the symbolic colors have substantial overlap within a color space, even when the lighting conditions are relatively constant. Within RoboCup most teams avoid the problem by adding controls to their color calibration process which govern overlap and outliers for symbolic colors within the color space (e.g. [9], [10]). Mayer *et al.* [11] reported that when playing middle-size league soccer under natural light they experienced substantial overlap between white and other symbolic colors. Their unsatisfactory solution was to simply give "priority" to colors other than white. A common problem within the legged-league is that when looking down at the orange ball it can appear red, and one team [12] tried to compensate for this by building two color tables – one for when the robot is looking down, and one for all other situations. However, they report mixed success, as orange tends to merge with red and having two color tables did not solve their problems of color misclassification. In the most related approach to our work, [13] report on initial attempts to identify overlapping color signatures within the color space. They describe pixel values for which no overlap exists as "core" colors, and pixel values for which overlap exists as "maybe" colors.

4 Our Approach

Rather than focusing on building a color constancy system that can overcome drastic lighting changes through mathematical calculations of the scene illuminant, we have focused our efforts on developing a vision system which can provide an expressive representation for reasoning about the uncertainty of colors. We are motivated by our longer term aim of allowing the robots to reason about the color of pixels and objects using their knowledge about the environment, such as lighting conditions, camera, and prior experiences.

The first step of our approach to the color constancy problem is to identify the pixel values of different symbolic colors that overlap in color space, and instead of removing or ignoring these particular pixel values, we provide the robot with the complete set of possible candidate colors for any given pixel value. Importantly, this reduces the search space for classifying pixel values whose color signatures overlap. Secondly, we created a symbolic color class called "dark noise" to capture dark areas within the image in which much color overlap occurs, such as shadow. Next, the color classification algorithm assigns each pixel a value which indicates the set of possible colors for that pixel. For pixel values with more than one possible color, color classification relies upon local image area statistics, the pose of the robot, and other heuristic based knowledge.

4.1 Image Sampling and Training

We use a color labeling process in which a human trainer labels regions within the image that correspond to the objects of interest, such as the ball, field, robot uniforms, landmarks, and so forth. A custom built software system, using a relational database, stores every unique raw pixel value "*p*" that the user selects for every symbolic color "*c*". Thus, given a set of symbolic colors, e.g. $C = \{$*white, green, pink, orange …*$\}$, it is possible for any pixel value to be a member of an arbitrarily assigned subset of C, depending upon the pixel to symbolic color relationships identified by the human trainer. We call this subset of C the *candidate colors* for a pixel value. While many pixel values will share the same symbolic color relationships, and hence candidate colors, e.g. $p_1 \equiv p_2$, invariably a large proportion of pixel values will have different symbolic color relationships. For example, $p_1 \in \{$*green, robot blue, beacon blue*$\}$, $p_2 \in \{$*orange, red*$\}$, $p_3 \in \{$*orange*$\}$, and so forth. In accordance with the terminology used in [18], we call pixel values for which there is only one candidate color "core-colors" (e.g. $p \in \{$*orange*$\}$), and pixel values for which there are multiple candidate colors "maybe-colors" (e.g. $p \in \{$*orange, red*$\}$). In other words, a core color is a pixel value for which there exists no overlap within the color space – they have only ever been assigned to one symbolic color - while a "maybe-color" is a pixel value which has been assigned to two or more symbolic colors.

At any point during the training process, the user can generate three artifacts that are required by the robot's vision system:

1. A structured file containing the complete set of unique candidate color combinations, with each combination possessing a unique index for the purposes of identification.
2. A color lookup table, which for every possible raw pixel value, provides an index to the corresponding set of candidate colors.
3. A file containing the mean (prototypical) value for each symbolic color in terms of raw pixel value.

4.2 Color Labeling and Image Segmentation

Our color calibration system has provided our robots with a more detailed level of color perception. In previous systems a particular pixel value was either unknown, or it belonged to a specific symbolic color. Now, a pixel value can be either unknown, belong to one specific symbolic color, or belong to a specific subset of the entire spectrum of symbolic colors. Thus, for many pixels within the image we are forced to make a new decision: which color is the correct one? To answer this question we trialed a variety of simple and efficient computational techniques, all of which can operate in real-time on both an ERS7, as well as the older and more computationally challenging ERS210.

The algorithm which provided best results for varying lighting conditions within our research laboratory was surprisingly simple. The algorithm takes advantage of the distinction between core colors and maybe-colors, by treating core colors as influential local area predictors for maybe-colors. For example, if a maybe color pixel could be either red or orange, but is surrounded by more orange core colors than red core colors, then it will be assigned the color orange. By only considering candidate colors, and not the complete set of colors, we are able to reduce the search space, increase the speed of the algorithm, and provide surprisingly natural results. In the absence of candidate core colors within a local area of the image (which can occur in images in which there are large concentrations of maybe-colors), or when there is an equal abundance of different neighboring candidate colors (e.g. 4 red and 4 orange neighboring pixels), Manhattan distance metric is used to find the closest candidate color. However, our aim is not to present, or find, the most sophisticated algorithm for correctly color segmenting an image, but rather to demonstrate how knowledge of relationships between pixel values and overlapping symbolic color signatures is a powerful alternative for overcoming color constancy issues. Code containing the implementation of this algorithm can be obtained from [14].

4.3 Results

Fig. 1 displays a raw image taken from an ERS7, together with images indicating the maybe colors within the image and the final segmented image.

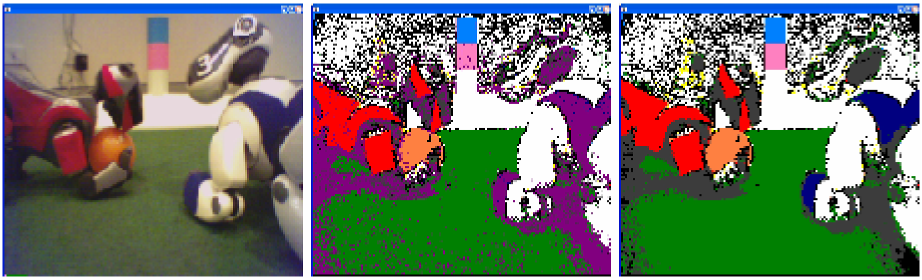


Fig. 1. An image from an AIBO ERS7 *(left),* the overlapping colors in the corresponding image are represented in purple (*centre*), and the processed image in which overlapping colors are assigned to symbolic colors (*right*). In the raw image there is a blue discoloration around the edge of the image, and there is little contrast or separation between the robot's blue uniform, shadows on the field, and the darker colors of each robot. The blue uniform consists almost entirely of maybe-colors.

A consistently surprising feature of processed images was the ability to accurately classify the regions of the image which corresponded to shadows on the field, and in some cases also on the robot. While such features are currently not used by our object recognition routines, models of color constancy which involve some level of scene understanding will require robots to detect such features. Fig. 2 displays an image in which two robots almost collide, and the proximity of the two robots causes a decrease in the illumination within the image.

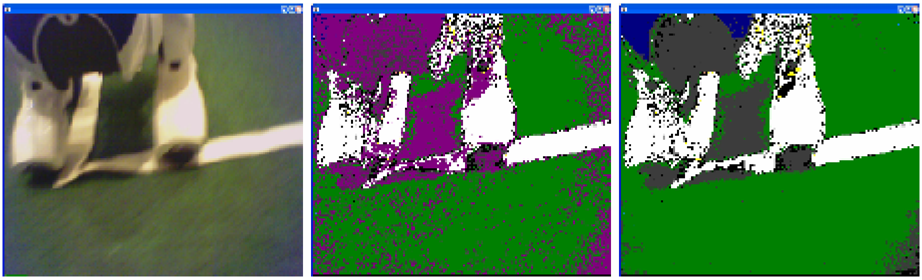


Fig. 2. Raw image from an AIBO ERS7 *(left),* the overlapping colors in the corresponding image are represented in purple (*centre*), and the processed image in which overlapping colors are assigned to symbolic colors (*right)*. The vision system is able to correctly segment the blue of the robot's uniform, and the area of shadow underneath the robot.

To demonstrate our results we have displayed images that we feel are indicative of the vision system's general performance. It is interesting to note that evaluating the performance of a color constancy system empirically is challenging. For any color constancy system there exists no automated method for recording the number or percentage of pixels within each image that are classified "correctly". Such notions of correctness or ground truth must be specified manually by a human tester. In much of the robotic research relating to color constancy systems are evaluated through behavioral performance tests. However, the performance of behaviors is also related to the performance of higher level routines (e.g. object recognition). Evaluating the performance of perception systems is an area of increasing significance for robotics [15].

One method adopted to evaluate the system's robustness was to vary the lighting conditions, and to also change the camera settings of the system. We were able to create color calibration tables that could function over a range of camera settings and lighting conditions. Fig. 3 displays an image in which the camera shutter speed was set at "fast", but the calibration tables used were created when using the "medium" shutter speed (effectively decreasing the brightness within the image).

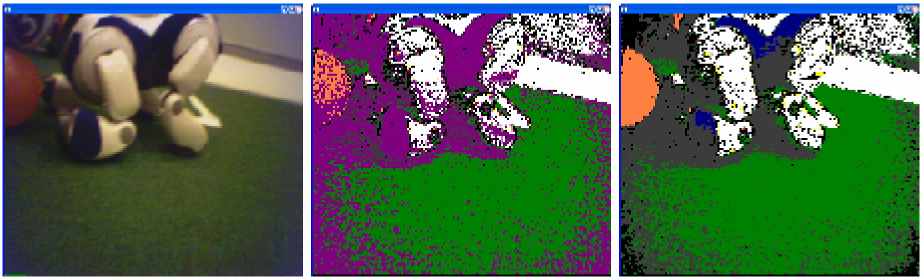


Fig. 3. Raw image from an AIBO ERS7 taken at fast shutter speed *(left).* The color tables used to segment the image were created at medium shutter speed. Due to the darker conditions an increased amount of overlap colors were present in the image (*centre*). Large parts of the ball overlap with robot red. The processed image effectively segments the ball and most of the robot's uniform (*right).*

5 Discussion

We have implemented a novel approach to deal with color constancy. Rather than avoiding or removing overlapping color space signatures, we have developed a system which uses the relationships between pixel values and overlapping symbolic color signatures to segment color images.

Our approach offered several immediate benefits. Color calibration can be undertaken more quickly, as the calibration method encourages the human trainer to identify all possible pixel values for each color of interest, rather than avoiding those that may cause misclassification (e.g. those that occur in shadow or on the borders of different objects within the image). Image segmentation has improved due to richer and more expressive color tables. Lastly, object recognition has also improved, due to not only image segmentation performance, but because object recognition routines can reason about the different levels of color uncertainty indicated by core colors, maybe colors, and unknown colors. For example, object recognition routines can exploit simple statistics, such as the percentage of maybe-colors within a blob, to reason about the likelihood of false identification of an object.

Future research will involve developing mechanisms for automatically generating the rules for determining the color membership of overlapping pixel values. When a human trainer labels the colors of pixels within an image, a wealth of contextual knowledge and scene understanding affects our interpretation of a pixel's color. A longer term aim is to investigate color training mechanisms that can embed this knowledge within the robot. For example, the human trainer compensates for the blue discoloration around the edge of the ERS7's image without conscious effort. Thus we are recording features such as the pixel's location in the image which allow us to calculate probabilistic rules for color membership which consider constant distortions of the camera.

References

1. Horn, B. Robot Vision, MIT Press, Cambridge, MA, 1987.
2. Cameron, D., & Barnes, N.: Knowledge-based autonomous dynamic color calibration. In: Robocup 2003, Padua, Italy (2003)
3. Jungel, M.: Using Layered Color Precision for a Self-Calibrating Vision System. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, RoboCup-2004: Robot Soccer World Cup VIII, Springer Verlag, Berlin, 2005.
4. Schulz, D., & Fox, D.: Bayesian Color Estimation for Adaptive Vision-based Robot Localization. IROS-04.
5. Gönner, C., Rous, M., and Kraiss K.: Real-Time Adaptive Colour Segmentation for the RoboCup Middle Size League. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, RoboCup-2004: Robot Soccer World Cup VIII, Springer Verlag, Berlin, 2005.
6. Lenser, S. and Veloso, M.: Automatic detection and response to environmental change. In: Proceedings of the International Conference of Robotics and Automation, May 2003.
7. Sridharan, M. and Stone, P.:Towards Illumination Invariance in the Legged League. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, RoboCup-2004: Robot Soccer World Cup VIII, Springer Verlag, Berlin, 2005.

8. Mayer, G., Utz, H., Kraetzschmar, G.:Towards Autonomous Vision Self-Calibration for Soccer Robots. In Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, 2002.
9. German Legged Team Report, 2004. Available at: http://www.tzi.de/4legged/bin/view/Website/Teams2004
10. University of Texas Legged Team Report. 2004. Available at: http://www.tzi.de/4legged/bin/ view/ Website/Teams2004
11. Mayer, G., Utz,H., Kraetzschma, G.: Playing Robot Soccer under Natural Light: A Case Study. 238-249. In RoboCup 2003.
12. University of Newcastle Legged Team Report. 2004. Available at: http://www.tzi.de/4legged/bin/view/Website/Teams2004
13. University of New South Wales Team Report. 2003. Available at: http://www.cse.unsw.edu.au/ ~robocup/reports.phtml
14. University of Technology Sydney Team Report and Code Release, 2004. Available at: http:// unleashed.it.uts.edu.au
15. Gardenfors, P., Karol, A., McCarthy, J., Stanton, C, and Williams, M. A Framework for Evaluating Groundedness of Representations in Agents: From Brains in Bottles to Mobile Robots. Workshop on Agents in real-time and dynamic environments, IJCAI 2005.

A Region-Based Approach to Stereo Matching for USAR

Brian McKinnon, Jacky Baltes, and John Anderson

Autonomous Agents Laboratory
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, R3T 2N2, Canada
{ummcki01, jacky, andersj}@cs.umanitoba.ca

Abstract. Stereo vision for mobile robots is challenging, particularly when employing embedded systems with limited processing power. Objects in the field of vision must be extracted and represented in a fashion useful to the observer, while at the same time, methods must be in place for dealing with the large volume of data that stereo vision necessitates, in order that a practical frame rate may be obtained. We are working with stereo vision as the sole form of perception for Urban Search and Rescue (USAR) vehicles. This paper describes our procedure for extracting and matching object data using a stereo vision system. Initial results are provided to demonstrate the potential of this system for USAR and other challenging domains.

1 Introduction

This paper describes our current research into practical stereo vision for autonomous and teleoperated robots. Stereo vision as a form of perception has many benefits for autonomous intelligent systems: in ego motion detection and simultaneous localization and mapping (SLAM) for example. For a teleoperated vehicle, stereo vision can be used to assist a human operator in judging distances, marking landmarks for localization purposes, and identifying desired objects in the environment.

The domain with which we employ stereo vision is that of Urban Search and Rescue (USAR). The goal of USAR is to allow robotic vehicles to explore urban areas following a disaster, locating human victims as well as dangerous situations (e.g. gas leaks). Robotic vehicles have the advantage of being able to traverse narrow voids that would be difficult for humans to reach, and also alleviate the need to place human rescue workers in dangerous situations. While this is a extremely useful application for robotic, computer vision, and artificial intelligence technologies, it is also a important challenge problem for these areas. NIST, for example, provides a standard challenge domain at a number of locations around the world each year, allowing a practical testbed for researchers to compare approaches to various aspects of this problem.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 452–463, 2006.

Our own primary interests lie in artificial intelligence and computer vision. We design autonomous robots that use vision as the sole sensor to support ego-motion detection, localization, and mapping. Recognizing that it will be some time before embedded systems become powerful enough and AI technology sophisticated enough for an autonomous system to perform well in such a challenging domain, we also work to provide vision-based solutions to enhance human teleoperation.

In order to deal with the unpredictability of the USAR domain, we follow two main principles in our work. First, all solutions must make as few assumptions regarding the nature of the domain as possible. For the purposes of vision, this means that we cannot assume that any camera calibration will remain perfect throughout a run, or that we can make assumptions about the nature of lighting. Secondly, our vehicles must be considered disposable, since damage and loss can occur. We thus attempt to provide solutions using commonly available equipment. This in turn supports the principle of generality: cheaper vehicles with less specialized hardware force us to deal with problems using intelligent software. For example, using visual ego-motion detection as opposed to relying heavily on shaft encoders for localization [2].

This paper describes the process by which we provide stereo vision for autonomous and teleoperated robotic vehicles under the conditions typical of USAR. We outline a novel algorithm for matching stereo images based on regions extracted from a stereo pair, and detail the steps taken at various points in the overall vision process to adhere to the twin goals of basic hardware and general solutions. We begin by reviewing other recent efforts to use vision as a primary perceptual mechanism, and follow by describing the phases involved in visual interpretation using our approach. Initial results of employing this approach in practice are then provided.

2 Related Work

Stereo vision is attractive because it generates a depth map of the environment. There are two basic approaches to the matching of stereo images: pixel-based and region-based approaches. Pixel-based approaches use feature points to match between images. Matching pixels between images is typically made more efficient through the use of epipolar constraints: if the cameras are perfectly calibrated, only one row in the image needs to be examined to find the same pixel in both images. While this is a reasonable approach from a computational standpoint, this method suffers from the problem of mismatching feature points. A single pixel, on the whole, provides very little evidence to support a match. Calibrating the cameras in order to support this is also non-trivial, and in domains such as USAR, the expectation that a fine calibration will remain accurate over time is an unreasonable one. Matching regions rather than pixels is an alternative intended to decrease mismatching, because much larger areas are matched to one another. However, these larger regions require correspondingly greater computational resources for a match to be performed. The approach we detail in

Section 3 improves on standard region-based matching through the simplification of regions, requiring fewer computational resources for matching while still maintaining robust matching.

The two most important steps in region-based matching are the identification and representation of features in the image. Research is active in this area, since current approaches often encounter environments that cause failure rates to become unmanageable. Examples of approaches currently being advocated include those of Lowe [9], Carson et al., [5] and Ishikawa and Jermyn[7].

Lowe's work [9] introduces an object recognition system known as Scale Invariant Feature Extraction (SIFT). This approach uses a feature representation that is invariant to scaling, translation, and rotation, as well as partially invariant to changes in illumination. Scale invariance is achieved through the use of the Gaussian kernel as described in [8]. For rotational invariance and efficiency, key locations are selected at the maxima and minima from the difference of the Gaussian function applied in scale space. Once a set of keys are defined for a given object, live images are scanned and objects are selected using a best-bin-first search method. Bins containing at least three entries for an object are matched to known objects using a least square regression. Experimental results show that the system is effective at detecting known objects, even in the presence of occlusion, since only three keys are necessary for a match to occur. Lowe and others have employed this method to implement a localization for reasonably structured environments [11], but nothing as unstructured as USAR.

In Carson et al.'s [5] Blobworld representation, pixels in an image are assigned to a vector containing their color, texture, and position. Colors are smoothed to prevent incorrect segmentation due to textures, and are stored using the L*a*b* color format. Texture features employed for categorization include contrast, anisotropy (direction of texture), and polarity (uniformity of texture orientation). Regions are grouped spatially if they belong to the same color and texture cluster. A gradient is generated in the x and y directions, containing the histogram value of pixels in that region. For matching, the user must begin by selecting blobs from the image that will be used for comparison against a database. Regions are matched to the database by the quadratic distance between their histograms' x and y values, in addition to the Euclidean distance for the contrast and anisotropy texture. This method was used as a basis for an optimal region match in [4]. It is unclear, however, how robustly the method handles translation of the blobs. In addition, this system is not directly usable for an autonomous system, since the user must select the candidate blobs.

Wavelet-based Indexing of Images using Region Fragmentation (WINDSURF) [1] is another recent approach to region extraction. In this approach the wavelet transform is employed to extract color and texture information from an image. A clustering algorithm is used on the output coefficients from the wavelet transform, producing regions that contain similar color and texture waves. By using only the coefficients, regions are clustered without considering spatial information. This means that images cannot be compared based on the location of the regions. However, it allows matching that is invariant to position and orientation differences. One limitation in this approach is that the region count must be defined, so clustering of dissimilar regions can occur in the presence of images that contain more features than expected.

3 Pragmatic Stereo Vision for USAR

The aim of our overall approach is to identify useful regions and match them between stereo images, with limited computational resources and under conditions typical of the USAR domain. In fully autonomous systems, stereo-matched regions are intended as input to routines for ego-motion detection, localization, and map-building (as employed originally in [3, 2] using a single camera). In teleoperated systems, this is intended to enhance the remote perception of the teleoperator by providing information about the distance and movement of objects.

We divide the process of performing region matching in stereo vision into six stages: Color Correction, Image Blur, Edge Detection, Region Extraction, Region Simplification, and Stereo Matching. Each of these stages performs a specific function in terms of allowing a visual scene to be matched using stereo vision. We examine the role and implementation of each of these phases in turn.

3.1 Color Correction

Under the varied conditions of Urban Search and Rescue, we cannot assume uniform lighting. Thus, there will be situations where imbalances exist in the color response of the two cameras capturing the stereo image.

We have found that in general extracting stereo matches can proceed with little inaccuracy due to color differences between the two stereo images (which can easily be seen in many stereo image pairs, such as the raw image pair shown at the top of Figure 1). We have also found, however, that normalization is useful in supporting human teleoperation in situations where ambient lighting is low. For those purposes, we perform color correction by normalizing the color channels.

Our method for normalization involves using the mean and standard deviation of the color channels. While the naive computation methods for these would require two passes through the data with a third to normalize, we do this in a single pass. This is done by relying on the assumption that the mean and standard deviation over a sequence of images are relatively stable, which will generally be the case in a reasonably slow-moving vehicle in a USAR scenario. Thus, at time t we use the mean of the image from time $t-2$ and the standard deviation of the image at time $t-1$ as approximations to the current values, and normalization can ensue at the same time future standard deviation and mean values are calculated.

Normalization is performed by defining a range of two standard deviations on either side of the mean as the entire range for the color channel. This allows outliers to be discarded, resulting in a more accurate representation of the color range in the image.

3.2 Image Blurring

Raw images obtained under the conditions typical of a USAR domain are extremely prone to noise. In addition to texture and lighting variations in the domain itself, noise is exacerbated by the approach we take to the USAR problem. Given current technology, any equipment that can be viewed as cheap enough to minimize the loss of a robot will have to include inexpensive video capture devices, which are highly prone to noise. This noise makes the detection of edges, and ultimately regions and objects, extremely difficult, and thus inconsistencies introduced by noise must be minimized through some form of image smoothing, such as blurring.

We employ Gaussian blurring in this process for a number of reasons. First, Gaussian blurring provides circular symmetry [12] - that is, lines and edges in different directions are treated in a similar fashion.

More importantly, a Gaussian blur can deliver this and remain efficient. In order to perform a Gaussian blur, a bell curve is approximated with integer values, typically binomial coefficients selected from Pascal's Triangle. These particular numbers have a useful principle that allows for an efficient implementation: to apply a blur of $N = k$, the coefficients of i and j, such that $i + j = k$ can be convoluted. For example, to apply a 3x4 blur, k =5 is selected, but rather than having to use the coefficients of 5 (the set $\{1\ 5\ 10\ 10\ 5\ 1\}$), the coefficients of 2 (the set $\{1\ 2\ 1\}$) and 3 (the set $\{1\ 3\ 3\ 1\}$) can be used - the first horizontally, and the second vertically to the result of the first. The result for each pixel is then normalized by dividing by the sum of the two coefficients. The practical result of this is that a small area of blur is repeated to generate large areas, rather than requiring the additional computational expense of blurring with a larger set of coefficients. The middle element of Figure 1 illustrates the result of a such a blur on a sample stereo image pair.

3.3 Edge Detection

Preprocessing via color correction and smoothing leaves the image in a state where we can begin to identify objects in the image with the expectation of some degree of accuracy. The first step in this process is to determine rough boundaries through edge detection.

We employ Sobel edge detection in our implementation, because it is computationally simple and proves robust under a variety of conditions. Sobel edge detection involves the application of convolution masks across the image. We employ two masks, for the horizontal and vertical dimensions respectively:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

After applying each mask, normalization is performed by dividing each pixel value by four. The resulting pixels are examined against a threshold value, where values larger than the threshold indicate an edge, as shown in the bottom element of Figure 1.

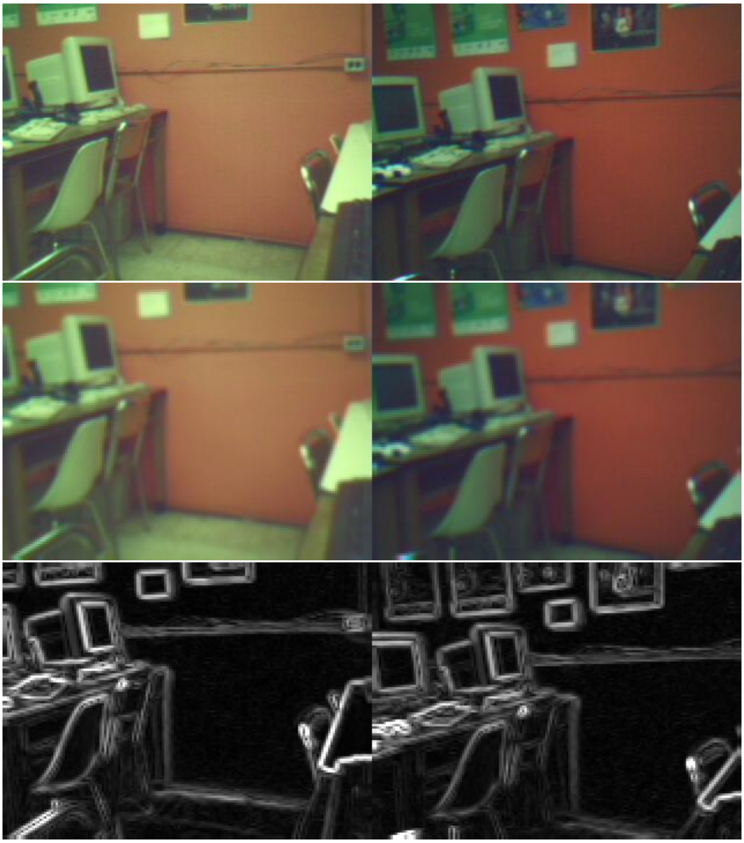


Fig. 1. The product of applying a Gaussian blur (middle) on the raw image pair (top). The blurred image is then subjected to Sobel edge detection (bottom).

3.4 Region Growing

Having obtained a set of strong edges from a smoothed image, we employ this along with the original smoothed image to determine a set of regions in each individual image.

Our approach to region segmentation involves growing regions from individual pixels using a stack-based approach. At any point, the pixels on the stack represent those from which future expansion of the region will take place. We begin by marking each pixel in the smoothed image as unexamined, and marking a single unexamined pixel as examined and placing it on the stack. We then repeatedly pop the topmost entry off the stack, and attempt to grow the region around it by examining the pixels immediately above and below, and to the left and right of that pixel. Each of these is tested to see if it is an edge pixel in the edge map, in which case it is ignored (allowing edges to form a strong boundary for regions). Each is also tested to see if it is a color match to the region being built, by summing the squares of the differences across all color channels. If this value falls below a defined threshold for error, the pixel is considered to be a part of the current region, and that pixel is placed on the stack to further extend the region; if not, the pixel is ignored. The threshold for color error is the mean color value of all pixels currently in the region, allowing the threshold to adapt as the region is grown. The algorithm terminates with a completed region once the stack is empty. A threshold is set on the acceptable size of a grown region, and if the region size falls below this level, the region is discarded.

To extend this algorithm to grow a set of regions, we must recognize two things: first, it should be possible for an area to be part of more than one region in initial stages, since an image will generally be factorable into regions in a number of different ways. Thus, the algorithm must allow any pixel to be potentially claimed by more than one grown region. Second, once we throw a region away as being too small, we do not wish to start growing other regions within this same area, as this has already proved unfruitful. Similarly, once we have defined a region, it will be more useful to start new regions outside that defined area.

Our initial approach was to begin searching the image for a non-visited pixel, growing a region using the algorithm described above (while marking each pixel as examined when it is placed on the stack), and then starting the next region by searching for an unexamined pixel. This approach is functional, but in practice, linear scanning wastes resources because many unsuccessful regions are attempted. We have found it more fruitful to begin with randomly selected points (20 for a 320 x 240 image), selecting the location of each after regions have been grown from all previous points.

We also attempt to merge regions based on degree of pixel overlap. Each region is examined with others that it abuts or overlaps, and regions are merged if one of two thresholds are exceeded. The first of these is the percentage of pixels that overlap - this value requires a significant overall similarity, and is generally most useful in merging small regions. For merging larger regions, the likelihood of a large percentage overlap is small, and so the threshold used is a total pixel overlap. By using overlap rather than color separation as a basis for merging, shadows can be properly joined to the objects that cast them, for example, or glare to the objects the glare is placed upon, without having to set an excessively high color threshold.

At this point, we have a collection of strong regions in each of the two images (the top stereo pair in Figure 2). Each region is represented by a map between the original image and the region (a set of boolean pixels where each 1 indicates a pixel present in the image), as well as a set of region attributes: its size, mean colour value, centroid, and a bounding box.

3.5 Region Simplification

The next step in providing useful visual information to a robotic rescue agent is the matching of regions between a pair of stereo images. This, however, is a complex process that can easily consume a great deal of the limited computational resources available. Our initial stereo matching process involved examining

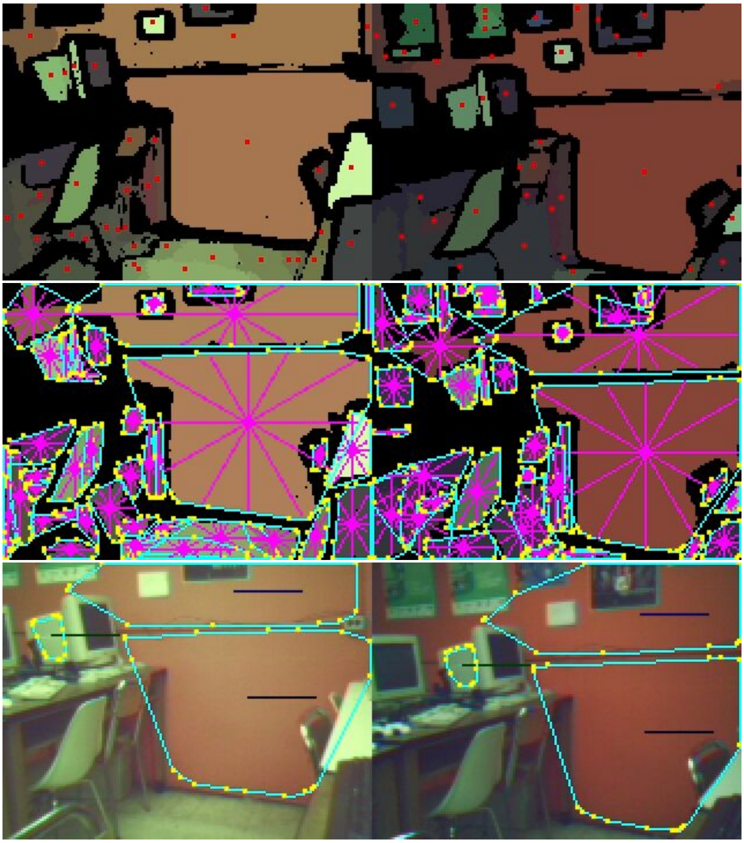


Fig. 2. Segmented regions (top), with convex hulls plotted and distance lines from the centroid added (middle). Stereo-matched regions (bottom) are bounded by a colored box, with a line emanating from the centroid of the image.

all pixels that could possibly represent the same region across the stereo pair, requiring checking for a match between hundreds of pixels for each potential match. We have considerably simplified this process by simplifying the structure of the regions themselves, allowing us to match a much smaller set of data. This process is analogous to smoothing noise out of an image before looking for edges.

We simplify regions by generating a convex hull for each, allowing us to replace the set of points outlining the region with a simpler set describing a polygon P, where every point in the original point set is either on the boundary of P or inside it. We begin with the boolean grid depicting each image. The exterior points along the vertical edges (the start and end points of each row) are used to generate a convex hull approximating the region using Graham's Scan [6]. We form a representation for the convex hull by drawing radial lines at 5 degree intervals, with each line originating at the centroid of the region and extending to the hull boundary. The length of each such line is stored, allowing an array of

72 integers to describe each region. The middle stereo pair in Figure 2 illustrates the result of this simplification process.

3.6 Stereo Matching

Once an image has been segmented into regions and simplified, regions must be matched across stereo images. Before simplifying regions, our original approach was limited in that it required superimposing region centroids and matching pixels. This was particularly troublesome for large regions. With convex hull simplification, however, the efficiency of matching can be greatly improved. With each convex hull, the very first stored value represents the distance from the centroid to the hull boundary at the 0-degree mark. A comparison of the similarity of two regions can then be easily performed by summing the squares of the differences of the values in the 72 corresponding positions in the two arrays (implicitly superimposing the centroids). Beyond greatly decreasing the number of individual points to match, this representation allows time required to make a comparison independent of region size. There is no particular threshold to a match - each region is matched to its strongest partner in the corresponding stereo image. We do, however, constrain matches for the purposes of maintaining accuracy by forcing a match to be considered only after its appearance in three successive video frames. This is particularly useful for noisy and poorly lit environments such as USAR. The bottom stereo pair in Figure 2 illustrates the matching of three regions between the raw stereo sample pair. The lines in each region are used as an indication to a teleoperator the angle that one would region have to be oriented to match the orientation of the other. That is, straight horizontal lines require no reorientation. The use of these lines will be explained momentarily.

Since we are matching regions without regard to the location in the visual frame, similar regions can be matched despite unreasonable spatial displacement. This is equally true without employing convex hulls, and is part of the nature of this domain. Because the robot is moving over very uneven terrain, cameras are likely poorly calibrated, and as the domain is unpredictable, we cannot make strong assumptions about the position of a region in each of a pair of stereo images. If this were employed in a domain where such assumptions could be made, the process could be made more accurate by strongly constraining the distance between potential matches in regions in a stereo pair, thereby lowering the number of potential matches that would have to be considered.

4 Performance

This system has been implemented and tested using an unmodified area in the Computer Science department at the University of Manitoba. *Spike*, the robot used in this project, is a one-sixth scale model radio-controlled toy car with rear wheel drive (See Figure 3). The radio controller has been modified to allow the vehicle to be controlled through the parallel port of any PC. The PC used, a 533MHz C3 Eden VIA Mini-ITX, with a 256Mb flash card, is carried in the interior of the vehicle. For this hardware configuration, we developed our own

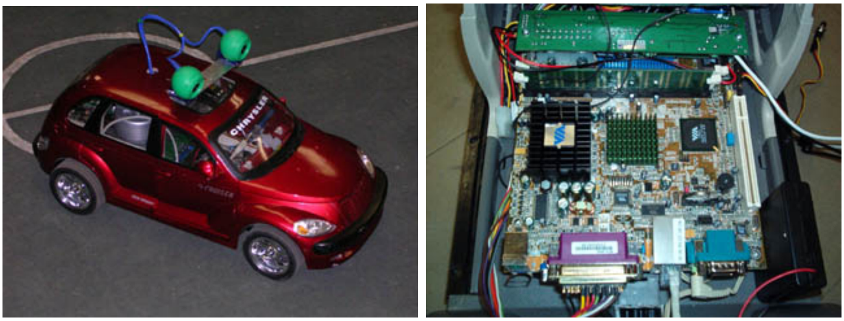


Fig. 3. Spike, the mobile robot used for this experiment

miniaturized version of the Debian Linux distribution, refined for use on systems with reduced hard drive space. The vision hardware consists of two USB web cameras capable of capturing 320 by 240 pixel images. The cameras are mounted on a servo that allows the stereo rig to pan in a range of plus or minus 45 degrees.

Figures 2 and 4 illustrate the matching abilities of this system. The raw image shown previously results in 40 regions for the right image and 42 regions for the left (including regions carried forward from previous frames). For applications in teleautonomous vehicle control, we currently display to the operator only the three strongest stereo matches present, in order to provide useful information without cluttering the camera view. In the first sample match (the bottom image pair in Figure 2), the three strongest image matches (as indicated by the colored boxes surrounding the matched pairs) are all correct, despite the offset in images due to the camera angle. Lighting in the area varies over time, resulting in changes in the matched pairs. The second sample match (the top image pair in Figure 4) illustrates a mismatch between two similarly shaped hulls. The line emanating from the image on the right hand side indicates the angle by which the right camera would have to be rotated for these shapes to match with a similar orientation, allowing a teleoperator or autonomous system to judge the likelihood of such an occurrence. The third image set (the bottom image pair in Figure 4) illustrates a similar correct match, but with one different stereo pair forming one of the three strongest matches.

We have observed very good improvement through the use of region simplifications with no decrease in match accuracy. With all vision processing turned off using the computational resources described above (that is, running only the other control functions), we achieve a capture rate of 2.5-2.9 frames per second. Activating region matching (including all phases described above) using convex hulls as the basis for a match results in a frame rate of 2.3-2.5 fps, while not employing convex hulls results in a frame rate of only 1.5-1.8 fps. We are currently investigating methods for speeding up this process further. In particular, the matching of regions is computationally expensive because the stereo system is entirely uncalibrated at the moment. Therefore, each region must be compared

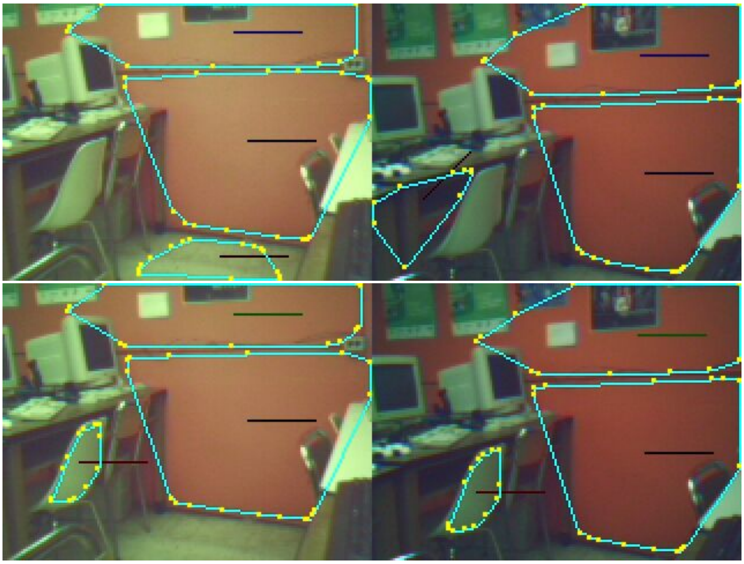


Fig. 4. Demonstration of matching in an unknown environment. The top pair shows a spatially incorrect match, while the bottom pair shows a different match than Figure 2.

against all other regions. By adding at least a rough calibration, some of these comparisons can be avoided: for example, matching regions with extremely large of spatial disparities are unlikely even in a USAR environment.

5 Conclusion

This paper has described our approach to stereo vision matching, which forms the basis of visual perception for both autonomous and teleoperated robots. With the region-based object extraction and stereo matching implemented, the ground work is laid for the use of higher level facilities employing stereo vision. These include 3D scene interpretation, mapping, localization, and autonomous control, some of which we have already employed in systems that use single-camera vision [3, 2].

The next step in this ongoing development is to include elements of camera calibration suitable for the USAR domain. The goal is to design a self-calibrating system that can produce the Fundamental Matrix without human interaction [10]. The Fundamental Matrix allows the object matching search to be constrained to a single line, rather than the entire image. This will improve the running time and accuracy of the stereo pair matching process. Parallel to this, we intend to replace the elements of vision based sensing in our autonomous systems with stereo vision using this approach. This will involve taking the matched stereo pairs and calculating the distance to each object found in the images by measuring the disparity observed in each image. Once the set of objects have a distance associated with them, these will serve as the basis for map generation (which is currently performed by using ego-motion detection across the entire image), which in turn will support better localization and path planning.

The research presented in this paper represents a core component in the development of vision to support autonomous and teleoperated processing in complex domains such as USAR. It is also applicable to many other domains, and indeed, will be even more efficient in domains where assumptions about lighting, color calibration, and predictability in the environment can be made.

References

1. Stefania Ardizzoni, Ilaria Bartolini, and Marco Patella. Windsurf: Region-based image retrieval using wavelets. In *DEXA Workshop*, pages 167–173, 1999.
2. Jacky Baltes and John Anderson. A pragmatic approach to robot rescue: The keystone fire brigade. In *Proceedings of the AAAI Workshop on Rescue Robots*, 2002.
3. Jacky Baltes, John Anderson, Shawn Schaerer, and Ryan Wegner. The keystone fire brigade 2004. In *Proceedings of RoboCup-2004*, Lisbon, Portugal, 2004.
4. Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. A sound algorithm for region-based image retrieval using an index. In *DEXA Workshop*, pages 930–934, 2000.
5. Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.
6. Thomas H. Cormen, Charles E. Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2nd edition, 2001.
7. Hiroshi Ishikawa and Ian H. Jermyn. Region extraction from multiple images. In *Eigth IEEE International Conference on Computer Vision*, July 2001.
8. Tony Lindeberg. Scale-space: A framework for handling image structures at multiple scales. In Egmond aan Zee, editor, *Proc. CERN School of Computing,*, September 1996.
9. David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.
10. Quang-Tuan Luong and Olivier Faugeras. The fundamental matrix: theory, algorithms, and stability analysis. *The International Journal of Computer Vision*, 17(1):43–76, 1996.
11. Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *I. J. Robotic Res*, 21:735–760, 2002.
12. F. Waltz and J. Miller. An effecient algorithm for gaussian blur using finite-state machines. In *SPIE Conf. on Machine Vision Systems for Inspection and Metrology VII*, 1998.

An Application Interface for UCHILSIM and the Arrival of New Challenges

Juan Cristóbal Zagal, Iván Sarmiento, and Javier Ruiz-del-Solar

Department of Electrical Engineering, Universidad de Chile,
Av. Tupper 2007, 6513027 Santiago, Chile
{jzagal, isarmien, jruizd}@ing.uchile.cl
http://www.robocup.cl

Abstract. UCHILSIM is a robot simulator recently introduced in the RoboCup Four Legged League. A main attractive of the simulator is the possibility of reproducing with accuracy the dynamical behavior of AIBO[1] robots as well as providing good graphical representations of their surroundings on a soccer scenario. Learning over virtual environments can be performed with successful transfers of resulting behaviors to real environments. Previous version of the simulator had a major drawback: Only the UChile1 team could make use of it since the developed system had high dependency on the team code. In this paper we present results of a development work which was envisioned on the first presentation of UCHILSIM; an application interface for allowing any OPEN-R software code to be directly used over the UCHILSIM simulator. The possibility of having this kind of tool opens a great field of developments and challenges since more people will develop OPEN-R software, even without having the robotic hardware but the simulator. Other recent improvements on our simulator are briefly presented here as well.

1 Introduction

Simulation is becoming a hot topic on robotics. An increase in the number of simulation related publications can be observed over the main journals of the field. New robotics simulators are emerging into the field either for research or commercial applications, ranging from general purpose simulators to specifics to certain robotic task, such as grasping or arm soldering trajectory design. New developments on robotic simulators are being recognized by the research community, examples of these are the robotic grasping simulator GraspIt! [7] that won the NASA Space Telerobotics *Cool Robot of the week* prize on February 2002, the general purpose robotic simulator WebBots [6] that recently won the second place on of EURON[2] *Technology Transfer Award*, and a publication related to our UCHILSIM simulator won the *RoboCup Engineering Challenge Award* on RoboCup 2004[3].

[1] AIBO and OPEN-R is a trademark or registered trademark of Sony Corporation.

[2] EURON is the European Robotics Research Network, http://www.euron.org

[3] The robot soccer world federation http://www.robocup.org

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 464 – 471, 2006.

Under our perspective the main attractive of simulation in robotics is to enhance learning, although this doesn't seems to be the main reason for its current expansion on the field. Industrial robots on the automotive industry for example offer simulators of their products which allow users to get familiar with them by practicing their kinematics before running the real hardware. Researchers use simulators for testing new approaches on arbitrary scenarios and to use multiple robot agents without having the actual hardware. Applications of simulation on this field are very broad, and the generation of good simulators is being enforced by the increase on computer power as Moore's law establish [8], as well as the exponential improvements on graphics hardware power [4]. Main criticism to robot development under simulation given by Brooks [2] several years ago cannot still be defended while facing the current achievements of computer simulations. But overall a main justification to fight towards simulation on robotics is given by the emerging supporting theories such as the Theory of Mind [9]. The authors have also proposed some theoretical basis such as Back to Reality [14].

Towards Improving Our Simulator

UCHILSIM [12] is a dynamics robotic simulator introduced for the RoboCup Four Legged League; the simulator reproduces with high accuracy the dynamics of AIBO motions and its interactions within a soccer scenario. The simulator has shown to be a useful tool for learning into virtual environments with successful behavioral transfers to reality. In [13] experiments are shown on the generation of dynamics AIBO gaits form simulation to reality, in [14] experiments are presented about learning to kick the ball using the Back to Reality approach and the UCHILSIM simulator.

We believe that this is a relevant kind of tool to promote on future developments of RoboCup. Aiming at improving further our simulator such that it can become a general use platform for the four legged league, we have generated a list of main requirements to fulfill by a simulator: (1) Use a generic and flexible definition of robots, (2) allow to incorporate other user defined objects into the simulation, (3) allow multiple robots to share a common virtual environment, (4) use of different robotics platforms, (5) use a fast and realistic dynamics engine, (6) provide good graphics and visualization methods for the desired task, (7) use a fast and robust collision detection system, (8) provide good programmatic interfaces in order for anybody to use the system, and (9) run over multiple host platforms. Prior to this publication UCHILSIM satisfied points 1 to 7, however there was no programmatic interface for allowing any generic OPEN-R software to run over UCHILSIM. This paper deals precisely with this point. Here we present an application interface for the UCHILSIM simulator which will allow spreading the use of this tool.

The reminder of this paper is as follows, section 2 present the implemented interface for UCHILSIM, section 3 present examples of using and testing this interface, section 4 discuss possible applications of this tool, on section 5 we describe briefly some recent improvements on the simulator and finally on section 6 we present conclusions and envision future challenges for this system.

2 An Application Interface for UCHILSIM

The UCHILSIM simulator has been restricted to the use of the UChile1 RoboCup four legged team. This restriction was expressed on the form of several code dependencies among the simulator and the team source code. In order for any OPEN-R developer to make use of the simulator it would have involved rewriting a large amount of code for each particular application.

The idea of building an interface for the simulator was announced on the first UCHILSIM publication [12], however there were just some ideas at that time. Among these ideas we considered first to construct a simulator programmatic interface by writing a large number of primitive functions for accessing the simulated hardware similarly as one does when using the OPEN-R Sony Software Development Kit [10]. Writing such programmatic interface would have been almost equivalent to generate a complete SDK for our simulator. A main drawback of this approach is that it would involve for any user to rewrite its particular application using the set of functions provided by a parallel SDK. Fortunately we found another alternative at a lower level, before going into its details we should describe briefly the OPEN-R SDK for which it was implemented.

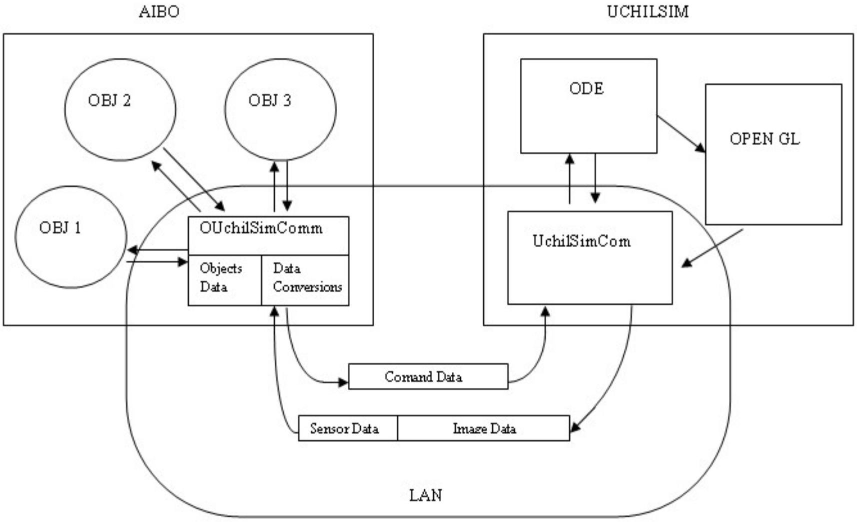


Fig. 1. Diagram showing how the OUChilSimComm and the UChilSimComm interface objects exchange command, sensor and image data trough the network

2.1 Description of the Target SDK

The OPEN-R SDK is an interface proposed by Sony in order to promote the developments of robots software and hardware, refer to [10]. The interface enhances the development of modularized pieces of software which are called OPEN-R objects.

The objects are implemented as independent processes which run concurrently and intercommunicate by means of messages. The connections among objects are described by communication services described on a boot time readable file. This is a very important characteristic since it allows objects to be replaceable components at an operative level.

Under OPEN-R the interface to the system layer is also implemented by means of inter object communication. There is a specific object provided on OPEN-R called *OVirtualRobotComm* which is in charge of providing a low level driver interface of data with the robot hardware by means of exchanging command, sensor and image data with other objects, this relation is established trough the same configurable communication service file.

2.2 Description of the New Interface

The idea is to replace the low level object interface OVirtualRobotComm provided under OPEN-R by another OPEN-R object designed for interfacing data with a simulated robot under UCHILSIM instead of a real robot. The interface object that we have developed is called *OUChilSimComm*. This object is designed to run either over an AIBO robot or on a host computer by using OPEN-R Remote Processing [10]. Although this object runs embedded in the space of OPEN-R objects, it should interchange data with the UCHILSIM simulator which runs on a host computer. This communication is performed by network TCP/IP connection among OUChilSimComm and an interface developed at the simulator side. We call this interface as *UChilSimComm*. Figure 1 shows a diagram of the relations among these interface modules. Command data is collected at the OUChilSimComm module and then dispatched to the simulator across the network; similarly sensor data and image data are packed by the UChilSimComm interface at the simulator side using fixed sized data structures. Then data is exchanged using TCP/IP connections either across platforms or over the same host machine (using the local host IP). There are many choices for implementing that since OUChilSimComm runs over the robot or on a host computer as well as the other OPEN-R modules.

Data Structures and Packets: The interface between the module and the simulator uses two different and independent network connections, one for the sensor and image data and another for the command data. Data packets used for image and sensor data are of fixed length while packets used for transmitting command data are of variable length. OUChilSimComm maintains a buffer of sensor and image data which is constantly updated with data coming from the simulator. This data is dispatched to the calling objects as requested. The command data which is received from the objects is immediately dispatched to the simulator. The following is a description of each data flow and how the structures are arranged. This structures slightly differ depending on the robot model being used (ERS7/ERS210).

Sensor Data: The digital sensor data is generated at UCHILSIM with a similar rate as the existing on the real robot. After each dynamic integration step, the actual joint sensor values are collected from the virtual robot joints. Simplistic values are given to the acceleration sensors, as well as for the switch sensors. A data transmission packet is filled with all these values containing a header with timestamps related to the data. When the packet is received by the OUChilSimComm object a *OSensorFrameVectorData* OPEN-R structure [10] is constructed by calling the corresponding data constructor provided with OPEN-R, and then filling the corresponding fields with the incoming data. If this data structure is requested by other objects then it is dispatched, otherwise the data is stored into a limited size buffer.

Image Data: The digital image data is generated at UCHILSIM with a similar rate as the existing on the robot camera. After each new YUV image frame is acquired from the simulator a data structure equal to *OFbkImage* [10] is generated, and then the data is split into three packets for the simplicity of network transmission, each packet contains their corresponding time stamps and sequence identifiers. When the packet is received by the OUChilSimComm object the OFbkImage structure is reconstructed and then the *OFbkImageVectorData* [10] structure is updated by directly incorporating OFbkImage data. The OFbkImageVectorData is constructed by using an existing OPEN-R constructor.

Command Data: The commands are generated by any running OPEN-R object and then transmitted to the OUChilSimComm module. The OPEN-R data structure which contains these data is *OCommandVectorData* [10]. Once this structure is received the task of OUChilSimComm is to extract the joint command reference values and timing data, disregarding any LED command. Then a transmission packet is generated containing a header which indicates command type, number of data frames and timing data. When the packet arrives UCHILSIM (trough UChilSimComm interface), the corresponding joint commands are executed as position references for the motors located at each joint, these reference values are taken by the corresponding PID controller located at each simulated joint.

3 Using and Testing the Interface

As it can be seen the interface is implemented at the system level rather than at the programmatic level, and therefore the developers don’t need to perform modifications on their code, just to re define the communication services and to recompile their own code to the host computer in case this is desired to be used. The user should modify the stub.cfg file replacing the OVirtualRobotComm service connections with the OUChilSimComm service. Then on the target directory it should make sure that CONECT.CFG file contains the right connections.

From the interface side, a configuration file should be updated indicating the corresponding network connections where the robot and UCHILSIM process are located. It should also be specified the corresponding robot model.

The presented interface has been successfully tested with the simple source code examples provided with OPEN-R, such as *MovingLegs7* [10]. The test was performed running all processes on a single host computer (2.5 GHz processor, 512 Mb ram, no graphics accelerator). We have also tested our own source code by applying simple vision related tasks such as ball following behaviors.

4 Possible Applications of This Tool

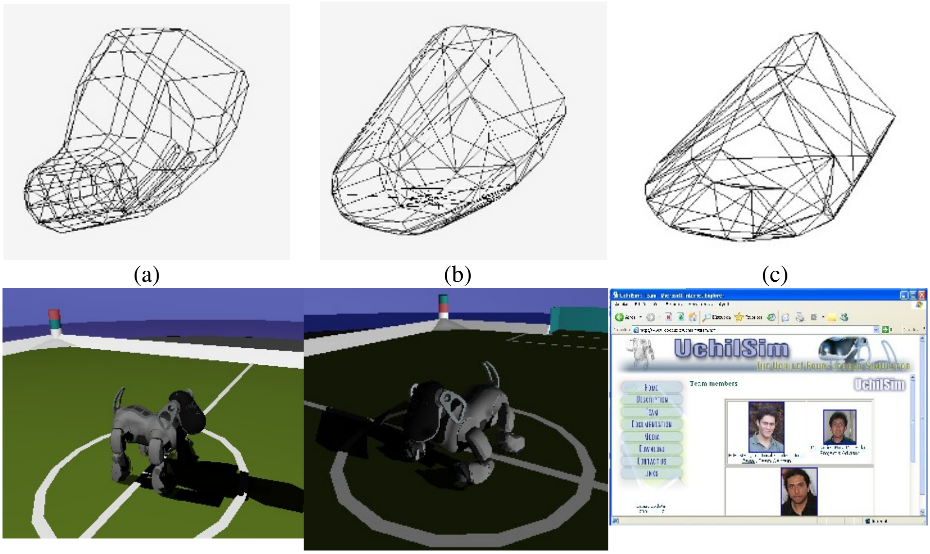
We believe that one of the major applications of this type of tool is that it will allow more developers to enter RoboCup and in general to program software for robots by directly using OPEN-R without having to access directly the robot. Another main advantage of this tool is to accelerate developments into the four legged league by providing a standard test bed for new ideas. Certainly with this simulator it is no longer necessary to worry about destroying the hardware or even about the long time required for specific experiments. Since any OPEN-R code can be used, with this application interface team code can be evaluated using more statistically proven strategies such like making two teams to compete for very long trials, this kind of tool can be established as a standard way of testing code prior to real competitions for example. It is also important to test and to compare specific parts of team code especially given the trend of the league on the modularization and specialization on the functions of vision, localization and strategy. Since the interface is based on the idea of modular objects which can communicate along a network it is possible to have extensive distributed systems which share a common dynamical environment. This can be even extended to the use of Internet. The idea of parallelism can be exploited further and we can use parallel computing for example for evolving team behaviors.

5 Other Recent Improvements on UCHILSIM

Multitasking and Task Scheduling: An important limitation of the simulator was the high processor consumption due to graphics. Therefore we have implemented separated processing threads on the simulator, one is specific for the graphics and the other is specific for the dynamical computations. This allows us to have always fast dynamics while having just a best effort result over the graphical representation. A consequence of this is that the graphics seen by the user might appear blinking when the window size is too large; however there is always good speed for the dynamics. Eventually the graphic representation can be totally disconnected from the simulation; this might be useful for experiments on which there is no need of having the robot camera, such as offline locomotion learning tasks. Another implemented alternative is to have hard control of the different tasks such as graphics and dynamics computations which should be executed. In this respect we have implemented a task scheduler which allows us to control specific timings for the different tasks.

Graphics and Mesh Improvements: We have incorporated the computer graphics technique of Shadow Volumes [11]. The attractive of this technique is that it allows producing precise shadows on real time. Since overlapping shadows generate areas of varying intensities this allows to reproduce the effect that can be observed when we have strong sources of light over the soccer scenario; the robot produce shadows in the directions opposite to the different sources of light. Figure 2 shows some examples of using this technique. Other improvements that we have introduced, proposed in [1], are CMOS filters and introduction of camera aberration over the AIBO lenses.

We have implemented an tested also a set of offline tools for optimizing the robot meshes which are provided by Sony. In particular the interest for the simulator is to have representative and simple shapes for collision detection. By using the quick hull algorithm[4] in conjunction with the GLOD library [3] we are able to considerably reduce the amount of points which are used for describing a given shape. Figure 2 shows result of applying this tool, a given limb segment originally consist of 178 vertex, after applying convex hull we get a model of 84 vertex, finally after applying GLOD tools we get just 68 vertex on our model.



(a) (b) (c)

(d) (e) (f)

Fig. 2. Mesh improvements on a portion of the ERS-7 robot leg, on (a) the original mesh is presented, (b) is the result of computing the convex hull of the mesh and finally (c) shows the result of reducing the level of detail by using GLOD library tools. On (d) and (e) examples are shown of the implementation of the Shadow Volumes CG technique. On (f) a screenshot of the new website of UCHILSIM is shown.

6 Conclusions and Further Challenges

It was presented an application interface for the UCHILSIM simulator. We envision the arrival of new challenges related to the optimization of this tool and the use that other teams might give to it. From our team perspective there are still some tasks to fulfill; these are the development of a networking interface, improvement of sensor models and probably a sound interface. People at our group for example is currently quite motivated on performing experiments with distributed simulation using parallel computing for producing behavioral learning over large search spaces.

[4] please refer to http://www.qhull.org

Acknowledgements

We thank Paul Vallejos, Carlos Gortaris and Alvaro Neira for their collaboration. This research was partially supported by Departamento de Investigación y Desarrollo (Universidad de Chile) under project ENL 05/03.

References

1. Asanuma, K., Umeda, K., Ueda, R., Arai, T.:Development of a Simulator of Environment and Measurement for Autonomous Mobile Robots Considering Camera Characteristics In: 7th International Workshop on RoboCup 2003, Lecture Notes in Artificial Intelligence, Springer, Padua, Italy (2003)
2. Brooks, R.A.: Flesh and Machines: How Robots will Change Us. Phanteon, USA, February (2002)
3. Cohen, J., Luebke, D. Duca, N., Schubert, B.: GLOD: A Geometric Level of Detail System at the OpenGL API Level. IEEE Visualization 2003, Seattle, WA (2003).
4. Kelly, F., Kokaram, Anil.: Graphics hardware for gradient-based motion estimation. Embedded Processors for Multimedia and Communications, San Jose, California (2004) 92-103
5. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
6. Michel, O.: Webots: Professional Mobile Robot Simulation. In: International Journal of Advanced Robotic Systems, Vol. 1, Num. 1 (2004) 39-42
7. Miller, A., Allen, P.: GraspIt! A versatile simulator for Robotic Grasping, IEEE Robotics and Automation Magazine, December (2004) 110 -122
8. Moore, G.E.: Cramming More Components Onto Integrated Circuits. Electronics Journal, April 19 (1965)
9. Scassellati, B.: Theory of Mind for a Humanoid Robot. First IEEE/RSJ International Conference on Humanoid Robotics (2000)
10. Sony Corporation.: OPEN-R SDK Programer's Guide and Level2 Reference Guide. Published by Sony Corporation (2004)
11. Watt, A., Watt, M.: Advanced Animation and Rendering Techniques, Theory and Practice. Addison-Wesley, New York (1994)
12. Zagal J.C., Ruiz-del-Solar J.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: 8th International Workshop on RoboCup 2004, Lecture Notes in Artificial Intelligence, Springer, Lisbon, Portugal, (2004)
13. Zagal J.C., Ruiz-del-Solar J.: Learning to Kick the Ball Using Back to Reality. In: 8th International Workshop on RoboCup 2004, Lecture Notes in Artificial Intelligence, Springer, Lisbon, Portugal (2004)
14. Zagal J.C., Ruiz-del-Solar J., Vallejos, P.: Back to Reality: Crossing the Reality Gap in Evolutionary Robotics. IAV 2004 the 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal (2004)

Autonomous Parking Control Design for Car-Like Mobile Robot by Using Ultrasonic and Infrared Sensors

Tzuu-Hseng S. Li, Chi-Cheng Chang, Ying-Jie Ye, and Gui-Rong Tasi

IC^2S Laboratory, Department of Electrical Engineering
National Cheng Kung University, Tainan 70101, Taiwan, R.O.C.
thsli@mail.ncku.edu.tw, n2692415@ccmail.ncku.edu.tw

Abstract. This research presents the design and implementation of the intelligent autonomous parking controller (APC) and accomplishes it in a car-like mobile robot (CLMR). This car possesses the function to accept and estimate the environment by integration of infrared and ultrasonic sensors. We propose five parking modes including parallel-parking mode, a narrow path parallel-parking mode, garage-parking mode, a narrow path garage-parking mode, and none parking mode. And the CLMR can autonomously determine which mode to use and park itself into the parking lot. Finally, it is perceived that our intelligent APC is feasible from the practical experiments.

1 Introduction

In recent years, an increasing amount of the CLMR researches has focused on the problem of autonomously parking and avoidable collision. The parking problem of the car-like mobile robot consists of finding the parking lot and planning the trajectory for parking. Basically, the parking problems can be classified into two categories: garage-parking problem and parallel-parking problem [1-20]. In this research, we propose an autonomously parking control base on the fuzzy logic control (FLC). In this control we fuse the measurements of infrared sensor and ultrasonic sensor as the inputs of the FLC.

To solve the problem of the path-planning from the usage of sensor point of view, An et al. [8] develop an online path-planning algorithm that guides an autonomous mobile robot to a goal with avoiding obstacles in an uncertain world with a CCD camera, and Han et al. [21] use the ultrasonic sensor to build the environment about the car, follow a moving object and avoid the collision.

For parking control, [18] utilizes the CCD camera to detect the global vision of the parking lot, [19] adopts six infrared sensors to measure the relative distance among the CLMR and the surroundings, and [20] uses the sensor fusion techniques to combine the ultrasonic sensors, encoders, and gyroscopes with a differential GPS system to detect and estimate the dimensions of the parking lot. In this paper, we want to integrate the information of the ultrasonic and infrared sensors to measure the parking environment.

Fuzzy set theory is arisen from the desire of linguistic description for complex system and it can be utilized to formulate and translate the human experience. This

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 472 – 479, 2006.

kind of human intelligence is easily represented by the fuzzy logic control structure. Most advanced control algorithms in autonomous mobile robots can benefit from fuzzy logic control. In this paper, the CLMR equips with infrared and ultrasonic sensors. We fuse the measurements of the two kinds of sensors on the car to obtain the information in an unknown environment, and utilize this information to determine the velocity and the steering angle of the car by the proposed FLC.

This paper is organized as follows. In Section 2, parking lot measurement, kinematic equations, and the FLC for garage parking and parallel parking are derived. The parking lot measurement can help to decide the trajectory of the CLMR. Section 3 addresses the hardware architecture of the CLMR that consists of the following four parts, CLMR mechanism, FPGA module, sensor module, and electronic driver module. Experimental results about the intelligent APC in different modes are given in Section 4. Section 5 concludes this paper.

2 Design of APC

The main advantage of the study is that the developed APC can park the car successfully, though the absolute coordinates of the car and parking lot are unknown. Fig.1 presents the appearance of the CLMR and the top view and the sensor arrangement of the CLMR.

In the beginning, we address the fuzzy wall following control (FWFC) problem. By the FWFC in [19], the X_d and X_e are the inputs and ϕ is the output of the FLC. X_d is the distance between the CLMR and the wall that is defined in equation (1), and X_e is the sloping angle of CLMR that is described in equation (2). In this paper, we introduce a new variable X, which is the sum of X_d and X_e, to reduce the number of fuzzy if-then rules. The corresponding rule table is listed in Table 1. Fig. 2 indicates the member functions of the input and output of the FLC.

$$X_d = Right_1 - dis \tag{1}$$

$$X_e = Right_1 - Right_2 \tag{2}$$

$$X = X_d + X_e \tag{3}$$

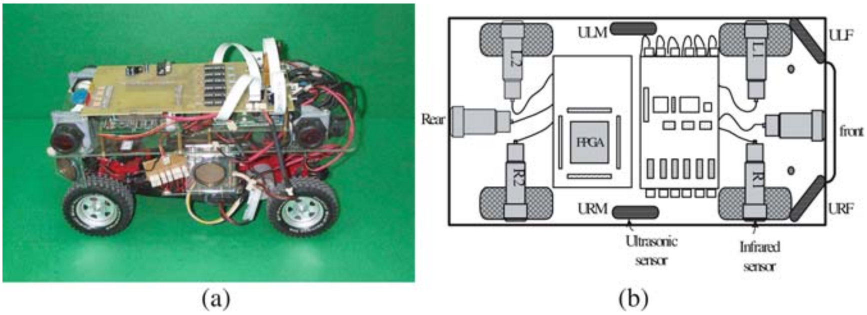
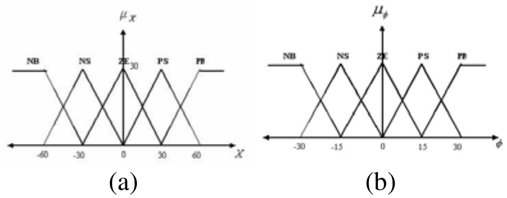


Fig. 1. (a) The appearance of the CLMR (b)Top view and the sensor arrangement of the CLMR

where $Right_1$ is the information of the right front infrared sensor, *dis* presents the safety distance of the CLMR, and $Right_2$ is the information from the right rear infrared sensor.



(a) (b)

Fig. 2. (a) Fuzzy membership function for the CLMR in put X (b) Fuzzy membership function for the CLMR output ϕ

Table 1. Fuzzy rule table of the steering angle

Antecedent part		Consequence part	
If X is	NB	then ϕ is	NB
	NS		NS
	ZE		ZE
	PS		PS
	PB		PB

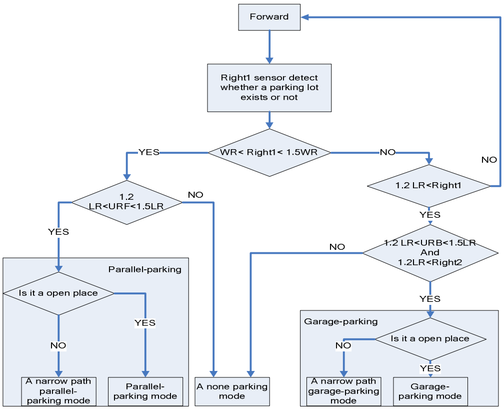


Fig. 3. Flow chart of mode detection

In order to autonomously park the CLMR, the parking lot detection is an important issue. In this paper, we consider five parking modes for the CLMR. The flow chart of mode detection is shown in Fig. 3, where the WR is the width of the CLMR and LR is the length of the CLMR. In fact, three parking conditions are included and will be examined as follows.

Case 1: Parallel-parking condition
The basic constraints are
(WR<Right 1<1.5WR) and (1.2LR< URF<1.5LR).

In this case, there are two modes should be considered. One is parallel-parking mode and the other is a narrow path parallel-parking mode. At first, we explain the parallel-parking mode, which is showing Fig. 4(a). For backward parking, the CLMR passes though the parking lot a little distance once the sensors have detected the parallel-parking lot. Then we turn the steering wheel to the right end and drive the car backwards. If the rear and Right 2 infrared sensors detect the CLMR is entering the lot then the CLMR will turn the steering wheel to straight until the front infrared sensor detect the wall. Then we make the steering wheel turn left until the rear sensor detect the wall is close to the safety distance or the car is parallel to the wall. Now, we discuss how to execute a narrow path parallel-parking mode. In this mode, we not only use the same parallel-parking mode but also consider the left sensors' information to avoid colliding with the wall. If both the left 1 and 2 sensors measure the distance between the left wall and the car is less than 0.7 WR, which means that there is not enough distance for the CLMR to turn round, and then we terminate the parking mission.

Case2: Garage-parking condition
The primary conditions are
(WR<Right 1<1.5WR) and (1.2LR< URM<1.5LR) and (1.5WR<Right 2)

In this situation we also consider two modes. One is the garage-parking mode, which is depicted in Fig. 5(a), and the other is the narrow path garage-parking mode. The detection of the garage-parking condition is accomplished by combining three sensors. At first, Right 1 sensor can detect whether a parking lot exists or not. And if both the measurements of the ultrasonic sensor URM and the infrared sensor Right 2 satisfy the constraint 1.2LR< URM<1.5L and 1.5WR<Right 2, then one can conclude that the parking lot is deep enough to execute the garage parking and there is not any obstacle or car in the parking lot. For garage-parking control, we turn the steering wheel left and move forward a little distance as the Right 2 sensor detects that the CLMR passes the parking lot. Then, we turn the steering wheel right and move backward to the garage. As the Right 2 and Left 2 detect the CLMR enters the garage then the CLMR turn the steering wheel to straight and move backward until the rear sensor detects the CLMR is close to the safety distance. If it is not in the center position then goes forwards and backwards by using the FWFC. For narrow path garage-parking control, the basic concept is the same as that of the narrow path

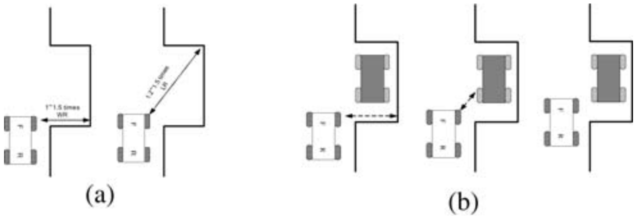


Fig. 4. (a) Parallel- parking mode. (b) None parking mode.

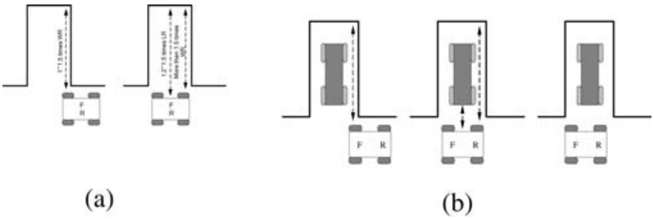


Fig. 5. (a) Garage- parking mode. (b) None parking mode.

parallel-parking control. That is, suppose the Left 1 and 2 sensors find that the distance between the left wall and the CLMR is less than 0.7 WR, then we stop the parking command.

Case 3: None parking condition
It is seen in Fig. 4(b) and Fig. 5(b) that the CLMR notice there is a car in the parking lot. In these cases, we call the CLMR is in none parking condition. We stop the car once it passes though the parking lot.

3 Hardware Architecture

The CLMR consists of the following four parts, CLMR mechanism, FPGA module, sensor module, and electronic driver module. The CLMR mechanism includes car body, driving motor, and steering motor. The car body is a 1/10th scale four-wheeled mobile robot vehicle with rear wheel drive system and front steering wheels. The robot carries FPGA board, daughter board and battery.

The FPGA module is the Altera FLEX 6000 family. The Altera FLEX 6000 programmable logic device (PLD) family provides a low-cost alternative to high-volume gate array designs. The FPGA utilizes these information come from the sensor module to determine the velocity and the steering angle of the car.

The sensor module contains six infrared sensors and four ultrasonic sensors. All of them are reflection sensors, so we need the reflective object. The specifications of the infrared sensor (UF55MG) are: measurement range 50~500mm, input volt 20~30V, output volt 0~10V, response time 10ms, and opening angle 12°.

Because the measurement rang of the infrared sensor is only 50mm~500mm, we can not measure obstacle out of this range. For this reason, the ultrasonic sensors (6500 Sonar Module) are adopted in the CLMR. The 6500 Series is an economical sonar ranging module, with a simple interface, is able to measure distances from 6 inches to 35 feet.

The electronic driver module consists of a STL293D DC motor driver IC and a FPGA. The STL293D IC is assembled in a PCB board to drive the DC motor to change the speed of the rear wheels. We employ the FPGA to generate PWM signal to drive the DC servomotor to control the steering angle of the front wheels.

4 Experiment

Fig. 6 shows that the developed CLMR can detect there is a parallel-parking place and perform the parallel-parking control successfully. Fig. 7 illustrates the CLMR can

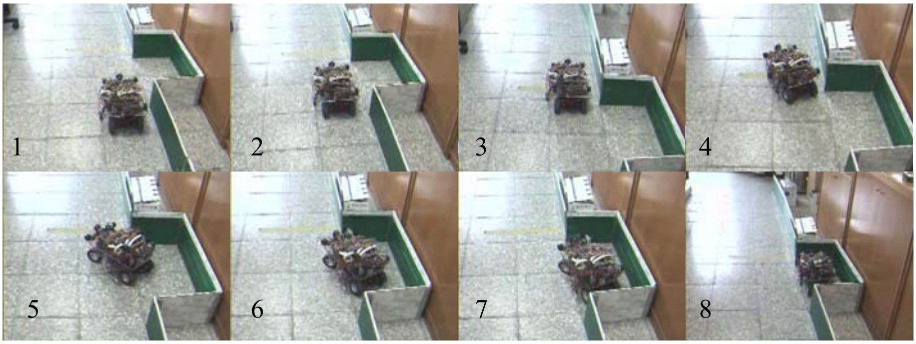


Fig. 6. Experimental results of the parallel-parking control

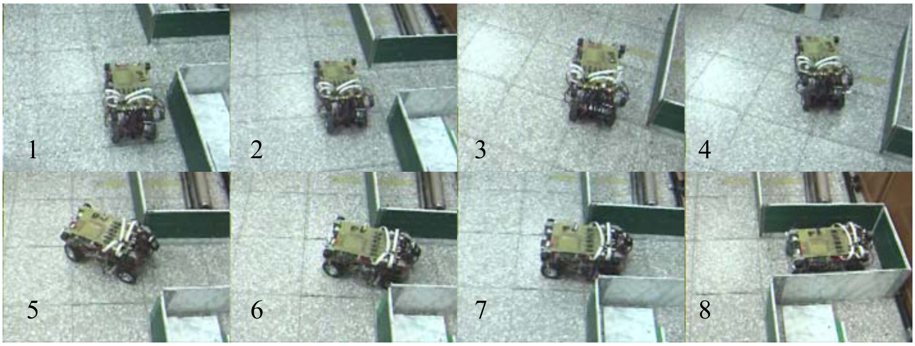
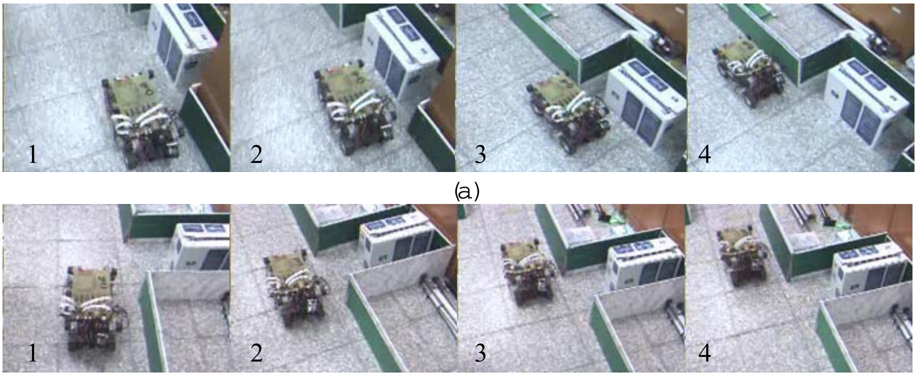


Fig. 7. Experimental results of the garage-parking control



(a)

(b)

Fig. 8. None parking case: (a) Obstacle in parallel-parking lot, (b) Obstacle in garage-parking lot

detect the existence of a garage-parking place and successfully execute the garage-parking control. Experimental results depicted in Fig. 8 demonstrate that there are obstacles in the parking lot, and then the CLMR can just pass the parking lot and stop.

5 Conclusion

In this paper, we have succeeded in solving the parking problems by the proposed intelligent APC on the basis of the infrared and ultrasonic sensors. Five parking modes have been developed in the APC and realized in the FPGA chip. The developed scheme can be also applied to a real car if it equips with these sensors and the FPGA or micro processors. For future study, we want to set up the CMOS sensor on the CLMR. The CMOS sensor does not need the reflective object and can get more information about the environment.

Acknowledgment

This work is supported by the National Science Council, Taiwan, Republic of China, under grant NSC 93-2213-E-006-129.

Reference

1. M. Sugeno and K. Murakami, "An experimental study on fuzzy parking control using a model car," in Industrial Applications of Fuzzy Control, M. Sugeno, Ed. North-Holland, The Netherlands, 1985, pp. 105–124.
2. M. Sugeno, T. Murofushi, T. Mori, T. Tatematsu, and J. Tanaka, "Fuzzy algorithmic control of a model car by oral instructions," *Fuzzy Sets Syst.*, vol. 32, pp. 207–219, 1989.
3. A. Ohata and M. Mio, "Parking control based on nonlinear trajectory control for low speed vehicles," in *Proc. IEEE Int. Conf. Industrial Electronics*, 1991, pp. 107–112.
4. S. Yasunobu and Y. Murai, "Parking control based on predictive fuzzy control," in Proc. IEEE Int. Conf. Fuzzy Systems, vol. 2, 1994, pp. 1338–1341.
5. W. A. Daxwanger and G. K. Schmidt, "Skill-based visual parking control using neural and fuzzy networks," in Proc. IEEE Int. Conf. System, Man, Cybernetics, vol. 2, 1995, pp. 1659–1664.
6. A. Tayebi and A. Rachid, "A time-varying-based robust control for the parking problem of a wheeled mobile robot," in Proc. IEEE Int. Conf. Robotics and Automation, 1996, pp. 3099–3104.
7. M. C. Leu and T. Q. Kim, "Cell mapping based fuzzy control of car parking," in Proc. IEEE Int. Conf. Robotics Automation, 1998, pp.2494–2499.
8. H. An, T. Yoshino, D. Kashimoto, M. Okubo, Y. Sakai, and T. Hamamoto, "Improvement of convergence to goal for wheeled mobile robot using parking motion," in Proc. IEEE Int. Conf. Intelligent Robots Systems, 1999, pp. 1693–1698.
9. D. Lyon, "Parallel parking with curvature and nonholonomic constraints," in Proc. Symp. Intelligent Vehicles, Detroit, MI, 1992, pp. 341–346.
10. I. E. Paromtchik and C. Laugire, "Motion generation and control for parking an autonomous vehicle," in Proc. IEEE Conf. Robotics Automation, vol. 4, Minneapolis, MN, 1996, pp. 3117–3122.

11. C. Laugier, T. Fraichard, I. E. Paromtchik, and P. Garnier, “Sensor-based control architecture for a car-like vehicle,” in Proc. IEEE Int. Conf. Intelligent Robots Systems, 1998, pp. 216–222.
12. K.Y. Lian, C. S. Chin, and T. S. Chiang, “Parallel parking a car-like robot using fuzzy gain scheduling,” in Proc. 1999 IEEE Int. Conf. Control Applications, vol. 2, 1999, pp. 1686–1691.
13. K. Jiang and L. D. Seneviratne, “A sensor guided autonomous parking system for nonholonomic mobile robots,” in Proc. IEEE Int. Conf. Robotics Automation, vol. 1, 1999, pp. 311–316.
14. K. Jiang, “A sensor guided parallel parking system for nonholonomic vehicles,” in Proc. IEEE Conf. Intelligent Transportation Systems, 2000, pp. 270–275.
15. J. Xiu, G. Chen, and M. Xie, “Vision-guided automatic parking for smart car,” in Proc. IEEE Intelligent Vehicles Symp., 2000, pp. 725–730.
16. D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg, “Neural network architecture for trajectory generation and control of automated car parking,” IEEE Trans. Contr. Syst. Technol., vol. 4, pp. 50–56, Jan. 1996.
17. S. Lee, M. Kim, Y. Youm, and W. Chung, “Control of a car-like mobile robot for parking problem,” in Proc. IEEE Int. Conf. Robotics Automation, 1999, pp. 1–6.
18. T.-H. S. Li and S.-J. Chang, “Autonomous fuzzy parking control of a car-like mobile robot,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 33, pp. 451–465, 2003.
19. T.-H. S. Li, S.-J. Chang, and Y.X. Chen, “Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot”, IEEE Trans on Industrial Electronics, vol. 50, NO.5, pp. 867-880, 2003.
20. I. Baturone, F. J. Moreno-Velo, S. Sánchez-Solano, and A. Ollero, “Automatic design of fuzzy controllers for car-like autonomous robots” IEEE Transactions on Fuzzy Systems, vol. 12, NO. 4, pp 447-465, 2004.
21. Y. Han, M. Han, H. Cha, M. Hong, and H. Hahn, “Tracking of a moving object using ultrasonic sensors based on a virtual ultrasonic image” Robotics and Autonomous Systems, vol.36 pp. 11–19, 2001.
22. C. S. Ting, T. H. S. Li, and F. C. Kung, “An approach to systematic design of the fuzzy control system,” Fuzzy Sets Syst., vol. 77, pp. 151–166, 1996.
23. T. H. S. Li and M. Y. Shieh, “Switching-type fuzzy sliding mode control of a cart-pole system,” Mechatronics, vol. 10, pp. 91–109, 2000.

Behavior-Based Vision on a 4 Legged Soccer Robot

Floris Mantz, Pieter Jonker, and Wouter Caarls

Quantitative Imaging Group, Faculty of Applied Sciences,
Delft University of Technology, Lorentzweg 1, 2628 LC Delft,
The Netherlands
{floris, pieter, wouter}@ph.tn.tudelft.nl

Abstract. In this paper the architecture of a 4 legged soccer robot is divided into a hierarchy of *behaviors*, where each behavior represents an independent sense-think-act loop. Based on this view we have implemented a *behavior-based vision system,* improving performance due to *object-specific* image processing, *behavior-specific* image processing and *behavior-specific* self localization. The system was tested under various lighting conditions, off-line using sets of images, and on-line in real tests for a robot in the role of goalkeeper. It appeared hat the performance of the goalie *doubled*, that it could play under a wider range of lighting and environmental conditions and used less CPU power.

1 Introduction

Soccer playing robots are *the* playground to gain experience with embodied intelligence. The software architectures of those robots can serve as an example for more complex embedded systems. Those designs are often built around a single sense-think-act cycle, as presented in Fig. 1a. Here, e.g. an image processing group solves the *sense* task, the control theory group solves the *act* task, an AI group solves the *think* task, whereas software and mechanical engineers take the responsibility over the overall software and mechanical hardware design and maintainability. After an initial architecture phase, interfaces are quickly established and all groups retract to their own lab to solve their part of the problem, thereby often making assumptions what is or should be done by the others. Data is "thrown over the wall" to the others who have to cope with it. As those embedded systems increase in complexity over the years, the software and hardware complexity grows, and all groups start to make their system part versatile and robust and hence complex. Then, to cope with that, they start to locally optimize their parts, making sources even more obscure. From 1991 onward it was suggested [1,2,3] that a different layered modular architecture should be followed in the sense that higher layers control the lower layers, by invocation actions from the lower layers or by promoting or suppressing behaviors from the lower layers, i.e., on the lowest level on the hardware devices. To program all behaviors of a system that - without tedious recalibration - functions under al circumstances on any setting, is like maintaining a house of cards. One cannot foresee all possible states in which the system might end up in the future.

With growing complexity of the modules and the limited time students have for their graduation, RoboCup students mostly do not understand the full complexity of

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 480 – 487, 2006.

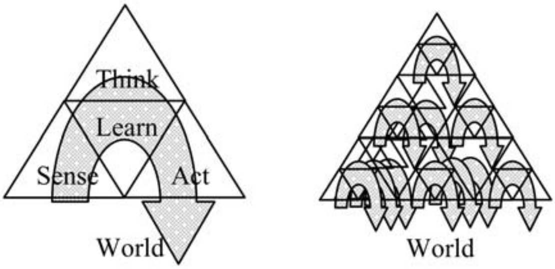


Fig. 1. Architecture based on: a) scientific disciplines b) required behaviors

the existing modules; so often their new software ruins or hinders good functionality elsewhere in the code, they start anew from scratch forgetting past lessons, or they build extra features from which the impact on the total system is not and cannot be overseen. In industrial embedded system projects, this is often not different! To cope with this, we have set-up a new software architecture for the Dutch AIBO Team (DAT) based on a hierarchy of modules in which each module is a separate relatively simple sense-think-act loop. See Fig. 1b. In the end we aim for a pluggable architecture in which it is relatively simple for students to understand the modules, to replace them with better ones or to add functionality by adding modules. The drawback of this approach is that we need students that need to understand the principles of image processing; mechanical engineering, control theory, AI *and* software engineering.

2 Behavior Based Vision

2.1 Advantages and Expected Performance Improvements

In this paper we show that a hierarchy of simple sense-think-act modules (Fig. 1b) performs better than an architecture based on monolithic modules (Fig1.a). To prove this, we changed the software of the goalkeeper of the Dutch AIBO team 2004 (DT2004) [10] and measured the differences with the NEW software for our goalie. As benefits we expect:

1. That each (sense-think-act) module is simpler and hence can be better understood and used to design new behaviors (copy-past-modify).
2. That effectively less and less complicated code is running in the new situation. We proved that this is true and that we made CPU cycles free.
3. That our goalkeeper performs better and more robust. The new software prevented more goals then the old software and, moreover, under a variation of lighting conditions. This was due to the capability of the goalie to localize itself better and more robust. And this was due to:
   a. Behavior specific self localization.
   b. Behavior specific image processing, or:
      i. Only search objects that you need to see for your task
      ii. Only search for objects at the place you expect them to see

c. Object specific image processing, or:
   i. process them using the shape knowledge over that object
   ii. process them using the color knowledge over that object.

2.2 Behavior Specific Processing Tools

To prove hypotheses 3 we used behavior-based vision to realize a goalie that can perform better and more robust. Fig. 2 is an implementation of Fig. 1b applied to the goalkeeper. For this role we can identify its basic behaviors. These are controlled by a meta-behavior that may invoke them. We will call this the governing behavior.

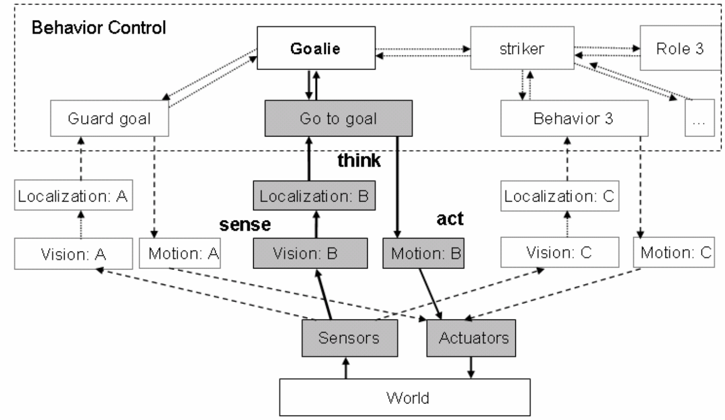


Fig. 2. Cut-out of the hierarchy of behaviors of a soccer AIBO, with emphasis on the goalkeeper role. Each behavior (e.g. go to goal) is an independently written sense-think-act loop.

Fig. 2 also shows that the cognition system of the robot is split into a vision system using grids and color tables [5] and a Monte Carlo self locator [6, 7] or particle filter. We have implemented our behavior-based vision on a Sony AIBO ERS-7 of the Dutch AIBO Team (DT2004) [4].We have implemented a behavior based architecture [8] by adding the following features:

- ***Behavior-specific self localization***
  When a robot is positioning (e.g. a goalie standing in the goal, or a field player walking around), the sensor input is qualitatively high and accurate localization is our aim; hence we use a fast update of the particles. When a robot is handling a ball, the sensor input is qualitatively low and the updating of the robot's pose is less urgent; hence we use a slow update of the particles.
- ***Behavior-specific image processing***
  Per behavior we only execute the image processing algorithms for detecting objects that are most likely to be seen, or that can be measured with high accuracy. A goalie guarding its goal, can most accurately and likely detect the lines of the goal area and its own corner flags. These objects are near and it is likely that they will not be occluded by opponents, which is the case for the opponent's goal and flags.

The drawback of this is that the robot can get stuck in a local loop, when making a wrong assumption about its position. A background process (at a lower pace) must detect this.

- ***Object-specific image processing***
  We use the type of objects, (goals, flags), color of objects (blue/yellow goal), and size of objects (far/near flag). For each type of object we define a *specific grid*, *color-table* and *specific size.* For each object we use a *specific color look-up table* (CLUT). CLUTs have to be calibrated [9]. Here we only calibrated the CLUT for the 2 or 3 colors necessary for segmentation. This greatly reduces the problem of overlapping colors. For each object we know its *preferred size* in order to be of interest. E.g., we might in some circumstances not be interested in a corner flag far away.

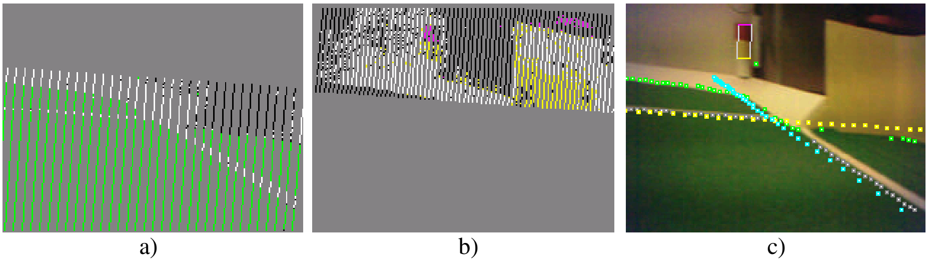


Fig. 3. Object-specific image processing: a) for line detection we scan the image below the horizon, using a green-white color table; b) for yellow flag detection we scan above the horizon using a yellow-white-pink color table; c) 2 lines and 1 flag detected in the image

2.3 The Basic Behaviors of a Goal Keeper and Its Behavior Based Processing

For the goalkeeper role we can identify 3 basic behaviors. They are:

- ***Goalie-return-to-goal.*** When the goalie is not in his goal area, he has to return to it. The goalie walks around scanning the horizon. When he has determined his own position on the field, the goalie tries to walk straight back to goal - avoiding obstacles - keeping an eye on his own goal. The localization relies on the detection of the own goal, detected line-points and detected border-points (no Hough-transform is used). The two own corner flags are also used for localization. All sensor input is used in a particle filter in which a detected own goal is used twice when updating the particles.
- ***Goalie- position.*** The goalie is in the centre of its goal when no ball is near. It sees the field-lines of the goal area often and at least one of the two nearest corner flags regularly. Hough transforms on detected line-points are used to calculate the distance and angle to the field lines. Detection of the own flags is based on a grid above the horizon, a 3-color LUT, and rejecting candidates that are too small. An orange/white/green color table was used for ball- and line detection. A particle filter was used that localized only on the detected lines and flags. A background process verifies the “in goal” assumption on the average number of detected lines and flags.

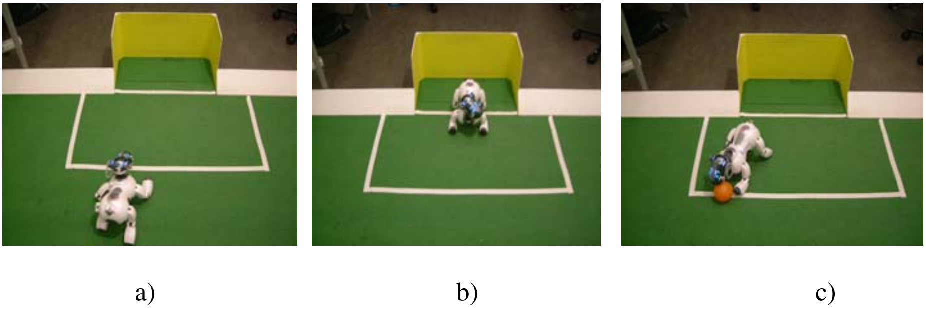


Fig. 4. Basic goalie behaviors: a) **Goalie-return-to goal,** b) **Goalie-position**, c) **Goalie-clear ball**. For each behavior a different vision system is used and a different particle filter setting.

- *Goalie-clear-ball.* When the ball comes near, the *goalie-clear-ball* behavior is switched on, meaning that the goalie is defending the line between ball and the center of the goal. The closer the ball comes to the goal area, the less time there is for the goalie to verify with the vision system where the center of the goal is. The goalie will gradually rely more on odometry than on vision. If the ball enters the goal area, the goalie will clear the ball. Walking to and controlling the ball, the quality of the vision input is very low. Although the same image processing is used as in *goalie-position-behavior,* the particles in the self-locater are updated much slower. Flags and lines detected at far off angles or distances are ignored.

3 Performance Measurements

3.1 General Setup of the Measurements

To prove our hypothesis 3, we have performed measurements on the behavior of our new goalie. Our test is twofold:

1. How fast can the new goalie find back his position in the middle of the goal on a crowded field in comparison with the old goalie
2. How many goals can the new goalie prevent on a crowded field within a certain time slot in comparison with the old goalie

As our improvements are due to three measures we have taken, we would like to know the contribution of each of the measures to the final result, i.e.;

a. Object-specific image processing
b. Behavior-specific image processing
c. Behavior-specific self localization.

3.2 Influence of Object-Specific Image Processing

We compared the DT2004 code with a general version of our NEW code. The latter does not (yet) use behavior specific image processing nor self-localization. However, it does use object specific grids and color tables. The tests consisted of searching for the goals, the flags, and all possible line- and border-points. The images were captured with the robot's camera, under a large variety of lighting conditions. For the DT2004 code, a single general CLUT was calibrated for all colors that are meaningful in the scene (blue, yellow, white, green, orange and pink) taking 3 hours. For the NEW image processing code we calibrated five 3-color CLUTs (white-green lines, blue-goal, blue-flag, yellow-goal, yellow-flag). This took 1 hour for all tables.

For all image sequences we counted the number of objects detected correctly (*N true*) and falsely (*N false*). We calculated the correctly accepted rate (CAR), being the number of objects that were correctly detected divided by the number of objects that were in principle visible. Table 1 shows the results on detecting flags and lines. It shows that the performance of the image processing largely increased. The correctly accepted rate (CAR) goes up from about 45 % to about 75%, while the number of false positives is reduced. The correctly accepted rate of the line detection even goes up to over 90%, also when a very limited amount of light is available.

Table 1. The influence of object-specific algorithms for goal, flag and line detection

Goals and flags	DT200 4			NEW			DT2004	NE W
	N true	CAR (%)	N false	N true	CAR (%)	N false	Lines (%)	Lines (%)
1 flood light	23	19	0	65	54	0	18	94
Tube light	54	45	9	83	83	1	58	103
4 flood lights	86	72	0	99	99	0	42	97
Tube +flood lights	41	34	1	110	92	0	24	91
Tube,flood+natural	39	33	0	82	68	0	42	91
Natural light	47	39	0	68	57	0		
Non calibration set	131	44	28	218	73	16		

3.3 Influence of Behavior Based Vision

Below we show the performance improvement due to behavior based switching of the vision system **and** the self localization algorithm. We used the following scenarios:

- *Localize in the penalty area.* The robot is put into the penalty area and has to return to a predefined spot as many times as possible within 2 minutes.
- *Return to goal.* The robot is manually put onto a predefined spot outside the penalty area and has to return to the return-spot as often as possible within 3 minutes.
- *Clear ball.* The robot starts in the return spot. The ball is put in the penalty area each time the robot returns. It has to clear the ball as often as possible in 2 minutes.
- *Clear ball with obstacles on the field.* As above but with many strange objects and robots placed in the playing field, to simulate a more natural playing environment.

To distinguish between the performance increase due to object-specific grids and color-tables, and the performance increase due to behavior-dependent image processing and self localization, we used 3 different configurations, resulting in Figs 5 and 6:

- *DT2004:* The old image processing code with the old general particle filter.
- *Striker:* The new object-specific image processing used in combination with the old general particle filter of which the settings are not altered during the test.
- *Goalie*: The new object-specific image processing used in combination with object-specific algorithms as well as with a particle filter of which the settings are altered during the test, depending on the behavior that is executed.

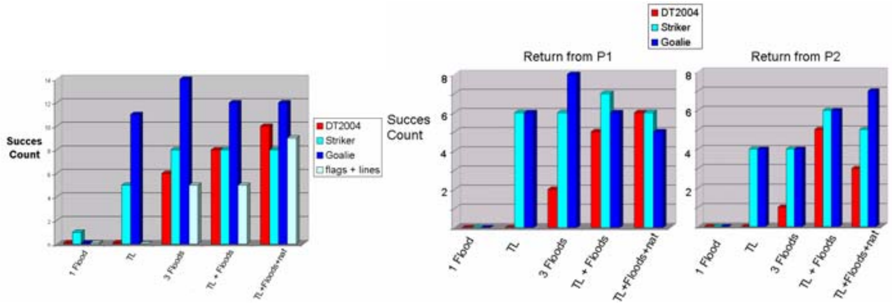


Fig. 5. Left. *Results for localization in the penalty area*. The number of times the robot can relocalize in the penalty area within 2 minutes. The old DT2004 vision system cannot localize when there is little light (TL). The performance of the object specific image processing is shown by the “flags and lines” bars. In contrast with the DT2004 code, the striker uses object specific image processing. The goalie uses both object specific image processing *and* behavior based vision processing *and* behavior based self localization. **Right.** Results of the return to goal test. The robot has to return to its own goal as many times as possible within 3 minutes. The striker vision systems works significantly better than the DT2004 vision system. There is not a very significant difference in overall performance between the striker (no behavior-dependence) and the goalie (behavior dependence).

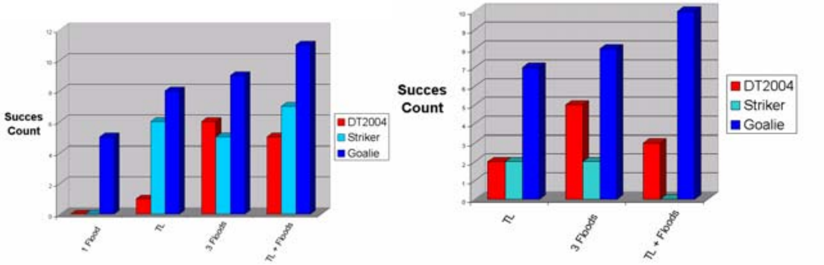


Fig. 6. Left. Results of the clear ball test. The robot has to clear the ball from the goal area as often as he can in 2 minutes. Both the striker and the goalie vision systems are more robust in a larger variety of lighting conditions than the DT2004 vision system (that uses a single color table). The goalie’s self locator, using detected lines and the yellow flags, works up to 50 % better than the striker self locator, which locates on all line-points, all flags and goals. **Right.** Results of the clear ball with obstacles on the field test. The goalie vision system, which uses location information to disregard blue flags/goals and only detects large yellow flags, is very robust when many unexpected obstacles are visible in or around the playing field.

4 Conclusions

- With object-specific grids and color-tables, the performance of the image processing (reliability) under variable lighting conditions has increased with 75-100%, while the color calibrating time was reduced to 30%.
- The impact of behavior-specific image processing and self localization can be seen from the localization test in the penalty area. The vision system of the goalie, with behavior based vision and self-localization, performs 50 % better on the same task as a striker robot with a vision system without behavior based vision and self localization. Note that both do use object specific image processing in this case.
- The impact of behavior-specific image processing on the reliability of the system can clearly be seen from the clear ball behavior test with obstacles on the field. With 1 flood light, where none of the robots can detect flags, the goalie can still clear a number of balls before being lost.
- Using object specific image processing reduced the CPU load with 50%. Using only specific algorithms as is done in behavior based vision and localization, then it is possible to decrease the load to about 10% of that in the old DT2004 code.
- Due to the new architecture, the code is more clean and understandable; hence better maintainable and extendable. The price is that one has to educate system engineers instead of sole image processing, software, AI, and mechanical experts.

References

1. R.A. Brooks. Intelligence without Representation. Artificial Intelligence, Vol.47, 1991, pp.139-159.
2. R.C. Arkin. Behavior based robotics, MIT press 19989, ISBN 0-262-01165-4
3. Parker, L. E. (1996). On the design of behavior-based multi-robot teams. Journal of Advanced Robotics, 10(6).
4. Stijn Oomes, P.P. Jonker, Mannes Poel, Arnoud Visser, and Marco Wiering, The Dutch AIBO Team 2004, Proc. Robocup 2004 Symposium (July 4-5, Lisboa, Portugal, Instituto Superior Tecnico, 2004, 1-5. see also http://aibo.cs.uu.nl
5. J. Bruce, T.Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00), volume 3, pages 2061-2066, 2000.
6. S.Thrun, D.Fox, W. Burgard, and F. Dellaert, Robust monte carlo localization for mobile robots. Journal of Artificial Intelligence, Vol. 128, nr 1-2, page 99-141, 2001, ISSN:0004-3702
7. T. Rofer and M. Jungel. Vision-based fast and reactive monte-carlo localization. In The IEEE International Conference on Robotics and Automation. In The IEEE International Conference on Robotics and Automation, pages 856-861, Taipei, Taiwan, 2003.
8. Scott Lenser, James Bruce, Manuela Veloso, A Modular Hierarchical Behavior-Based Architecture, in A. Birk, S. Coradeschi, and S. Tadokoro, editors, RoboCup-2001: The Fifth RoboCup Competitions and Conferences, Springer Verlag, Berlin, 2002.
9. Jüngel, M. (2005). Using Layered Color Precision for a Self-Calibrating Vision System. In: 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences, lecture Notes in Artificial Intelligence. Springer (to appear).

Coaching with Expert System Towards RoboCup Soccer Coach Simulation

Ramin Fathzadeh[1], Vahid Mokhtari[1], Morteza Mousakhani[1],
and Alireza Mohammad Shahri[2]

[1] Young Researchers Club, Mechatronics Research Laboratory
Department of Computer Engineering, Islamic Azad University
Qazvin Branch, Qazvin, Iran
{fathzadeh, mokhtari, mousakhani}@mrl.ir
http://www.mechatronics.ws,
http://www.yrc-ir.org
[2] Electrical Engineering Department
Iran University of Science and Technology, Tehran, Iran
shahri@iust.ac.ir

Abstract. In this paper we will describe our research in case of using Expert System as a decision-making system. We made our attempt to expose a base strategy from past log files and implement an online learning system which receives information from the environment. In developing the coach, the main research effort comprises two complementary parts: (a) Design a rule-based expert system in which its task is to analyze the game (b) Employing the decision-making trees for generating advice. Considering these two methods, coach learns to predict agent behavior and automatically generates advice to improve team's performance. This structure is tested previously in RoboCup Soccer Coach Simulation League. Using this approach, the MRLCoach2004 took first place in the competition held in 2004.

1 Introduction

In soccer coach simulation league, the coach is one autonomous agent providing advice to another autonomous agent about how to act. This advice is in format of standard coach language called CLang, [1], [2]. The goal of the coach agent is designing intelligent systems to control and observe multiple robots and provide the robots with the suitable methods to enhance their performance, [3]. In accordance with the coach specifications, we have recruited Expert System as a decision management system.

An expert system is a program which attempts to mimic human expertise by applying inference methods to a specific body of knowledge. In this system the information is consistently sensed from the environment, and using a forward-chaining method, system that compares data in the working memory against the conditions of the rules and determines which rules to fire, then the suitable advice is generated and sent to players, [4], [5].

In developing the coach, the main research is concentrated on designing an online learning system. For this purpose, before starting the match log analyzer analyzes the

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 488 – 495, 2006.

provided fixed opponent log files and gathers data consist ball position, players' positions, game score, play modes and generates an initial strategy based on these data against the opponent team, then puts it in the shared library for use in online coach. Online coach uses this strategy exposed by log analyzer to begin the match against fixed opponent. During the match coach calculates capabilities of the opponent and receives the information such as match time, result of the match, etc. from the field. At first, using expert system with predefined rules, coach models an appropriate strategy against the opponent team. Then by receiving statistical data (such as player position, the players' activity area, etc.) from the environment and using decision-making trees in message generation phase, coach generates suitable advice conforming the CLang structure, afterward sends the messages to the players.

The remainder of this paper is organized as follows: Section 2 comprises log analyzer and online coach. Log analyzer is to provide strategy for online coach. Online coach includes sense phase, expert system, message generation and act phase which its aim is to detect behavior of opponent team and providing a suitable strategy toward opponent team. Section 3 presents the results of our detailed experiments and finally discusses future work and conclusion.

2 Architecture of Coach Agent

This part presents the behavior of coach agent that involves log analyzer and online coach. In Figure 1 this structure is shown:

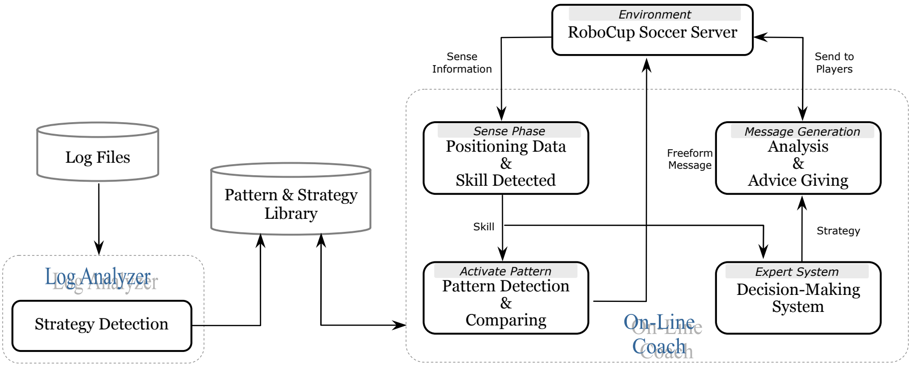


Fig. 1. Agent architecture and its components relation

2.1 Log Analyzer

Before starting the match, log analyzer verifies the provided opponent's log files and collects information such as ball position, players' positions, game score, play modes, sequence possession and number of shoots to target then with similar algorithms used in message generation phase in online coach, creates an initial strategy based on these data against the opponent and puts it in the shared library for use in online coach.

2.2 On-Line Coach

Online coach is divided to sense phase, expert system phase, message generation phase and act phase.

Sense Phase. By a set of features in which consist: our score, opponent score, goal difference, cycle number, distance from each player to the ball, coordinate of the ball and coordinate of each player calculates some skills such as detecting formation, pass graph, each player's activity area, ball position, sequence possession and number of shoots to target then the results of these skills are used to predict opponent ability and generating appropriate advice.

Expert System. In this phase coach makes its effort to analyze and investigate abilities of the opponent team and finally models the state of match in order to provide a strategy against the opponent team. This structure is shown in Figure 2.

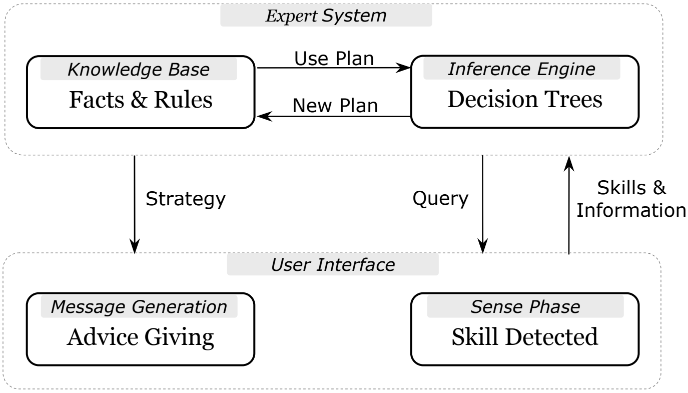


Fig. 2. Process of providing a strategy against the opponent team

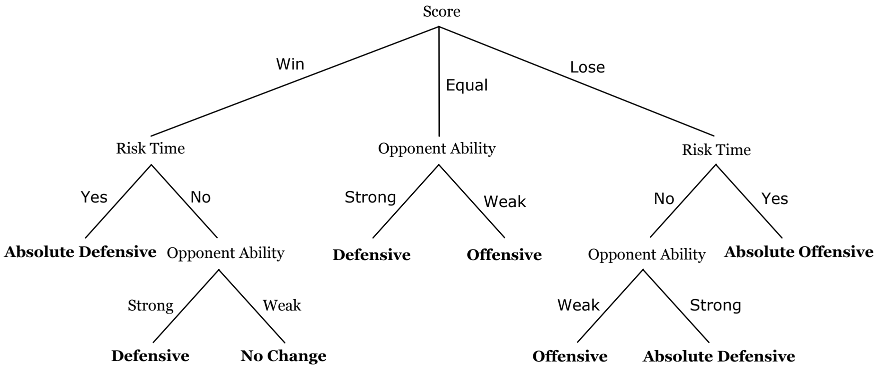
To estimate the abilities of the opponent team, information such as ratio of shoots to goal (which amounts to number of attacks of either team), successful passes, ball possession and compression ratio in triple areas of field is investigated. We define some weights for the parameters mentioned above to calculate opponent ability using the following formula:

$$\sum_{i=1}^{max}\left(\left({}^{\kappa_i}\!\big/_{|\kappa_i|}\right)\left[\beta_i + \left(|\kappa_i|\right)\times\alpha_i\right]\right) \tag{1}$$

max: number of parameters
β

$\boldsymbol{\beta_i}$: base point for each parameter
$\boldsymbol{\kappa_i}$:difference between the parameters of our team in comparance with the opponent
$\boldsymbol{\alpha_i}$: point scored for each $\kappa_{i.}$

For next step, using a decision-making tree designed on the basis of rule-based expert system architecture, the opponent team is modeled. In Figure 3 the decision-making tree for diagnosing opponent team's performance and giving the strategy against opponent team is presented:



```
Assertions:
IF (GoalDifference > 0) assert (Score Win)
IF (GoalDifference < 0) assert (Score Lose)
IF (GoalDifference = 0) assert (Score Equal)
IF (OpponentAbility > 0) assert (OpponentAbility Strong)
IF (OpponentAbility < 0) assert (OpponentAbility Weak)

Rules:
IF (Score = Equal AND OpponentAbility = Strong)
     OR
     (Score = Win AND RiskTime = No AND OpponentAbility = Strong)
THEN switch to defensive mode
IF (Score = Win AND RiskTime = Yes)
     OR
     (Score = Lose AND RiskTime = No AND OpponentAbility = Strong)
THEN switch to absolute defensive mode
IF (Score = Equal AND OpponentAbility = Weak)
     OR
     (Score = Lose AND RiskTime = No AND OpponentAbility = Weak)
THEN switch to offensive mode
IF (Score = Lose AND RiskTime = Yes)
THEN switch to absolute offensive mode
IF (Score = Win AND RiskTime = No AND OpponentAbility = Weak)
THEN don’t change strategy
```

Fig. 3. Decision tree learnt and its IF-THEN rules for giving strategy

Each path from root to leaf analyzes the state of match and opponent team. Finally a strategy is resulted in leaf. This strategy is sent to next phase, companying information and the needed skills for creating advice.

For example assuming that our team has scored more goals than the opponent team, two cases are considerable:

If game cycle is subsequent to Risk Time (This particular time is calculated based on experience and the environment and during this time there is opportunity for scoring goals for the team), in this case coach suggests absolute defensive strategy to keep the result of the game to our side regardless of abilities of opponent team.

If game cycle is prior to Risk Time, coach calculates abilities of opponent team. If the ability is estimated strong, to maintain the attained result, defensive mode is suggested otherwise if the opponent team is weak, the formation of team stays unchanged.

Message Generation. After learning strategy, coach attempts to generate proper advice. The advice is categorized into the sorts below:

Formation: The first step is creating a suitable formation. This formation involves two offensive and defensive modes in which each of them includes predefined templates suiting that state. Each offensive and defensive mode has absolute and ordinary modes as two sub modes.

Let the suggested strategy for team be absolute defensive mode, the arrangement is as follows (our recommended idea as a rule): 2 defenders more than the opponent offenders, 1 middle player of team more than the opponent middle players and other players are put as offenders.

$$\begin{aligned}&\textbf{\textit{No (our\_defenders) = 2 + No (opp\_offenders)}}\\&\textbf{\textit{No (our\_middles) = 1 + No (opp\_middles)}}\\&\textbf{\textit{No (our\_offenders) = 10 – [No (our\_defenders) + No (our\_middles)]}}\end{aligned} \tag{2}$$

Or in ordinary offensive mode, we put one player of our team for each player of the opponent in triple parts of the field.

And this process is applied to any other strategies.

For example assume that the formation of opponent team is detected as 433 and the suggested formation from expert system is absolute defensive, i.e. the opponent team has more strength than ours. In this case with attention to the predefined templates, formation of the team is calculated as follows:

No (our_defenders) = 2 + 3 = 5
No (our_middles) = 1 + 3 = 4
No (our_offenders) = 10 – [5 + 4] = 1
So the final formation would be 541.

Positioning: By learning formation of team, the positions of players are calculated. Positioning is divided into two modes (1) Marking the offenders of opponent team by our defenders (2) Positioning with ball for rest of the players. The following is the way each is calculated:

1) Mark: To generate the position, the activity area of offenders of the opponent team is calculated, then for marking (i.e. trapping) each offender one defender is summoned. To do this a circle with radius 'r' is assumed around the offender of opponent team, when each of defenders must mark this region. Regarding Figure 4

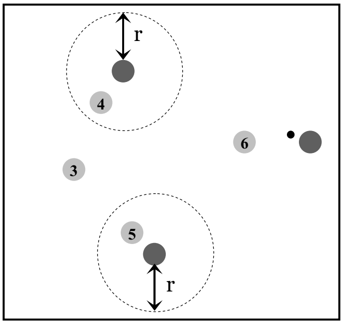


Fig. 4. Marking opponent

For example assume ball owner is one of the opponent players, related CLang rule which our player 5 marks offender is shown below:

```
(define (definerule Mark5 direc ((and (bowner opp {X})
       (bpos (rec (pt -52.5 -34) (pt 0 34))))
       (do our {5} (markl (arc (pt opp {$P}) 0 r 0 360)))) ))
```

2) Positioning with ball: In accordance with the calculated formation and activity area of the opponent players, our players' activity area is deduced. Now we segment the field into distinct areas as shown in Figure 5. We define the positions of our players in their activity area, according to the position of ball in the segments.

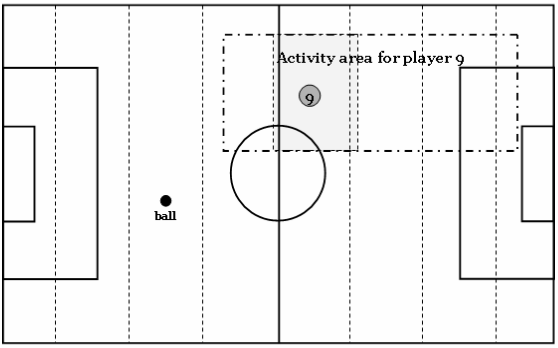


Fig. 5. Positioning player 9 with ball

For example the activity area of player number 9 is shown in Figure 5. With respect to the position of ball, the position of player would be the dark rectangle.

In CLang:

```
 (define (definerule Pos4_P9 direc ((and (bowner opp {X})
        (bpos (rec(pt -13 -34)(pt -25 34))))
        (do our {9} (pos (rec(pt 0 -34)(pt 15 34))))) ))
```

Pass Graph: Subsequent to learning the formation, the coach models pass graph of team. For each player we have a decision-making tree to create pass rule. The pass rule is calculated based on the positions of the other players. For player 'i' the pass rule is declared as follow, [6], [7]:

Pass (k):
Pass to teammate with uniform number k Є{X} – {i}
X is set of players whose x-coordinate is greater than x-coordinate of player 'i'.

Assume player number 5 is ball owner and players 6, 7 and 8 are the nearest players to it. This is shown in Figure 6.

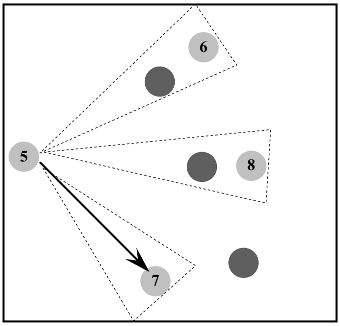


Fig. 6. If there are no opponents in the corresponding passing lane, pass to player k

Pass rule for player number 5 is generated as follows:
if (there aren't opponents in lane$_8$) pass (8)
else if (there aren't opponents in lane$_7$) pass (7)
else if (there aren't opponents in lane$_6$) pass (6)
else do *another action*

In CLang:

```
(define (definerule Pass5 direc ((and (bowner our {5})
      (not (ppos opp {X} 1 11 (tri (pt our 5)
      ((pt our 7) + (pt 0 5)) ((pt our 7) + (pt 0 -5))))))
      (do our {5} (pass {7}))) ))
```

“Another action” could be one of the actions shoot, clear, pass to region and dribble with respect to the position of player. For instance, in offense lane "shoot", in defensive lane "clear", and in the middle of field "pass to region" or "dribble" is chosen.

Act Phase. The act phase contains a queue in which the generated advice of previous phase is put in. Then they are sent to the players as commands.

3 Experimental Result

The MRL coach came in first place out of 12 entries in the 2004 RoboCup Coach competition. The competition consisted of three rounds. In each round, the coached team played three ten-minute games against a fixed opponent. Coaches were evaluated based on goal difference: the number of goals scored by the coachable team minus the number of goals scored by the opponent.

The fixed opponents were all teams that competed in the main simulator competition: Raic2004 in round 1, Hana in round 2, and Kshitij in round 3.

The score differences and rankings for the top four finishing teams are shown in Table 1. Our coach was ranked 1st place in all three rounds. MRL along with FC Portugal, Caspian and Sepanta progressed to the final round with MRL coming out on top.

Table 1. Total scores and rankings for the top four finishing teams in the 2004 RoboCup coach competition. The score consists of the number of goals scored by the coached team followed by the number scored by the fixed opponent.

Coach	*1st Round (RaiC2004)*		*2nd Round (Hana)*		*3rd Round (Kshitij)*	
MRL	0:11	1st	2:6	1st	1:12	1st
FC Portugal	1:13	2nd	2:9	3rd	0:11	2nd
Caspian	1:15	3rd	1:7	2nd	2:15	3rd
Sepanta	0:18	4th	0:13	4th	0:18	4th

4 Conclusion and Future Work

Our prospective effort is to develop a learning system which is able to generate advice in online mode with usage of algorithm for modeling the opponent team. We also plan and try to improve and develop learning algorithms and find out better solutions for generating advice.

References

1. Murray, J., Noda, I., Obst, O., Riley, P., Stiffens, T., Wang, Y., Yin, X., RoboCup Soccer Server User Manual: for Soccer Server 7.07 and later, 2002.H. Simpson, Dumb Robots, 3rd ed., Springfield: UOS Press, 2004, pp.6-9.
2. Patrick Riley and Manuela Veloso, An Overview of Coaching with Limitations. In Proceedings of the Second Autonomous Agents and Multi-Agent Systems Conference, pp. 1110–1111, 2003.
3. Patrick Riley and Manuela Veloso, 2004. Coaching Advice and Adaptation. In Daniel Polani,Andrea Bonarini, Brett Browning, and Kazuo Yoshida, editors, RoboCup-2003: The Sixth RoboCup Competitions and Conferences, Springer Verlag, Berlin.
4. Keith Darlington, the Essence of Expert Systems, England: PERNTICE HALL, 2000.
5. Peter Jackson, Introduction to Expert Systems, Second Edition, England: ADDISON WESLEY, 1990.
6. Gregory Kuhlmann, Peter Stone and Justin Lallinger, The Champion UT Austin Villa 2003 Simulator Online Coach Team. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, RoboCup-2003: Robot Soccer World Cup VII, Springer Verlag, Berlin, 2004.
7. Peter Stone, Layered Learning in Multiagent Systems: A Winning Approach to RoboticSoccer. Intelligent Robotics and Autonomous Agents. MIT Press, 2000.

Conceptual Representation for a Soccer Commentary Generator

Damian Stolarski

Adam Mickiewicz University
Faculty of Mathematics and Computer Science
ul. Umultowska 87, 61-614, Poznan
Poland
dst@amu.edu.pl

Abstract. The issue of this paper is bound with the project of building a commentary system for a soccer game. We surveyed an example of natural commentary in order to examine various sorts of comments used by a human reporter to cover the soccer match. The content and function of the messages was taken into account. This survey enabled us to create a conceptual representation for the RoboCup soccer game, which constitutes the input for our commentary generation system. We also give some details about how the representation is created from the raw quantitative data. The presented results may be interesting for human-machine communication researchers and software engineers involved in a similar project.

1 Introduction

Multi-agent system is an environment in which several autonomous individuals coexist. Such systems provide a rich problem domain and offer new challenges for artificial intelligence and robotics. Research effort focuses mainly on various interactions between the agents. This includes general topics like cooperation, competition and communication between the agents. Both new learning techniques and inventive knowledge modelling have to be developed for these problems. The issue presented in this paper is oriented more toward the aspect of the communication, though the information conveying between the agents is not directly studied. We study various forms of the commentary used for describing a situation and events in a multi-agent game. Our research is focused on the presentation techniques used by a human reporter to comment a soccer match. This research topic has been suggested in [7]. The practical purpose of this project is building a soccer commentary generator based on empirical information. The methodological steps leading from the system design to its evaluation are also our research objectives.

The number of works on the automatic generation of soccer commentary is limited. We are aware of three important projects from this area ([1], [2]) namely Soccer, Rocco, and Mike. Two of them are directly connected with the RoboCup environment. These projects are oriented toward the various topics like an explanation of complex events, description of spatial relations, and dynamic content selection. So

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 496 – 503, 2006.

far, the commentary forms have been studied only in [6]. The obtained results are interesting, but from our point of view they are too general. A more detailed survey of a human commentary is necessary for our purposes. The approach proposed in this project depends on empirical data. We suggest analysing an example of a commentary text[1], which is a video recording transcription of the real match. By examining the example we can learn how the reporter covers various match events and situations of the game. The analysed text had been divided on small coherent blocks of comments. These blocks were next surveyed and classified. More details of this part of the research can be found in [5]. The analysis enabled us to identify the concepts within the domain of soccer game and the various forms of comments appearing in the text. Then, we tried to determine a relation between the elements of the conceptual representation and the natural language expressions. In this way, we hope to construct an automatic generation system, which could potentially achieve more natural and cohesive output. Since the inspiration for our work comes from the RoboCup program, we started to develop our commentary system for this platform [4]. There are some practical advantages using RoboCup. RoboCup offers many ready-to-use logs of simulated games. The logs contain the low-level data. On such basis we can automatically create a conceptual representation of a game, which is in essence a high-level record of the match course. One disadvantage is that many essential commentary concepts like a team history, personal information, have a rather limited use in the RoboCup competition. That's why we neglected a number of the important background concepts. However, we believe that the RoboCup games may be successfully utilize to evaluate the accuracy of the generator.

Next in the other part of this paper we present the part of our project connected with building the conceptual representation of a soccer game. First, we show how the concepts have been selected on the basis of the example survey. Then we deal with some technical details of converting the Soccer Server raw output into conceptual elements of the match representation. This problem was an important part of our work. Finally, we present a short example of the match representation.

2 The Conceptual Representation of the Game

In order to generate a human-like commentary one have to identify the concepts contained in the comments of a human reporter. To the best of our knowledge the previous projects didn't include a serious investigation of the commentary conceptual structure. Although it is difficult to evaluate, it seems that their output text incorporated many unnatural elements. For example, the statements of the Mike generator seem to contain too much explicit statistical information. The analysis of the example commentaries let us select the most suitable conceptual elements and surely increases the conceptual coverage of the system. To determine what kind of information is important for a commentator we analysed the content of the particular messages. In order to illustrate the method of gaining these elements, we present some example comments and the attributes, which can be ascribed to them.

[1] The analysed test as well as the generated output is in polish language.

Comment	**Attributes (content)**
Ronaldo	actor-of-action
Immediately on the penalty area	time-of-reaction, region-of-pitch
But only corner kick	result-of-action,
Good interception by the French team	type-of-action, team-of-action, result-of-action
There is Leonardo on the penalty area	actor-of-action,region-of-pitch
Dangerous Cafu dribble	type-of-action, chance-for-goal, actor-of-action
Long pass on the left side	type-of-action, qualitative-distance, region-of-pitch
Two defenders staying next to him	number-of-opponents-staying-close
This is his first dribble	type-of-action, order-number-of-action

To compare the distribution and establish the importance of different attributes we selected approximately 280 comments from the first 25 minutes of the commentary and examined their content. The distribution of the attributes in the comments is as follows:

Action information	**Distribution in %**
Comments containing actor of action	76 %
Comments containing type or result of action	31 %
Comments containing team of action	12 %
Spatial and temporal information	
Comments containing region of the pitch or direction	5 %
Comments containing time of player's reaction expressions	3 %
Comments containing dynamics or speed of action expressions	3 %
Comments containing distance expressions	1 %
Comments containing statistical information	**3 %**

Taking into account the result of our survey we have prepared a set of elements representing a course of the RoboCup soccer match. In the current version of our system the most fundamental elements of the conceptual representation have been implemented. First of all, the attributes representing the actions and main game events have been included. We also added the most important spatial concepts and some team performance measures. We focused on these elements, which seemed to be the most appropriate for the RoboCup domain. The current set of the attributes encompasses:

1) Attributes representing the individual action of player and match events

type-of-action	: Pass, Shot, KeepBall, Dribble, Defence, KickOut, etc.
result-of-action	: Receive, Loss, Out, Goal, Corner, Interception, etc.

actor-of-action	: identifier of the player
type-of-event	: Beginning-of-match, End-of-match, Break, etc.
state-of-action	: Active, Finished

2) Attributes representing the action of team

state-of-action	:Beginning, Course, Break, Continuation, End
type-of-action	: Counter-Attack, Positional-Attack, etc.
team-of-action	: name of the team

3) Attributes of the individual and team's action

region-of-pitch	: Middle, Rival-penalty-area, Left-half, etc.
distance	: appropriate qualitative value
direction	: Right, Left, Forward, Back
time-of-reaction	: Immediately, Normal, Slow
speed-of-action	: appropriate qualitative value
dynamics-of-action	: Acceleration, Constant, Slow-Down
order-number-of-action	: a counter of team actions
chance-for-goal	: None, Little, High, Very-High

4) Attributes describing player's situation

number-of-opponents-staying-close	: a number
number-of-team-mates-staying-close	: a number
player-position	: Free, Marked
distance-to-enemy-goal	: a value from {Far, Near, Very-Near}

5) Attributes describing team's performance

initiative	: a degree of team's prevail in the match
ball-possession	: a percent of ball possession in the match
dangerous-situations	: a counter of team's dangerous situations

6) Attributes representing the match situation

time-of-game	: current game time
result-of-game	: current game score

We hope that after providing some background commentary concepts this representtation would be rich enough for generating an interesting soccer commentary. The distributions of the particular attributes may be applied to defining the generation rules, so as the proportions were comparable with a human text. On the other hand the collected data can be used to evaluate quality of the generated output. After having selected the main elements of the conceptual representation, we had to cope with the problem of extracting these concepts from the Soccer Server low-level data.

3 Building the Conceptual Representation

In this section the way of converting the Soccer Server data into the conceptual representation is outlined. The raw input received directly from the Soccer Server or loaded from a game log is consisted of the records containing:

- current location, state, and orientation of the players,
- current location of the ball,
- current score, time, and mode of the game.

This is the low-level quantitative layer, because the data provides us with no conceptual information about the events and the situation of the game. Such representation can not be directly used for a commentary generating. So, this is the starting point for building a higher conceptual layer of a match representation. The raw data is processed in order to extract the necessary conceptual information. After the conversion, a file which contains the conceptual representation of the match is returned as a result. Such file can be used as an input for the generator. The conversion of the raw data can be divided into a few successive steps executed for each time frame. The main routine of the conversion is as follows:

```
t = current_time_of_game();
calculate_values_of_spatio_temporal_primitives(t);
new_mode_event_search(t);  //a new mode event search
new_state_event_search(t); //a new state event search
if (lastKickEvent->type-of-action == KeepBall) ||
   (lastKickEvent >type-of-action == Dibble)
{
  check_player_properties(currentKickEvent->actor-of-action,t);
  report_player_properties(currentKickEvent->actor-of action);
}
update_team_action(LeftTeam);
update_team_action(RightTeam);
```

1. First, we calculate the basic geometrical primitives of the representation. They represent basic properties of the objects and different spatial relations which occur between them. This includes:
   - Velocity and acceleration of the movable objects.
   - Finding a player who is nearest the ball.
   - Tracking the ball movement (velocity change and direction change).
   - Region of the pitch where player is located.

2. Then, we check for new events. The events can be divided into:
   1) Signalised by the change in the game mode (*newModeEvent*).
   2) Signalised by the change in the player's state (*newKickEvent*).

Every state event is in essence a player's action represented by a separate object of *SingleKickEvent* type. The system stores the information about every three successive events: *newKickEvent*, *currentKickEvent*, and *previousKickEvent*, because they can be utilized for recognizing the type (i.e. *pass* or *dribble*), and the result (i.e. *goal*) of the action. If a new event is detected then an appropriate action is being taken depending on the event's sort. For example when a new kick action is detected then the following code is executed:

```
newKickEvent = new SingleKickEvent(time, actor-of-action);
      //a new object is being created
 if (currentKickEvent != NULL)
{
  currentKickEvent->setFinishEvent(newKickEvent);
      //the new kick event finishes the current one
  currentKickEvent->recognize();
      //the type of current event is being determined
  currentKickEvent->setActionResult();
      //the result of the current event is being determined
  currentKickEvent->report();
      //this call reports the type and the result
};
```

If a new kick action has been detected then a type of the current one is determined. We experimented with using the recognition automata [2] to model the basic actions like *pass, dribble, shot*, and etc. An interesting idea for recognizing the complex actions in a dynamic multi-agent environment has been outlined in [3], but it still remains purely theoretical because of the computational complexity. Currently we prefer to use a simpler technique that is based on matching the patterns of successive kicks and the game events. The recognition of the type and the result of an event is done by special rules which use spatio-temporal primitives and a set of helpful functions like: *team*(*player*) which returns the identifier of player's team, or *is_pass_possible*(*start_time, actor-of-action, team-mate*) which checks whether the pass was possible. For example, one of the recognition rules says that:

```
if (team(actor-of-action)==team(newKickEvent->actor-of-action))
{
  type-of-action    = Pass;
  result-of-action  = Receive;
}
else
{
  if (is_pass_possible(start_time, actor-of-action, team-mate))
     type-of-action = Pass;
  else
     type-of-action = KickBall;
  result-of-action = Loss;
};
```

The above rule just checks whether the ball was transferred between the two team-mates. If yes, the pass is recognized and its result is set to *Receive*. If no, then the pass is recognized only if it was intended by the actor. The result is set to *Loss*. The rules are triggered in a proper order. For each kind of action the appropriate attributes are calculated. For example, length is calculated for a pass, and the numeric value is transformed into a qualitative value: *short, normal, long*. Other attributes are calculated in a similar way. For a player who is in the ball possession the additional properties of his position are determined and reported. These properties include: *player-position,* the number of team-mates and opponents staying close, and *distance-to-enemy-goal.*

3. At the end, we update the data concerning the action of both teams. It encompasses setting the following attributes: *region-of-action*, *dynamics-of-action*, s*tate-of-action*, and *speed-of-action.* Also the *team-of-action* (the attacking team) is updated if necessary. Finally, the performance data are updated.

4 Example

Below, we give a short example of the RoboCup match representation created automatically from the Soccer Server output data. A similar representation of the real match fragments can be created manually, so that we could check how the generator is commenting the real match.

<*time*=1; *type-of-event* = beginning-of-match>
<*time*=6; *type-of-event* = beginning-of-game>
<*time*=6; *team-of-action* = ATTCMUnited2000; *region-of-pitch* = Middle;
state-of-action = Beginning; *number-of-action* = 1>
<*time*=6; *type-of-action* = KeepBall; *region-of-pitch* = Middle;
actor-of-action =Yellow8>
<*time*=7; *player-position* = Free>
<*time*=7; *distance-to-enemy-goal* = Far>
<time=7; *team-of-action* = ATTCMUnited2000; *region-of-pitch* = Middle;
state-of-action = Course; *number-of-action* = 1>
<*time*=8; *type-of-action* = Pass; *direction* = Left; *region-of-pitch* = Middle;
actor-of-action = Yellow8>
<*time*=11; *result-of-action* = Receive; *region-of-pitch* = Middle;
partner-of-action = Yellow9>
<*time*=11; *type-of-action* = KeepBall; *region-of-pitch* = Middle;
actor-of-action = Yellow9>
<*time*=12; *player-position* = Free>
<*time*=12; *distance-to-enemy-goal* = Far>
<*time*=12; *type-of-action* = Pass; *direction* = Left; *region-of-pitch* = Middle;
actor-of-action = Yellow9>
<*time*=17; *result-of-action* = Receive; *region-of-pitch* = Right-Half;
partner-of-action = Yellow5>
<*time*=17; *type-of-action* = KeepBall; *region-of-pitch* = Right-Half; *actor*=Yellow5>
<*time*=18; *player-position* = Marked; *number-of-opponents-staying-close* =1>
<*time*=18; *type-of-action*=Pass; *direction* = Forward; *region-of-pitch* = Left-Half;
actor-of-action = Yellow5>
<*time*=18; *distance-to-enemy-goal* = Far>
<*time*=38; *result-of-action* = Loss>

5 Conclusion

In this paper the conceptual structure of information for a dynamic multi-agent game was presented. The representation is designed for a soccer commentary generation system. The analysis of a human soccer commentary lead us to the representation which contains the attributes representing: the actions, the events and their spatial, temporal, and statistical properties. We also introduced two levels for representing an action of the game. The one for representing the actions of individual players and other level for a team action. Next, we outlined the way the representation is created from a low-level quantitative data. The topic beyond the scope of this paper is a commentary generation. In future research, we want to add more performance measures and the attributes representing the knowledge used in background commentaries. We also want to extract more statistical data from the example and apply them to the generation rules. The evaluation procedure for a soccer commentary generator is another open question. We are planning to test the accuracy of the generator and similarity between computer and human output.

References

1. Andre, E., Binsted, K.., Tanaka-Ishii, K., Luke, S., Herzog, G., & Rist, T.: Three robocup simulation league commentary systems. AI Magazine 21 (1), 2000, 57-66
2. Andre, E., Herzog, G., Rist, T.: On the simultaneous interpretation of real world image sequences and their natural language description: The system soccer. Proceedings of the 8th ECAI, pages 449-454, Munich, 1988
3. Lison, M.: Deformation Parameters in Dynamic Event Recognition. RoMoCo '02, Poland, 2002
4. Stolarski, D.: Reprezentacja konceptualna wieloagentowej gry dynamicznej w srodowisku irtualnym RoboCup. XIII National Seminar on Artificial Intelligence, Virtual Organization, AI-18’2003, Nr22, pp 51-59
5. Stolarski, D.: An empirical approach toward building a soccer commentary generator. Proceedings of 2nd Language & Technology Conference, Poznan, 2005
6. Tanaka-Ishii, K., Frank, I.: Multi-Agent explanation strategies in real-time domains. Proceedings of ACL2000, the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, 2000
7. Tanaka-Ishii, K., Frank, I., Noda, I., Matsubara, H., Hasida, K., Nakashima, H.: Automatic Soccer Commentary and RoboCup. In M. Asada (Ed.): Proc. of the second RoboCup Workshop, RoboCup-98: Robot Soccer world Cup II

Distributed Sensor Fusion for Object Tracking

Alankar Karol and Mary-Anne Williams

Faculty of Information Technology, University of Technology, Sydney, NSW 2007, Australia

Abstract. In a dynamic situation like robot soccer any individual player can only observe a limited portion of their environment at any given time. As such to develop strategies based upon planning and cooperation between different players it is imperative that they be able to share information which may or may not be in any individual player's field of vision. In this paper we propose a method for multi-agent cooperation for perception based upon the Extended Kalman Filter (EKF) which enables players to track objects absent from their field of vision and also to improve the accuracy of position and velocity estimates of objects in their field of vision.

1 Introduction

Robot Soccer is a relatively new research initiative and in terms of its development it is in its infancy. One of the current challenges for robots playing in the RoboCup Four-Legged League is to implement a game based upon cooperation and planning. Planning e.g. behaviour or a strategy based upon game play [2] can be facilitated by the efficient sharing of information among the team players. Players can share information regarding their own pose, estimate of the pose of the opponents and an estimate of the ball position and velocity.

Since each player (generally) knows its own pose with a high degree of accuracy sharing this information is fairly straightforward. However sharing information about the ball and the opponents is not that trivial. In this paper we ignore the problem of tracking opponents since we wish to avoid the problem of data association. We propose a solution for ball tracking by fusing information from multiple robots using an Extended Kalman Filter (EKF).

Kalman filters have been successfully used in robotics to track objects and achieve robot localisation with a high degree of accuracy. For robot localisation, Kalman filters effectively fuse information from position estimating sensors like sonars and vision with odometry [5] [7] [8] [4]. The observations are matched to a map of the local environment and then merged to update the robot pose. In the studies mentioned above, Kalman filters were successfully employed to fuse together information from different sensors located on the same robot. We undertake the task of fusing distributed simultaneous sensor information from different robots.

The problem of ball tracking by fusing multiple simultaneous observations of the same object from distributed vantage points has been tackled in the past [6] [9]. Stroupe *et al.* [6] used the method of Smith and Cheeseman [5] to combine two dimensional Gaussian distributions by a single matrix operation. Due to the associativity and the symmetry of the operation any number of distributions can be combined in any order

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 504–511, 2006.

to fuse the information from different robots leading to a highly efficient and reactive method. However the parameters of the observation in their case do not correspond to the canonical form of the Gaussian as assumed by the method of Smith and Cheeseman [5]. To overcome this obstacle they have to rotate their observations and then re-rotate them in the end to extract the position and uncertainty of the tracked object. Moreover, the method they describe is used only for locating the position of the ball when the robots are perfectly localised (i. e. ,the different robots know their pose with certainty) and standing still which is achieved artificially by placing the robots in specific locations. Also, since they only locate the position of the ball they are unable to extract information regarding the speed and the direction of the ball's motion.

The solution to the general problem of fusing the multiple simultaneous observations to accurately determine the position and the velocity of the ball is described by Weigel *et al.* [9]. They determined the ball position by a probabilistic integration of all ball measurements coming from the different players using a combination of Kalman Filters and Markov Localisation. Since Grid-Based Markov Localisation is computationally expensive they used only a two dimensional grid which does not allow them to store the velocity of the ball and as such they could not determine the position of the ball accurately when the ball was in motion. However their algorithm does allow for integrating the ball in a global sensor integrator placed off-board.

In the robocup legged-league off-board processing is not allowed and any algorithm for fusing the multiple simultaneous observations wold have to be constrained by the computational power of the AIBO robots. We propose a method for ball tracking to accurately determine the ball position and velocity when both the robots and the ball is in motion. We use an approach based upon Extended Kalman filters which does not make any assumptions of linearity. The algorithm is highly efficient and does not place undue load on an AIBO Robot and can be used either on an ERS-7 or an ERS-210 without degrading performance. The algorithm was successfully used in the RoboCup 2004 championships held in Lisbon where UTSUnleashed! exploited the power of information sharing in the Open Challenge where we demonstrated active passing between players. UTSUnleashed! also used the above algorithm in games to successfully create game plays based upon cooperation which led us to second position in the soccer competition in our second year of competition.

The remainder of the paper is set out as follows. In the next section we give a brief description of the theory of Extended Kalman filter. In Section 3 we present the state model and the algorithm for fusing multiple simultaneous ball observations by different robots. In section 4 we present the results of some analytical experiments and in section 5 we conclude with a brief discussion.

2 Extended Kalman Filter

The problem of ball tracking and robot localisation requires an estimate every time a new measurement is received, which calls for a solution comprising of a recursive filter. This means that the data can be processed sequentially rather than as a batch, which greatly enhances the execution of the filter algorithm since it is not necessary to store the complete data set nor is it necessary to reprocess the existing data when new measurements become available. Such a filter consists of essentially two stages: prediction and update. The prediction stage uses the system model to predict the state probability density function from one measurement time to the next. The state is usually subject to unknown disturbances which are modelled as random noise hence the prediction stage generally translates, deforms and spreads the state probability density function. The update operation uses the latest sensor measurements to modify the predicted probability density function.

Formally we want to consider the evolution of the state sequence $\{x_k, k \in N\}$ given by $x_k = f_k(x_{k-1}, v_{k-1})$ where f_k maybe a nonlinear function of the state x_{k-1} and v_{k-1} is a process noise sequence. The objective of tracking is to recursively estimate x_k from measurements $z_k = h_k(x_k, w_k)$ where h_k maybe a nonlinear function and n_k is a measurement noise sequence. In particular we seek filtered estimates of x_k based on the set of all available measurements $z_{1:k} = \{z_i, i = 1, \cdots, k\}$ up to time k.

From a Bayesian perspective, the tracking problem is to recursively calculate some degree of belief in the state x_k at time k, given the data $z_{1:k}$ up to time k. Thus it is required to construct the probability distribution function $p(x_k|z_{1:k})$. It is assumed that the initial pdf $p(x_0|z_0) \equiv p(x_0)$ is known a priori. Then in principle, the pdf $p(x_k|z_{1:k})$ may be obtained recursively in two stages: prediction and update.

Suppose that the required pdf $p(x_{k-1}|z_{1:k-1})$ at time $(k-1)$ is available. The prediction stage involves using the system model to obtain the prior pdf of the state at time k:$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1}$. At time step k, a measurement z_k becomes available, and this may be used to update the prior pdf via Baye's rule:$p(x_k|z_{1:k}) = \dfrac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})}$. In the update stage, the measurement z_k is used to modify the prior density to obtain the required posterior density of the current state.

The Extended Kalman filter assumes that the posterior density at every time step is Gaussian and hence, parametrised by a mean and a covariance. If $p(x_{k-1}|z_{1:k-1})$ is Gaussian then it has been shown [1] that $p(x_k|z_{1:k})$ is also Gaussian. It is then possible to write the above by means of matrix operations

$$\hat{F}_k = \left.\frac{df_k(x)}{dx}\right|_{x=m_{k-1|k-1}} \tag{1}$$

$$\hat{H}_k = \left.\frac{dh_k(x)}{dx}\right|_{x=m_{k|k-1}} \tag{2}$$

If we define $\mathcal{N}(x, ; m, P)$ as a Gaussian density with argument x, mean m and covariance P, the Extended Kalman filter algorithm can be viewed as the following recursive relationship:

$$\begin{aligned} p(x_{k-1}|z_{1:k-1}) &\approx \mathcal{N}(x_{k-1}; m_{k-1|k-1}, P_{k-1|k-1}) \\ p(x_k|z_{1:k-1}) &\approx \mathcal{N}(x_k; m_{k|k-1}, P_{k|k-1}) \\ p(x_k|z_{1:k}) &\approx \mathcal{N}(x_k; m_{k|k}, P_{k|k}) \end{aligned}$$

where

$$m_{k|k-1} = f_k(m_{k-1|k-1}) \tag{3}$$

$$P_{k|k-1} = Q_{k-1} + \hat{F}_k P_{k-1|k-1} \hat{F}_k^T \tag{4}$$

$$m_{k|k} = m_{k|k-1} + K_k(z_k - h_k(m_{k|k-1})) \tag{5}$$

$$P_{k|k} = P_{k|k-1} - K_k \hat{H}_k P_{k|k-1}. \tag{6}$$

In the above $(z_k - h_k(m_{k|k-1}))$ is termed as innovation and is the difference between the expected and the actual measurement. K_k is defined as the Kalman gain and is given by $K_k = P_{k|k-1} \hat{H}_k^T S_k^{-1}$. where S_k is the covariance of the innovation term and is given by:$S_k = \hat{H}_k P_{k|k-1} \hat{H}_k^T + R_k$.

3 Ball Tracking

In the case of robot localisation it is fairly simple to have a state model since the motion model which consists of the commands given to the actuators (walk forward, strafe left, turn clockwise) captures the robot's kinematics. The update model consists of the observations (distance, heading and elevation) made by the robot's camera to the landmarks, if seen in the frame. If the robot sees more than one landmark in any given vision frame then the estimate of the robot's pose is highly accurate since the uncertainties introduced by the observation of one landmark are compensated for by the observations to the other landmarks. Thus the more landmarks that a robot sees in any given frame the more accurate the estimate of the pose.

We use a similar logic for the problem of ball tracking by reversing the problem of robot localisation into a problem of ball localisation. Let us assume for the sake of explanation that the ball has a camera on it and the robots are the beacons. The observations made by the robots to the ball can then be viewed as observations made by the imaginary camera on the ball. One can then imagine that the ball uses the EKF described above to localise itself off the robots. The more robots that a ball views the better localised it will be since the uncertainties are compensated for by multiple observations.

The state vector of the ball is given by $(x, y, v_x, v_y)^T$ where x and y are the coordinates of the ball in a global reference frame and v_x and v_y are the speeds of the ball along the x and the y axis respectively. The heading of the ball with respect to the x-axis can then be easily calculated by $\phi_b = \tan^{-1}\left(\frac{v_y}{v_x}\right)$ and the speed of the ball is given by $v_b = \sqrt{v_x^2 + v_y^2}$.

Since it is impossible to capture the exact kinematics of the ball we model the state model by using the standard equations of kinematics.

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_f = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_i + \begin{bmatrix} (v_x + \frac{a\delta t}{2})\delta t \\ (v_y + \frac{a\delta t}{2})\delta t \\ a\delta t \\ a\delta t \end{bmatrix} \tag{7}$$

In the above system a is the magnitude of the retardation of the ball which is determined by experiments and δt is a time interval.

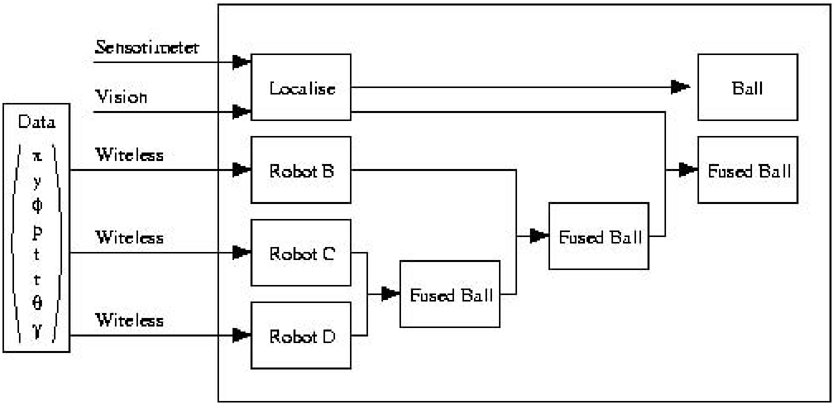


Fig. 1. Diagram representing the algorithm used to fuse information from different robots using the method described above

The measurement model comprises of the distance to the ball, the heading to the ball and the elevation to the ball with respect to the camera center of the robot. Formally:

$$\text{Distance} = \sqrt{(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2}. \tag{8}$$

$$\text{Bearing} = \tan^{-1}\left(\frac{D_v}{D_u}\right). \tag{9}$$

$$\text{Elevation} = \tan^{-1}\left(\frac{D_w}{D_u}\right). \tag{10}$$

Where, D_u, D_v, D_w are the unit vectors from the camera. The coordinates of the robot's camera center (x_c, y_c, z_c) can easily be calculated if the pose of the robot, the neck tilt and the head tilt and pan angles are known.

It is well known that the observations of landmarks leads to a reduction in the uncertainty of the pose when using an EKF [1][3]. Thus if a robot observes a landmark in a vision frame then its uncertainty is reduced. If the robot has not observed a landmark then we increase it's uncertainty by a constant factor determined by experiments. Thus the measure of the uncertainty gives an indication of how reliable the observations of a particular robot are. If the uncertainty of the robot is within some error bound then we consider it's observations to be reliable and the robots transmits it's observations to the team members. If on the other hand the uncertainty is large then it's observations would corrupt the EKF of the team members and hence the robot does not send its information. In this case the robot calculates the position of the ball relative to it's frame of reference. In this situation the robot is still able to react to the ball which is important in a game of robot-soccer.

The robots which satisfy the uncertainty threshold described above then transmit to their team mates information regarding their own pose, their neck tilt and pan angles and the observation of the ball (distance, heading, elevation). We send the neck tilt and pan angles as opposed to the camera coordinates and the unit vectors to reduce the overall transmission on the wireless network.

The receiving robot calculates the camera locations and the unit vectors for each robot with the aid of the above information and then updates the ball position and velocity as described in section (2). As a result of this algorithm the pose of the **fused ball** is unique and is common to all the robots. This pose might be different from the pose which is calculated independently by the robot based on its own observations.

Different issues that arise from such a procedure are tackled as such:

- If the robot can see the ball then behaviour is adopted based upon its own observation of the ball. This is necessary to accommodate for the fact that the data from other robots lags by at least one frame. Moreover if the fused ball is quite distinct from the robot's own perception of the ball then th robot is not forced to make wrong decisions or be frozen as a result of the disparity. This is important in a highly reactive game like robot soccer.
- If the uncertainty in the determination of the robot's own pose is high then it does not send the ball information to the other robots. This normally tends to happen when the robot is chasing a ball and is unable to see a landmark. In this case if the uncertainty in the fused ball is small (compared to some threshold) then it is possible for the lost robot to use the ball as a landmark and update its own pose. Complete localisation takes a few cycles and the robot will inadvertently see a landmark during that time.
- Since the ball identification and measurements are vision based (as opposed to a sonar) it is quite possible that erroneous information may enter the data set. For example a robot might see a false ball or get a wrong distance reading due to reflection. In this case we employ a 2 sigma gate procedure which does not allow any readings which deviate by 2 standard deviations to be integrated into the set.
- If no observation is made for 2 seconds (50 framesin an ERS210 and 60 frames in an ERS7) then the track is deleted and the ball position is reinitialised from observation.

4 Results

Experiments were performed using Sony AIBO ERS-210 robots. Several experiments trying to emulate the dynamic nature of a robot soccer game were carried out. A brief description of a prototypical example is given below and the results reported.

Robots were placed on the field and allowed to move. A ball was rolled along a particular trajectory and the estimates of the ball position and velocity from the robot were recorded. Figure (2) shows the configuration of the field and the position of the robots initially. Robot A was looking outfield and was not allowed to move. However the other robots were allowed to move. The robots were given a few seconds to localise themselves. A ball was then placed on the penalty box of the blue goal and made to roll diagonally to the penalty box of the yellow goal. Robots B, C and D could see the ball at all times with Robot D and C coming quite close to the ball although none of the robots managed to touch the ball. As robot A could not see the ball at any instant, information from the three robots was fused together according to the process described above and the trajectory of the ball according to Robot A's world model was plotted.

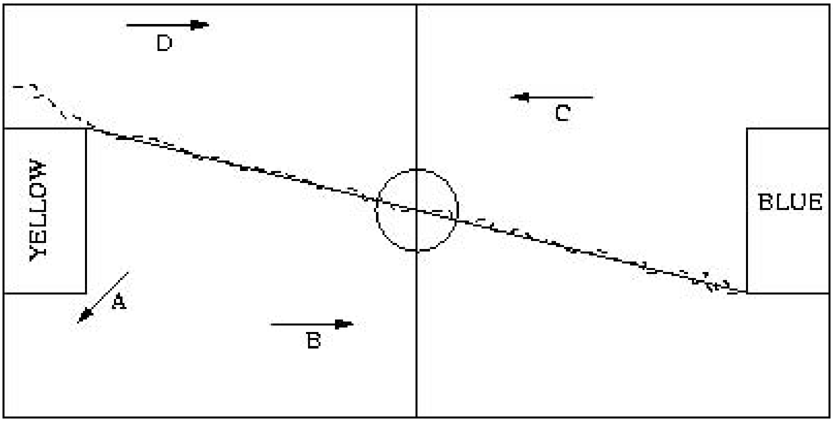


Fig. 2. Trajectory of the ball (solid line) and the trajectory as observed by Robot A (dashed line). Robot A was standing in place while robots B, C and D were allowed to move.

The result is shown in figure (2). As can bee seen from the figure the trajectory of the ball as evaluated by robot A is quite accurate.

Several trials of similar experiments were performed and on an average the mean deviation from the actual path was 5 cm with a standard deviation of 1 cm. Trials performed where all the robots could not see the ball simultaneously gave similar results reflecting the robustness of the algorithm.

5 Discussion

In this paper we present a method of integrating simultaneous observations of a single unique object from different robots using an extended Kalman filter. The computational ease and flexibility of the approach makes it an ideal candidate for object tracking in complex dynamic domains. The approach was tested in the domain of the Robocup where the task is made particularly difficult due to the dynamic and uncertain nature of the domain. The ability of moving robots to fuse information about moving targets enabled UTSUnleashed! to build a highly competitive team at RoboCup 2004.

References

1. Ho, Y.C. and Lee, R.C.K.: A Bayesian Approach to Problems in Stochastic Estimation and Control. IEEE Trans. Automat. Contr., vol. AC-9, 1964. pp. 333–339.
2. Karol,A., Nebel, B., Stanton, C., Williams, M-A.: Case Based Game Play in the Robocup Four-Legged League Part I The theoretical Model. Robocup Symposium 2003.
3. Maybeck, Peter S.: Stochastic Models, Estimation and Control: Volume I. Academic Press, 1979.
4. Sasiadek, J.Z. and Hartana, P.: Sensor data fusion using kalman filter. Proceedings of the third International Conference on Information Fusion, Vol 2, 2000. 19–25

5. Smith, R.C. and Cheeseman, P.:On the Representation and Estimation of Spatial Uncertainty. International Journal of Robotics Research, Vol 5, no. 4, Winter 1986. 56–68.
6. Stroupe, A., Martin, M.C. and Balch, T.: Distributed Sensor Fusion for Object Position Estimation by Multi-Robot Systems. Proceedings of the IEEE International Conference on Robotics and Automation- May, 2001. IEEE, 2001.
7. Tang, C.Y., Hung, Y.P., Shih, S.W. and Chen, Z.: A Feature Based tracker for multiple object tracking. Proceedings of the National Science Council, republic of China, Vol 23(1) Part A, 1999. 151–168.
8. Wang, H., Chua, C.S. and Sim, C.T.: Real Time Object Tracking from Corners. Robotica, Vol 16(1), 1998. 109-116.
9. Weigel, T., Gutmann, J.-S., Dietl, M., Kleiner, A. and Nebel, B.: CS Freiburg: Coordinating Robots for Successful Soccer Playing, IEEE Transactions on Robotics and Automation, 18(5),2002. 685-699.

Emergent Cooperation in RoboCup: A Review

Geoff Nitschke

Computational Intelligence Group
Department of Computer Science, Faculty of Sciences
De Boelelaan 1081a, 1081 HV Amsterdam
The Netherlands
nitschke@cs.vu.nl

Abstract. This article presents a survey of prevalent research results pertaining to emergent cooperation in RoboCup soccer. Results reviewed maintain particular reference to research that uses biologically inspired design principles and concepts, such as emergence and evolution, as a means of attaining cooperative behavior. The core of this article argues that even though emergent cooperative behavior derived within RoboCup (and the larger field of multi-robot systems) is still in its infancy, it holds considerable future potential, as a problem solver in domains where systems comprising many interacting components must cooperatively solve a global task.

1 Introduction

To date, research that qualitatively measures and evaluates mechanisms that underlie and motivate emergent cooperative behavior in real world[1] and artificial systems remains largely in stage of research infancy. The concept of emergent behavior has propagated many ideas about emergent cooperative behavior in biological systems, and roboticists and computer scientists alike have now adopted these ideas. Early research in decentralized systems [4], [5] suggested that complexity at a group level might be attainable with very simple individual agents, with no need for central control. A derivative of this idea is to use a biologically inspired design approach to engineering group behaviors in multi-robot systems. Such a design approach utilizes concepts such as evolution and self-organization with the goal of a global behavior emerges from interaction of the systems components [6], [7]. It is argued by certain researchers [8], [9] that the use of biologically inspired principles such as evolution and emergence are needed in the purposeful design of complex artificial systems in order to replace ineffective preprogrammed and centralized design methodologies. With few exceptions, and then only in multi-robot systems containing relatively few robots [10], the majority of research in emergent cooperative behavior is restricted to simulated problem domains given the inherent complexity of applying evolutionary design principles to collective behaviors in groups of real robots [11].

An important future direction that the survey emphasizes is the gaining of insightful knowledge into design algorithms for emergent cooperation. If emergent cooperative

[1] The emergence of cooperation has also been studied in a disparity of fields, such as ecological modeling [1], game theory [2], and economics [3].

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 512 – 520, 2006.

behavior was sufficiently understood, purposeful design of cooperative behavior could be applied to benefit a variety of application domains including telecommunications [12] and space exploration [13]. As a final introductory note, in the literature various researchers have adopted the use of nomenclature that is ambiguous in defining the term *cooperation*. Such terminology often suffers from the frame of reference problem [14], as definitions correspond to the perspectives and interests of the researchers conducting the study. Thus, for the purposes of this survey, a definition of cooperation is not provided, but rather a survey of research that uses biologically inspired design principles as a means of deriving cooperative behavior between two or more robots (players) given a specific task in the RoboCup domain.

2 Emergent Cooperative Behavior in RoboCup

Within RoboCup, various leagues, defining the types of robots used as players, and the types of game scenarios played, currently exist as research initiatives [15], [16], [17], [18], [19], [20], where each league maintains its own set of technical challenges and engineering accomplishments. One of the functions of RoboCup is that it provides a research platform for explorations in designing group behaviors (such as cooperation) that can potentially be applied to the broader field of multi-robot systems. Such research explorations have been developed in simulation [22], [23], [24], [26], [27], [28], [29], [30], [31], [32], [33], [34] as well as with real robots [35], [36], [37]. For example, Veloso *et al.* [35] used a team of small and autonomous two-wheeled robots, while Veloso and Uther [36] used a team of Sony AIBO ® robot dogs, to play in a RoboCup soccer tournament. It is obvious from these latter experiments that robotic systems provide a degree of realism that is never possible in simulation, though as a complementary research tool, simulators allow researchers to investigate issues such as cooperation, via implementing abstract and complex behaviors. This article surveys only emergent cooperative behavior research in simulated RoboCup systems.

2.1 Cooperative Passing Behavior with Neural Networks

Noda *et al.* [21], [31] used a RoboCup simulator [26] as a test-bed for the learning of cooperative behavior within groups of soccer agents. Learnt cooperative behavior took the form of one soccer agent learning when pass to a teammate and when to shoot the ball at the opponent goal area. Agents used a neural network with thirty hidden neurons and a back propagation method [38] to learn in which situations it was better to cooperate and in which situations it was better not to cooperate, according to the evaluation criteria of the number of goals scored, and the time taken to score. The learning approach was supervised given that over the course of several hundred training scenarios, a coaching agent provided a positive feedback signal when one of the offense players scored a goal, and a negative feedback signal when a shot aimed at the opponent goal area failed or a time limit expired. The authors illustrated that training the neural network using the back propagation method allowed the success rates of the shoot and pass actions to increase as agents learnt when to pass and when to shoot the ball depending upon the position of a defensive player relative to the goal area and other agents. Learnt cooperative behavior was evaluated in terms of the time taken to score a goal as well as the number of goals scored.

The key criticism of this research is that cooperative behavior was limited to a learnt decision making process for a single soccer agent, where the decision to pass or not defined cooperative behavior. The agents, environment, and learning mechanism were kept simple, so that this form of cooperative behavior could be successfully learnt. In their conclusions, the authors justified using a simple neural network learning mechanism as it provided a good starting point for the learning of more complex forms of cooperative behavior that would potentially be applicable to an entire team of soccer agents.

2.2 Cooperative Team Behaviors with Layered Learning

Stone and Veloso [32] introduced a layered learning approach to cooperative behavior, where soccer agents used neural networks to initially learn low-level individual behaviors such as intercepting a ball, and then decision trees [39] to learn higher-level cooperative behaviors such as deciding when, and to which soccer agent to pass the ball to. The layered learning approach was implemented within the RoboCup server [26] as a simulated environment, and allowed for a bottom-up definition of soccer agent capabilities at both the individual and team level. That is, learnt low-level individual behaviors formed the basis for, and were incorporated as part of, higher-level team behaviors. In various game scenarios, the layered learning approach provided the capability for a group of offensive soccer agents to cooperate via making strategic passes to each other, such that the probability of scoring a successful shot at the opponent goal area would be maximized.

The research of Stone and Veloso [25] extended their previous work and elaborated upon their approach for having a soccer agent decide if to cooperate with teammates, via passing the ball, or to not cooperate via shooting the ball directly at the opponent goal area. In this research, the authors used the layered learning approach to design an action selection mechanism that allowed soccer agents to anticipate if cooperation with a particular teammate would be advantageous. This action selection mechanism was implemented in several experiments, and proved successful in encouraging cooperative behavior, and outperformed an approach for team control that did not utilize this action selection mechanism. A comparison in performance was made in terms of the evaluation criteria, of the number of goals scored, and the time for which a team maintained control of the ball.

The cooperative team-level behaviors described in this series of research papers were not emergent in the sense that is typically referred to in artificial life literature, as these cooperative team-level behaviors relied largely upon individual agents learning action selection mechanisms based upon decision tree confidence factors. Cooperative behavior was emergent in the sense that a series of decisions by individual soccer agents regarding whether to pass the ball or not formed a team-level behavior that was more successful in terms of goals scored and the time for which the team maintained control of the ball. In many experiments, the game scenarios tested did not reflect a complete range of scenarios that would be required in an actual RoboCup soccer match, and in certain cases it was unclear if the cooperative behavior exhibited would generalize to a broader class of game situations. Though, the layered learning approach provided an excellent methodology for the learning of cooperative behavior in a task environment where its inherent complexity prevented the derivation of a direct mapping from sensors to actuators via the use of more traditional learning methods.

2.3 Cooperative Behaviors with Neuro-evolution

In the research theme of using artificial evolution to derive cooperative behavior within a team of soccer agents, Whiteson *et al.* [27] compared and evaluated two different neuro-evolution approaches for the synthesis of cooperative behavior. These methodologies attempted to derive cooperative behavior within a group of three soccer agents for the *keep-away soccer* task environment [28], [29], [30]. Using these neuro-evolution methodologies, soccer agents first learnt a small number of sub-tasks that were then combined, as dictated by an artificial evolution process, such that an overall complex behavior emerged. The authors compared two neuro-evolution approaches for evolving cooperative behavior amongst a team of keeper [28], [29], [30] soccer agents. In the first approach, genome strings encoded synaptic weights of a population of complete neural network controllers. These genomes were evaluated in each generation, the fittest individuals selectively reproduced, and subsequently propagated throughout the evolutionary process. In the second approach, the *enforced sub-populations* method [40] was used to evolve sub-populations of neurons, instead of evolving complete controllers as in the first approach.

Results elucidated that both approaches evolved a successful neural network controller, though the *enforced sub-populations* approach performed significantly better in terms of the evaluation measures defined for the keep-away soccer task and in facilitating emergent cooperative behavior. Although the *enforced sub-populations* approach proved superior in these experiments, an obvious criticism of this approach is that for more difficult tasks, for example those not executed in a grid-world environment, the solution space would be too large for an artificial evolution algorithm to search effectively and construct an appropriate controller.

2.4 Cooperative Behaviors with Layered Learning and Genetic Programming

Hsu and Gustafson [29], [30] investigated a methodology for facilitating emergent cooperative behavior using the keep-away soccer task. The methodology combined layered learning [32] and genetic programming [41] approaches. The authors argued that using a layered learning approach to genetic programming, as opposed to a pure genetic programming approach, team-level behaviors such as cooperation could readily be derived.

In several experiments, the authors compared a standard genetic programming approach [41] with their methodology that combined layered learning and genetic programming. These experiments highlighted that the layered learning approach was able to more quickly evolve cooperative behavioral strategies within the team of keepers, and with a higher fitness comparative to the standard genetic programming approach. The authors argued that the layered learning approach allowed for a workable decomposition of a complex problem into many readily solvable sub-problems, and that for each of these sub-problems, corresponding fitness functions were readily identifiable.

Using the layered learning approach, team-level behavior was formed via the evolution of complementary keeper strategies, such that when the three keepers interacted with each other, an overall cooperative behavior emerged. This cooperative team-level behavior was derived in a bottom-up manner, where keepers first learnt the skills necessary to cooperate as a pair of players, and then as a team of three players.

The key criticism of this research is that only homogenous teams were evolved, and team level cooperative behavior was derived from the use of only two layers in the layered learning approach. Specifically, only two low-level behaviors were used in the derivation of team-level behaviors. Also, the use of a grid world placed severe limitations on the form of cooperative team-level behavior that could be evolved.

2.5 Cooperative Team Behaviors with Genetic Programming

In a similar theme of research, Luke *et al.* [33] implemented genetic programming techniques within a RoboCup simulator, in an attempt to evolve cooperative behavior within an entire team of eleven soccer agents. The performance of different genetic programming techniques were compared for the derivation of cooperative behavior, where such cooperative behavior was evaluated according to the criteria of the number of goals scored by the team, the number of successful passes, and the period of time for which the team maintained control of the ball.

Luke *et al.* [33] used the *Strongly Typed Genetic Programming* (STGP) technique [42] to simultaneously test entire teams of soccer agents against each other in competitive co-evolution scenarios. After many generations of the co-evolutionary process, cooperative behavior emerged within each of the competing teams that effectively combined offensive and defensive team-level strategies. In a final set of evolutionary runs, cooperative team-level behaviors were evolved such that different groups of soccer agents within each team, cooperated with each other in a complementary manner, via simultaneously defending the goal area, and dispersing throughout the field holding certain positions so as to increase the chance of receiving a pass from fellow soccer agents. Such cooperative behaviors prevented the soccer agents from interfering with each other, as had occurred in early evolutionary runs, where many agents were closely gathered about the ball in an attempt to gain control of it.

The key problem with these experiments was that they relied purely upon a competitive co-evolution process, and the functionality of genetic programming in order to produce cooperative behavior within a team of soccer agents. Meaning that in order to evolve team-level cooperative behavior within a feasible amount of time, several constraints had to be placed on the artificial co-evolution process, such as limited population sizes, and teams composed of functionally simple agents. Additionally, cooperative behaviors that emerged under the co-evolutionary process could only be analyzed from a purely observational perspective. That is, fitness comparisons between the competing teams only illustrated progress and counter progress of emergent cooperative behaviors, and did not correspond to ‘true’ evolutionary progress [43] given the fitness landscape of both teams were continuously changing due to the Red Queen affect [44].

3 Future Directions

The aim of the survey was not to provide an exhaustive list or compilation of all research in RoboCup, but rather to highlight prevalent examples of when emergent cooperation facilitates a solution that would not otherwise be attainable. The survey primarily argued that the majority of RoboCup emergent cooperative behavior research utilized simple or limited game tasks usually involving only two or three simulated robots. It is evident from the literature that design methodologies based upon concepts such as artificial evolution, are deemed by many researchers, to be an effective means for investigating the conditions under which cooperation emerges. Unfortunately, current RoboCup systems lack a proven methodology that allows the transfer of emergent cooperative behavior design mechanisms to algorithms that can be used effectively in real world multi-robot systems. Additionally, the use of evolutionary computation was highlighted in many cases as being an effective means for the derivation of cooperative behavior. Though, use of artificial evolution is still largely in a stage of research infancy, so evolution of cooperation is currently limited to simple forms in task scenarios (aspects of the game only) comprised of simple individuals.

In the research reviewed, several key open problems were identified. These problems were not constrained by the nature of RoboCup, but rather by the infancy of the biologically inspired design mechanisms and a lack of analytical methods and techniques. It is obvious that if emergent cooperative behavior derived from RoboCup is to be used to any great benefit, especially in real-world multi-robot systems, then it is important that future research address certain open problems. For example, design methodologies for achieving desired emergent cooperation would ideally need to be scalable and defined by algorithms and methods of analysis that are equally applicable in the physical world. Given the early stage of research and development of RoboCup and the relative infancy of the notion of emergent cooperation as a means of solving multi-robot tasks, it is justifiable that standardized methods for deriving, testing, proving the convergence of, and evaluating emergent cooperative behavior do not yet exist.

References

1. Dugatkin, L. (1990). N-Person games and the evolution of cooperation: A model based on predator inspection in fish. *Journal of Theoretical Biology*, 142(1), 123-135.
2. Akiyama, E., and Kunihiko, K. (1995). Evolution of Cooperation, Differentiation, Complexity, and Diversity in an Iterated Three-Person Game. Artificial Life 2(1): 293-304.
3. Axelrod, R. (1984). *The Evolution of Cooperation*. New York, NY: Basic Books.
4. Cao, Y., Fukunaga, A., & Kahng, A. (1997). Cooperative Mobile Robotics: Antecedents and Directions. *Autonomous Robots* 4, 7-27.
5. Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press.
6. Kube, C., & Zhang, H. (1993). Collective Robotics: From Social Insects to Robots. *Adaptive Behavior*, 2(1), 189-218.
7. Kube, C., & Zhang, H. (1997). Task Modeling in Collective Robotics. *Autonomous Robots*, 4(1), 53-72.

8. Bongard, J. and Pfeifer, R. (2003). Evolving Complete Agents Using Artificial Ontogeny. In F. Hara, & R. Pfeifer, R (Eds.), *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*. Berlin, Germany: Springer-Verlag.
9. Harvey, I. (1997). Artificial Evolution and Real Robots. In S. Masanori (Ed.), *Proceedings of International Symposium on Artificial Life and Robotics* (pp. 18-20). Berlin, Germany: Springer-Verlag.
10. Mataric, M. (1992). Designing emergent behaviors: From local interactions to collective intelligence. In, J. Meyer, H. Roitblat, & S. Wilson (Eds.), *From Animals to Animats 2: Second International Conference on the Simulation of Adaptive Behavior* (pp. 432-441). Cambridge, MA: The MIT Press.
11. Floreano, D., & Nolfi, S. (2000). *Evolutionary Robotics*. Cambridge, MA: The MIT Press.
12. Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed Stigmergic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9, 317-365.
13. Brooks, R., & Flynn, A. (1989). Fast, Cheap and Out of Control: A Robot Invasion of the Solar System. Journal of the British Interplanetary Society 1, 478–485.
14. Pfeifer, R., and Scheier, C. (1999). Understanding Intelligence. Cambridge, MA: The MIT Press.
15. Veloso, M., Bowling, M., & Stone, P. (2000). The CMUnited-98 champion small-robot team. *Advanced Robotics*, 13(1), 753-767.
16. Kim, J. (1996). *Proceedings of the Micro-Robot World Cup Soccer Tournament*. Taejon, Korea: Elsevier Science.
17. Noda, I., & Stone, P. (2001). The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS Research. *Autonomous Agents and Multi-Agent Systems*, 7(1), 101–120.
18. Jamzad, M., Foroughnasirai, A., Kazemi, M., Ghorbani, R., Chiniforushan, E., & Chitsaz, H. (2000). Middle Sized Soccer Robots: ARVAND. In *Lecture Notes in Artificial Intelligence RoboCup-99: Robot Soccer World Cup III* (pp. 61-73). Berlin, Germany: Springer-Verlag.
19. Kitano, H, Fujita, M., Zrehen, S., & Kageyama, K. (1998). Sony Legged Robot for RoboCup Challenge. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 2605-2612). Leuven, Belgium: IEEE Press.
20. Kitano, H., & Asada, M. (2000). The RoboCup Humanoid Challenge as the Millennium Challenge for Advanced Robotics. *Advanced Robotics*, 13(1), 723-736.
21. Noda, I., Matsubara, H., & Hiraki, K. (1996). Learning of cooperative actions in multiagent systems: a case study of pass and play in soccer. In, S. Sandip (Ed.), *Adaptation, Co-evolution, and Learning in Multi-agent Systems: Papers from the 1996 AAAI Spring Symposium* (pp. 63-67). Boston, MA: AAAI Press.
22. Stone, P., & Veloso, M. (1998). The CMUNITED-97 Simulator Team. In H. Kitano (Ed.), *RoboCup-97: Robot Soccer World Cup* (pp. 389-397). Berlin, Germany: Springer-Verlag.
23. Stone, P., & Veloso, M. (1999). Task decomposition, dynamic role assignment, and low bandwidth communication for real time strategic teamwork. *Artificial Intelligence*, 110(2), 241-273.
24. Stone, M., Veloso, M., & Bowling, M. (1999). Anticipation as a Key for Collaboration in a Team of Agents: A Case Study in Robotic Soccer. In G. McKee and P. Schenker (Eds.), *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II* (pp. 134-141). Bellingham, WA: SPIE Press.
25. Stone, P., & Veloso, M. (1998). Towards Collaborative and Adversarial Learning: A Case Study in Robotic Soccer. *Evolution and learning in multi-agent syste*ms, 48(1), 83-104.

26. Noda, I., Matsubara, H., Hiraki, K., & Frank, I. (1998). Soccer Server: A Tool for Research on Multi-agent Systems. *Applied Artificial Intelligence*, 12(1), 233-250.
27. Whiteson, S. Kohl, N. Miikkulainen R., & Stone, P. (2003). Evolving Keep away Soccer Players through Task Decomposition. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 356-368). Chicago, IL: Springer-Verlag.
28. Di Pietro, A., While, L., & Barone, L. (2002). Learning In RoboCup Keep away Using Evolutionary Algorithms. In W. Langdon, E. Cantupaz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, N. Jonoska (Eds.), *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 1065-1072). New York, NY: Morgan Kaufmann.
29. Hsu, W, & Gustafson, S. (2001). Layered Learning in Genetic Programming for a Cooperative Robot Soccer Problem. In J. Miller, M. Tomassini, P. Lanzi, C. Ryan, A. Tetamanzi, & W. Langdon (Eds.), *Proceedings of the Fourth European Conference on Genetic Programming* (pp. 291-301). Como, Italy: Springer-Verlag.
30. Hsu, W, & Gustafson, S. (2002). Genetic Programming and Multi-Agent Layered Learning by Reinforcements. In W. Langdon, E. Cantupaz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, N. Jonoska (Eds.), *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 764-771). New York, NY: Morgan Kaufmann.
31. Noda, I., Matsubara, H., & Hiraki, K. (1996). Learning Cooperative Behavior in Multiagent Environment: a case study of choice of play-plans in soccer. In N. Foo, & R. Goebel (Eds.) *Proceedings of the Fourth Pacific Rim International Conference on Artificial Intelligence* (pp.570-579, Cairns, Australia: Springer-Verlag.
32. Stone, P., & Veloso, M. (1998). A Layered Approach to Learning Client Behaviors in the RoboCup Soccer Server. *Applied Artificial Intelligence*, 12, 165-188.
33. Luke, S., Hohn, C., Farris, J., Jackson, G., & Hendler, J. (1998). Co-evolving Soccer Softbot Team Coordination with Genetic Programming. In H. Kitano (Ed.), *RoboCup-97: Robot Soccer World Cup I* (pp. 398-411). Berlin, Germany: Springer-Verlag.
34. Stone, P., and Veloso, M. (1998). Using Decision Tree Confidence Factors for Multiagent Control. In H. Kitano (Ed.), *Proceedings of the Second International Conference on Autonomous Agents* (pp. 110-116).
35. Veloso, M., Bowling, M., Achim, S., Han, K., & Stone, P. (1999). The CMUnited-98 champion small robot team. In M. Asada., & H. Kitano (Eds.), *RoboCup-98: Robot Soccer World Cup II* (pp. 77-92). Berlin, Germany: Springer Verlag.
36. Veloso, M., & Uther, W. (1999). The CMTrio-98 Sony legged robot team. In M. Asada., & H. Kitano (Eds.), *RoboCup-98: Robot Soccer World Cup II* (pp. 491-497). Berlin, Germany: Springer Verlag.
37. Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara, H., & Osawa, E. (1997). RoboCup: A challenge problem for AI. *AI Magazine*, 18(1), 73-85.
38. Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart, & J. McClelland (Eds.), *Parallel Distributed Processing*, *Volume 1: Foundations* (pp. 318-362). Cambridge, MA: The MIT Press.
39. Mitchell, T. (1997). *Machine Learning*. Berkshire, England: McGraw Hill.
40. Bryant, B., & Miikkulainen, R. (2003). Neuro-evolution for Adaptive Teams. In *Proceedings of the 2003 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press.

41. Koza, J. (1992). Evolution of subsumption using genetic programming, In F. Varela and P. Bourgine, *Proceedings of the First European Conference on Artificial Life* (pp. 110-119). Cambridge, MA: The MIT Press.
42. Montana, D. (1995). Strongly typed genetic programming. *Evolutionary Computation*, 3(2), 199-230.
43. Floreano, D., & Nolfi, S. (1997). God save the Red Queen! Competition in co-evolutionary robotics. In J. Koza, D. Kalyanmoy, M. Dorigo, D. Fogel, M. Fogel, M. Garzon, H. Iba, & R. Riolo (Eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference*. San Francisco, CA: Morgan Kaufmann.
44. Van Valen, L. (1973). A new evolutionary law. *Evolution Theory*, 1, 1-30.

Flexible Coordination of Multiagent Team Behavior Using HTN Planning

Oliver Obst and Joschka Boedecker

Universität Koblenz-Landau, AI Research Group, 56070 Koblenz, Germany
{fruit, jboedeck}@uni-koblenz.de

Abstract. The domain of robotic soccer is known as a highly dynamic and non-deterministic environment for multiagent research. We introduce an approach using Hierarchical Task Network planning in each of the agents for high-level coordination and description of team strategies. Our approach facilitates the maintenance of expert knowledge specified as team strategies separated from the agent implementation. By combining high level plans with reactive basic operators, agents can pursue a grand strategy while staying reactive to changes in the environment. Our results show that the use of a planner in a multiagent system is both possible and useful despite the constraints in dynamic environments.

1 Introduction

Coordination among different agents in a multiagent system (MAS) is considered as one of the most import challenges in order to achieve a common goal. This task becomes increasingly difficult in highly dynamic, partially observable, and adversarial environments such as robotic soccer. Noisy sensor data and imperfect actuators further add to this problem.

For our work, we have in mind a multiagent system where the user specifies not only a general task for the system, but also the way specific situations are handled. On the other hand, the user can expect a certain set of available primitives that handle simple tasks on their own. The idea is that a designer can change the team behavior for specific situations. With the widely used reactive approaches, changes are complicated because of interdependencies so that simple changes can have impact on more than one situation. Another observation we made with previous approaches was the difficulty to specify strategies for reactive multiagent teams. Even though reactivity is a key quality in soccer, long term strategies seem to be as useful. Classical planning approaches are a solution to compute actions that lead towards given goals, but are in general regarded as not applicable to highly dynamic multiagent scenarios.

In this work, we suggest to use Hierarchical Task Network (HTN) planners in each of the agents in order to achieve coordinated team behavior, while at the same time our agents should always follow the strategy as suggested by the human expert. The expert knowledge should be separated from the rest of the agent code, in a way that it can be easily specified and changed. While pursuing the given strategy, the agents should keep as much of their reactiveness

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 521–528, 2006.

as possible. Different levels of detail in the description of strategies facilitate the generation of useful information for debugging or synchronization.

HTN planning is based upon work presented in [15]. It makes use of domain knowledge to speed up the planning and is able to solve classical planning problems orders of magnitude more quickly than classical planners if provided with a good set of HTNs. In our work, we followed the formalization of soccer domain knowledge in [5] but adapted it for the use in HTN planning.

Furthermore, we show how it is possible to use an HTN planner in the domain of robotic soccer, even though it is very different from environments used in classical planning with deterministic operators and a single planning agent being the only reason for changes in the world. For our approach, we have chosen a team of agents in the RoboCup Soccer Simulation League 3D [8].

We start by reviewing relevant related work, and then describe our approach in detail, outlining problems that arose while trying to adapt the planner to be used in the MAS, and the solutions to overcome them. We present and discuss the results of our first tests and give directions for future work.

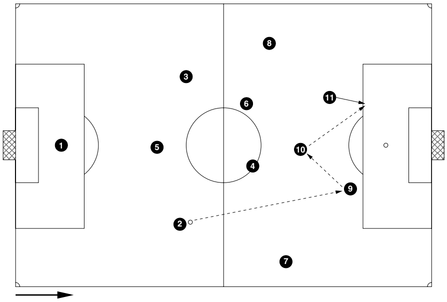
2 Related Work

Several approaches that use a planning component in a MAS can be found in the literature. In [4], the authors describe a formalism to integrate the HTN planning system SHOP [13] with the IMPACT [17] multiagent environment (A-SHOP). While the environment of this work clearly is a multiagent system, the planning is carried out centralized by a single agent. This is a contrast to our approach, which uses a planner in each of the agents to coordinate their actions.

Bowling *et al.* [2] presents a strategy system that makes use of *plays* to coordinate team behavior of robots in the RoboCup Small Size League. Multiple plays are managed in a *playbook* which is responsible to choose appropriate plays, and evaluate them for adaption purposes. Effects of individual plays are not specified due to the difficulties in predicting the outcome of operators in the dynamic environment. This is in contrast to our approach, as we use *desired effects* of the operators in our plans as described in section 3. The planning component in [2] is also centralized.

In [9], the use of *coordination graphs* [7] to coordinate a team of agents in dynamic environments without explicit communication is proposed. The coordination graphs are applied to the continuous domain of robotic soccer where a discretization of the state is achieved by assigning roles to the different agents, similar to the approach in [16]. The coordination graphs are then built on the derived set of roles.

In [11, 12], the behavior of agents in a Multiagent System is specified using UML statecharts. Agents are designed in a top-down manner with a layered architecture. At the highest level global patterns of behavior are specified in an abstract way. Explicit specification of cooperation and multiagent behaviors can be realized.



```
method diagram_4
pre [ is_current_formation(F),
      #formation(F,A,back,_,Side),
      #formation(F,B,forward,_,Side),
      #formation(F,C,_,attacking,center),
      #formation(F,D,forward,_,OtherSide),
      #different(Side,OtherSide),
      #different(Side,center)
    ]
subtasks [pass(A,B),
          pass(B,C),leading_pass(C,D)].
```

Fig. 1. Building up play with Diagram #4 from [10] (left) and its notation as HTN method for our planner (right). Preconditions prefixed with '#' denote function calls.

3 A Planner for Soccer Agents

The main focus of our work was to achieve high-level coordination for a team of several agents in a dynamic environment. All planning should be done in a distributed fashion. The system should allow for easy specification of team plans, and automatically generate individual actions for the agents during execution.

In [5], we present an approach to model soccer knowledge, as it can be found in soccer theory books. The focus of this work was the representation of diagrams used in [10] to describe moves to be used in specific phases of the match. We derive an ontology from [10] which breaks down the top level task *play soccer* into subtasks by means of aggregation and specialization. A subset of this ontology can be found in Tab. 1. The hierarchy of the tasks and subtasks facilitates the collection of useful debugging output during development.

In Hierarchical Task Network (HTN) planning, the objective is to perform tasks. Tasks can be complex or primitive. HTN planners use *methods* to expand complex tasks into subtasks, until the tasks are primitive. Primitive tasks can be performed directly by using planning operators. Changes to basic HTN planning algorithms are necessary in order to use it for the soccer domain, because here the outcome of operators is not deterministic.

Because it is impossible to foresee how the world will look like after a few actions, our planner generates what is called *plan stub* in [1], a task network with a primitive task as the first task. As soon as a plan stub has been found, an agent can start executing its task. The algorithm in Fig. 2 expands a list of

Table 1. Some complex tasks with subtasks ('|': alternatives; ',': sequences)

play_soccer	**offensive_phase** | defensive_phase | *nothing*
offensive_phase	upon_ball_possession, **build_up_play**, final_touch, shooting
build_up_play	**build_up_play_long_pass** | build_up_play_diagonal_pass
build_up_play_long_pass	diagram_3 | **diagram_4** | ...
diagram_4	pass(A,B), **pass(B,C)**, leading_pass(C,D)
pass(B,C)	do_pass(B) | do_receive_pass | do_positioning

```
**Function**: plan($s_{now}, < t_1, ..., t_k >, O, M$)
**Returns**: $(w, s)$, with $w$ an ordered set of tasks, $s$ a state; or *failure*
if $k = 0$ then return $(\emptyset, s_{now})$  // i.e., the empty plan
if $t_1$ is a pending primitive task then
    active ← $\{(a, \sigma) | a$ is a ground instance of an operator in O,
              $\sigma$ is a substitution such that $a$ is relevant for $\sigma(t_1)$,
              and $a$ is applicable to $s_{now}\}$;
    if active $= \emptyset$ then return failure;
    nondeterministically choose any $(a, \sigma) \in$ active;
    return $(\sigma(< t_1, ..., t_k >), \gamma(s_{now}, a))$;
else if $t_1$ is a pending complex task then
    active ← $\{m | m$ is a ground instance of a method in $M$,
             $\sigma$ is a substitution such that $m$ is relevant for $\sigma(t_1)$,
             and $m$ is applicable to $s_{now}\}$;
    if active $= \emptyset$ then return failure;
    nondeterministically choose any $(m, \sigma) \in$ active;
    $w \leftarrow$ subtasks$(m).\sigma(< t_1, ..., t_k >)$;
    set all tasks in front of $t_1$ to pending, set $t_1$ to expanded;
    return plan($s_{now}, w, O, M$);
else
    // $t_1$ is an already executed expanded task and can be removed
    return plan($s_{now}, < t_2, ..., t_k >, O, M$);
```

Fig. 2. Creating an initial plan stub (Notation according to [6])

tasks to a plan stub. At the same time, the desired successor state is computed and returned. The second algorithm (see Fig. 3) removes executed tasks from the plan and uses the first algorithm then to create an updated plan stub.

To handle non-determinism, we treat a plan as a stack. Tasks on this stack are marked as either *pending* or as *expanded*. Pending tasks are either about to be executed, if they are primitive, or waiting to be further expanded, if they are complex. Tasks marked as expanded are complex tasks which already have been expanded into subtasks. If a subtask of a complex task fails, all the remaining subtasks of that complex task are removed from the stack and it is checked if the complex task can be tried again. If a task was finished successfully, it is simply removed from the stack.

Our plan operators realizing primitive tasks describe only the *desired effects* of an action. Using desired effects of an action we can check if the action was executed successfully. Simultaneously executed actions which are not part of the desired effects of the operator are simply ignored in the description.

To give an example, we assume that the planner is about to plan for a situation like in Fig. 1 and the complex task `play_soccer` has already been partially expanded as shown in Fig. 4 (left). All the pending tasks in Fig. 4 are still complex tasks. At this level of expansion, the plan still represents a team plan, as seen from a global perspective. The team task `pass(2,9)` will expand to `do_pass(9)` for agent #2, agent #9 has to do a `do_receive_pass` for the same team task. The other agents position themselves relatively to the current ball

```
**Function**: step($s_{expected}, s_{now}, < t_1, ..., t_k >, O, M$)
**Returns**: $(w, s)$, with $w$ a set of ordered tasks, $s$ a state; or *failure*
if $k = 0$ then return $(\emptyset, s_{now})$  // i.e., the empty plan
if $t_1$ is a pending task then
    if $s_{expected}$ is valid in $s_{now}$ then
        $i \leftarrow$ the position of the first non-primitive task in the list;
        return plan($s_{now}, < t_i, ..., t_k >, O, M$);
    else
        // $t_1$ was unsuccessful; remove all pending children of our parent task
        return step($s_{expected}, s_{now}, < t_2, ..., t_k >, O, M$);
else
    // $t_1$ is an unsuccessfully terminated expanded task, try to re-apply it
    active $\leftarrow \{m | m$ is a ground instance of a method in $M$,
                $\sigma$ is a substitution such that $m$ is relevant for $\sigma(t_1)$,
                and $m$ is applicable to $s_{now}\}$;
    if active $= \emptyset$ then
        // $t_1$ cannot be re-applied, remove it from the list and recurse
        return step($s_{expected}, s_{now}, < t_2, ..., t_k >, O, M$);
    else
        nondeterministically choose any $(m, \sigma) \in$ active;
        $w \leftarrow$ subtasks($m$).$\sigma$($< t_1, ..., t_k >$);
        set all tasks in front of $t_1$ to pending, set $t_1$ to expanded;
        return plan($s_{now}, w, O, M$);
```

Fig. 3. Remove the top primitive tasks and create a new plan stub

```
pending-pass(2,9)
pending-pass(9,10)
pending-leading-pass(10,11)
expanded-diagram-4
expanded-build_up_long_pass
expanded-build_up_play
pending-final_touch
pending-shooting
expanded-offensive_phase
expanded-play_soccer
```

```
method pass(A,B)
pre [my_number(A)]
subtasks [do_pass(B) with pass(we,A,B),
          do_positioning].

method pass(A,B)
pre [my_number(B)]
subtasks [do_receive_pass with pass(we,A,B)].

method pass(A,B)
pre [my_number(C),#\=(A,C),#\=(B,C)]
subtasks [do_positioning with pass(we,A,B)].
```

Fig. 4. Plan stack during planning (left) and different methods to reduce the team task `pass(A,B)` to agent tasks (right)

position with `do_positioning` at the same time. The desired effect of `pass(2,9)` is the same for all the agents, even if the derived primitive task is different depending on the role of the agent. To express that an agent should execute the `do_positioning` behavior while taking the effect of a simultaneous pass between two teammates into account, we are using terms like `do_positioning with pass(we,2,9)` in our planner. Figure 4 (right) shows methods reducing the team task `pass(A,B)` to different primitive player tasks.

```
pending-(do_receive_pass with        pending-(do_positioning with
         pass(we, 2, 9)),                     pass(we, 2, 9)),
expanded-pass(2, 9),                 expanded-pass(2, 9),
pending-pass(9, 10),                 pending-pass(9, 10),
pending-leading_pass(10, 11),        pending-leading_pass(10, 11),
expanded-diagram-4,                  expanded-diagram-4,
...                                  ...
```

Fig. 5. Step 1: Plan Stubs for player 11 and player 9 (see also Fig. 1)

```
pending-(do_pass(10) with            pending-(do_positioning with
         pass(we, 9, 10)),                    pass(we, 9, 10)),
pending-do_positioning,              expanded-pass(9, 10),
expanded-pass(9, 10),                pending-leading_pass(10, 11),
pending-leading_pass(10, 11),        expanded-diagram_4,
expanded-diagram_4,                  ...
...
```

Fig. 6. Step 2: Plan Stubs for player 11 and player 9

In different agents, the applicable methods for the top team task `pass(2,9)` lead to different plan stubs. This is an important difference to the work presented in [1]. The plan stubs created as first step for agent 9 and agent 11 are shown in Fig. 5. When a plan stub is found, the top primitive tasks are passed to the `C++` module of our agent and executed. The agent has to execute all pending primitive tasks until the next step in the plan starts. If there are pending primitive tasks after one step is finished, these agent tasks are simply removed from the plan stack and the next team task can be expanded. Figure 6 shows the plan stub for the second step from the diagram in Fig. 1. For player 11, the expansion leads to a plan stub with two primitive tasks in a plan step while for player 9 there is only one task to be executed. Each step in plans for our team stops or starts with an agent being in ball possession. If any of the agents on the field is in ball possession, we can check for the desired effect of the previous action.

4 Results and Discussion

For our approach of generating coordinated actions in a team we implemented an HTN planner in Prolog which supports interleaving of planning and acting. Our planner supports team actions by explicitly taking the effects of operators simultaneously used by teammates into account. The planner ensures that the agents follow the strategy specified by the user of the system by generating individual actions for each of the agents that are in accordance with it. The *lazy evaluation* in the expansion of subtasks which generates plan stubs rather than a full plan, makes the planning process very fast and enables the agents to stay reactive to unexpected changes in the environment. The reactiveness could, however, be increased by adding a situation evaluation mechanism that is used prior to invoking the planner. This would improve the ability to exploit sudden, short-lived opportunities during the game.

Creating strong teams is possible with many approaches. We strongly believe that our approach leads to a modular behavior design and facilitates rapid specification of team behavior for *users* of our agent architecture. Our plans can describe plays as introduced in [2], which have shown to be useful for synchronization in a team. Our approach supports different levels of abstraction in plans. That means there are different levels of detail available to describe what our team and each single agent is actually doing. Additionally, the planner can find alternative ways to achieve tasks. The approach in [2] was used for Small Size League; but for larger teams, more opportunities are possible for which an approach using fixed teammates seems to restrictive. On the other hand, the approach in [2] supports adaptation by changing weights for the selection of successful plays. In our approach, the corresponding functionality could be achieved by changing the order in which HTN methods are used to reduce tasks. At this point in time, our approach does not support this yet. As soon as we do have an adaptive component in our approach, it makes sense to compare results of our team with and without adaptation.

Although more detailed evaluations have to be carried out, the first tests using the planner seem very promising and indicate that our approach provides a flexible, easily extendable method for coordinating a team of agents in dynamic domains like the RoboCup 3D Simulation League.

5 Conclusion and Future Work

We presented a novel approach that uses an HTN planning component to coordinate the behavior of multiple agents in a dynamic MAS. We formalized expert domain knowledge and used it in the planning methods to subdivide the given tasks. The hierarchical structure of the plans speeds up the planning and also helps to generate useful debugging output for development. Furthermore, the system is easily extendable as the planning logic and the domain knowledge are separated.

In order to use the system in RoboCup competitions, we plan to integrate a lot more subdivision strategies for the different tasks as described in the diagrams in [10]. A desirable enhancement to our work would be the integration of an adaption mechanism. Monitoring the success of different strategies against a certain opponent, and using this information in the choice of several applicable action possibilities, as e.g. outlined in [2], should be explored. The introduction of *durative actions* into the planner (see for instance [3]) would give a more fine grained control over the parallelism in the multiagent plans. Finally, we want to restrict the sensors of the agents to receive only partial information about the current world state, and address the issues that result for the distributed planning process.

References

1. Thorsten Belker, Martin Hammel, and Joachim Hertzberg. Learning to optimize mobile robot navigation based on HTN plans. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4136–4141, 2003.
2. Michael Bowling, Brett Browning, and Manuela Veloso. Plays as team plans for coordination and adaptation. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*, Vancouver, June 2004.
3. Alex M. Coddington, Maria Fox, and Derek Long. Handling durative actions in classical planning frameworks. In John Levine, editor, *Proceedings of the 20th Workshop of the UK Planning and Scheduling SIG*, pages 44–58, 2001.
4. Jürgen Dix, Héctor Muñoz-Avila, and Dana Nau. IMPACTing SHOP: Planning in a Multi-Agent Environment. In Fariba Sadri and Ken Satoh, editors, *Proceedings of CLIMA 2000, Workshop at CL 2000*, pages 30–42. Imperial College, 2000.
5. Frank Dylla, Alexander Ferrein, Gerhard Lakemeyer, Jan Murray, Oliver Obst, Thomas Röfer, Frieder Stolzenburg, Ubbo Visser, and Thomas Wagner. Towards a league-independent qualitative soccer theory for RoboCup. In Daniele Nardi, Martin Riedmiller, Claude Sammut, and José Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence*, pages 611–618, Berlin, Heidelberg, New York, 2005. Springer.
6. Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning Theory and Practice.* Morgan Kaufmann, San Francisco, CA, USA, 2004.
7. Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In *In 14th Neural Information Processing Systems (NIPS-14)*, 2001.
8. Marco Kögler and Oliver Obst. Simulation league: The next generation. In Polani et al. [14], pages 458–469.
9. Jelle R. Kok, Matthijs T. J. Spaan, and Nikos Vlassis. Multi-robot decision making using coordination graphs. In A.T. de Almeida and U. Nunes, editors, *Proceedings of the 11th International Conference on Advanced Robotics, ICAR'03*, pages 1124–1129, Coimbra, Portugal, June 2003.
10. Massimo Lucchesi. *Coaching the 3-4-1-2 and 4-2-3-1.* Reedswain Publishing, 2001.
11. Jan Murray. Specifying agent behaviors with UML statecharts and StatEdit. In Polani et al. [14], pages 145–156.
12. Jan Murray, Oliver Obst, and Frieder Stolzenburg. RoboLog Koblenz 2001. In Andreas Birk, Silvia Coradeschi, and Satoshi Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Artificial Intelligence*, pages 526–530. Springer, Berlin, Heidelberg, New York, 2002. Team description.
13. Dana S. Nau, Yue Cao, Amnon Lotem, and Héctor Muñoz-Avila. Shop: Simple hierarchical ordered planner. In *Proceedings of IJCAI-99*, pages 968–975, 1999.
14. Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors. volume 3020 of *Lecture Notes in Artificial Intelligence*. Springer, 2004.
15. Earl D. Sacerdoti. A structure for plans and behavior. American Elsevier, 1977.
16. Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 1999.
17. V. S. Subrahmanian, Piero Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan, and Robert Ross. *Heterogeneous Agent Systems.* MIT Press, 2000.

General-Purpose Learning Machine Using K-Nearest Neighbors Algorithm

Seyed Hamid Hamraz and Seyed Shams Feyzabadi

Department of Computer Engineering, Iran University of Science and Technology,
Tehran, Iran
hamid_hamraz@yahoo.com, sh.feyzabadi@gmail.com

Abstract. The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience. The aim of this paper is to describe a learner machine which can be used in different learning problems without any change in the system. We designed such a machine using *k*-nearest neighbors algorithm. How to optimize *k*-nearest neighbors algorithm to be effectively used in the machine is also discussed. Experimental results are also demonstrated at the end.

1 Introduction

Most of the machine learning jobs we do today are in the shape of solution-finding to a specific problem, in which we know what the exact problem is, we find a solution for that and this solution is specialized for that kind of problem or similar ones with their specific features. But now think about a new learning machine which is designed before defining a learning problem, and it can be applied to a large number of learning problems, or in other words a General-Purpose Learning Machine (GPLM). A GPLM should be suitable for using in different jobs with their different demands, and therefore should have the following characteristics:

1. *Disusing intellectual cost for adapting the machine to a special problem*: It means that the learning machine should receive the training sets as a block-box, and it should not need any other intellectual input about the current problem. For example, in Artificial Neural Networks (ANN) [1], [2], for each specific problem, a topological structure should be determined for the net and this requires a human intellectual power or in other words, designing a systematic algorithm for doing the task, to be used by a computer, would be troublesome.
2. *Fast and online learning*: A GPLM should also satisfy online learning problems.
3. *Fast and efficient response*: A GPLM should use its learned knowledge fast enough, to be utilized in real-time problems.
4. *Additional learning*: A GPLM should be able to receive new training sets any time, with minimum disturbance to the previous learned knowledge.
5. *No limitation for the problem-type it can be applied to*: For example, decision trees [1] are well-suited for discrete values, and they can not deal with real-valued numerals well.

Various learning methods are available, but few of them satisfy the generality points described above. Among them learning networks like ANN, Cerebellar Model

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 529 – 536, 2006.

Articulation Controller (CMAC) [3] and instance-based learning methods [1] are more relevant to our purpose. ANN provides a robust approach to approximating real-valued, discrete-valued, and vector-valued target functions. Nevertheless, as it was mentioned before, it does not satisfy the first, second and fourth points. CMAC, which is another type of learning nets and does local generalization, is a good candidate for applications where low cost embedded solution and high-speed operation with real-time online adaptation are required, e.g., in smart sensors, in measuring systems, etc. However, CMAC uses quantized input, so the number of the possible different input data is finite. It also can not be used for high-dimensional problems effectively. The two latter reasons suppress CMAC from covering the fifth generality point, which is so important for a general system.

Another generation of learning methods which do not provide an explicit definition of target function, after the training session, is instance-based learning algorithms. Generalization beyond training set is postponed until a new instance must be classified. Each time a new query instance is encountered, its relationship to the previous stored examples is examined in order to assign a target function value for the new instance. Instance-based methods are sometimes referred to as "lazy" learning methods because they delay processing until a new instance must be classified. Case-Based Reasoning (CBR) [1] is a more general type of instance-based method. In CBR instances are presented in a rich symbolic form. During the learning phase instances are saved in a database, and each time an unknown instance should be classified, a few *similar instances* are retrieved from the database and through a *generalization process* beyond them, the target value of the unknown instance is estimated. In this method, for each special problem, the presentation manner of the instances and also the definition of *similar instances* should be determined first, and it requires intellectual cost. Therefore K-Nearest Neighbors (KNN) [1] which is a more specific facet of CBR would be a better choice. In KNN the presentation form of the instances is fixed. Instances are presented in the shape of multi-dimensional points and the *similarity* is defined according to Euclidian distance between two instances. By now, we can indicate that KNN satisfies all the generality points above except the third one. This unsatisfactory refers to the "lazy" nature of instance-based methods, and can be covered using multi-dimensional index structures [4], [5]. We'll talk about it and other obstacles, in the way of matching KNN to be efficiently used in GPLM, later.

2 GPLM Structure

2.1 Where Does the Idea Come from?

The GPLM idea comes from human brain structure. Human being can learn various things without changing the structure of his brain. A simple view of the brain from learning point of view is shown in Fig. 1. a. Different samples from different fields enter the brain through the five senses, or maybe a feedback from the brain, and would be memorized. Each time an unknown instance enters; the brain uses its previous experiences and produces the output via some unknown procedures. The Brain should deal with different types of learning problems through a fixed structure. For example, one learning entry is about math and the other is about soccer. According to this issue, we can conclude that the brain uses the same learning procedures for different problems. This conclusion strikes the idea of designing a general machine, which uses the same learning process for different problems, to be utilized for various learning problems or better to say a GPLM.

2.2 GPLM as a Black Box

Overall structure of GPLM as a black box is shown in Fig. 1. b. A set of training instances for a specific problem enters the machine and machine life cycle starts. “Specific” here, does not mean that the machine is designed for special-purpose use. For another learning problem a copy of machine can be used or we can restart the previous one and reinitialize it for new problem. But each time a machine is initialized for a problem; it’ll be specialized to that. The training set, after entering the machine, would be used in order to find the output of unknown instances which get in from another entry. Moreover, the machine is able to learn new instances which are separately presented in the figure. The stippled line which links output to an additional learning entry implies that we can have feedback from output to input. A problem may strike the mind that is this job necessary: to use our estimated outputs for learning while the same procedure will be applied next time and the same output will be estimated? Indeed, it is not necessary and it can also disturb the future estimations, but if next time we need the same value as before, the operation will be sped up and repeated operations won’t be done.

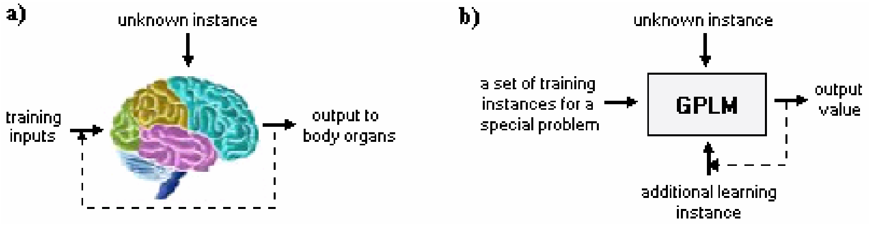


Fig. 1. a) A simple view of human brain from the learning point of view. Note that training inputs may be from different fields, e.g. how to solve a special kind of math problem, how to shoot in soccer, etc. **b)** A GPLM structure as a black box.

2.3 Why KNN?

We introduced GPLM as a black box, and we didn’t discuss what kind of processes is done inside. We proposed KNN for its internal structure and we’ll present an implemented sample of GPLM with this algorithm. But in this section we’ll talk about the philosophy of using KNN in the machine.

ANN and CMAC are imitations of the brain physical structure in a way. A similar structure to the brain is simulated in computer and used for solution-finding to learning problems. In other words, ANN and CMAC are imitations of the brain *hardware*. Now, we’d like to find out what kind of *software* controls the brain or better to say we’d like to mimic the brain *software*. Finding an answer to this question may be so difficult, but good sources which can help us find the answer are ourselves, how we think, how we learn and etc.

Clarifying the discussion, we propound an example. Consider a soccer player, while the ball is traversing in the air. He wants to estimate where the ball hits the ground in order to get there and intercept it. Soccer players do this well. What kind of processing takes place inside the player's mind for this estimation? Suppose that this position depends on the ball's *velocity, angle* and *gravity*. The position can be determined via physics. But does the player use physics formulas? Of course not. Now, suppose that the player is transferred to another planet with a different gravitational force. He can't do the job as well as before at the beginning, but little by little gets better. Therefore, we can firmly express that the player do the job by way of experience. How do the player's experiences help him estimate the solution in the new state? An interesting and reasonable idea is: each time the same event happens, an *illustration* (or maybe a *piece of film*) of that, is saved in player's mind. For a new state, while the ball is traversing in the air, the player recalls some *illustrations* similar to this state[1], and then through a set of comparative operations between the new one and the previous recalled instances, the value of the new state is estimated. The reason we emphasize on the word *illustration* is the astonishing power of human being in imagination and image processing, in which the computer is relatively weak. For example, using a map is easy for a man but if he is given the map in the shape of numeric positions, using it, is almost impossible. Computer works vice versa[2]. The player example gave us an idea about the brain *software*:

* *Instances are saved in the mind in the shape of illustrations. Each time an unknown instance should be predicted, a few similar illustrations are recalled and then through a set of comparative operations the target value of the new instance is predicted.*

Such image processing software in a computer would be so time-consuming and even impractical, because we have a completely different *hardware*: a large amount of neurons in the brain versus digital serial computers. We know that computers are robust in numeric computations and weak in image processing. Thus we are convinced with the idea of comparative operations in the brain and instead of applying image comparison we use numerals. Moreover, recalling similar states in the brain is equivalent to finding the nearest neighbors in KNN algorithm. So the idea above can be altered to the following, to become more feasible for implementing in digital computers:

** *Instances are saved in a database in the shape of multi-dimensional points. Each time an unknown instance should be estimated, a few of the nearest points are retrieved and then through a set of numeric comparisons the target value of the new instance is estimated.*

Comparing the * and the ** ideas, we can indicate that the *software* which controls the brain is similar to CBR, and because of the debate of generality mentioned before, KNN which is a more specific facet of CBR, would be suitable for the current digital computers.

[1] This situation, to recall similar things while seeing an object, happens very often. For instance, when you see somebody like your father, you recall your father as well.

[2] Interesting discussions about differences between human and computer can be found in [6].

3 Practical Design and Implementation of GPLM

We studied GPLM as a black box in the previous section and we did not say anything about its internal structure. In this section we'll design the internal system with KNN algorithm and study the obstacles in the way of reaching our scope. It is necessary to introduce some parameters:

N: total number of saved instances
k: number of required instances for generalization
m: number of attributes which the output value depends on

In practical applications:

$$N >> m,k \quad , \quad m \approx k \tag{3}$$

A GPLM which uses KNN as an internal system is displayed in Fig. 2. At the beginning of the machine job a set of *N* training instances, the value of *k* and an array which determines the range of each attribute enter the machine. These training points will be stored in the database. When an unknown instance enters, the machine finds *k* nearest points to it. Its output value then will be estimated through a generalization beyond these found points and will be sent out as output. Besides each time a new training instance enters the machine it will be stored in the database.

By now, overall process in GPLM has been covered. But for practical implementation, there are some issues which are discussed in the following sections.

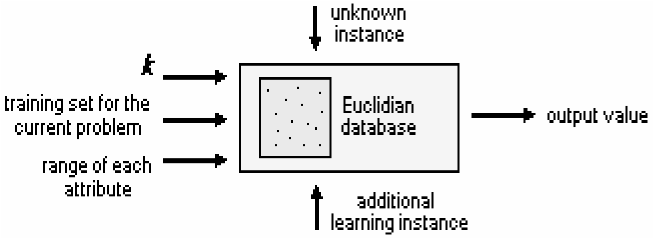


Fig. 2. Overall structure of GPLM, with KNN applied to it

3.1 Laziness

The way a GPLM works was described briefly, but implementing it in an optimal way is not that easy. In this structure, when the training instances enter, they are just saved, and then when we need to estimate an unknown instance we have to search among *N* points. This operation for great values of *N, m* is really costly and can not be applied. The operation of delaying process to the retrieval time is called *laziness.* In order to avoid this problem various indexing methods have been proposed, such as KD-Trees. An overview of these methods will show that finding *k* nearest neighbors exactly, for high-dimensional spaces (e.g. greater than 50), is really time-consuming and can not be applied. In order to overcome the problem, we should attend that the nature of learning is *approximation* according to the previous knowledge. So instead of finding the *k* nearest neighbors exactly, approximate *k* nearest neighbors can be found. For example, in KD-Trees the approximate nearest neighbor is found with the cost of $O(\log N)$, and then for finding the exact nearest neighbor an algorithm with the cost of $O(2^m)$ should be applied. Therefore eliminating the latter phase will substantially reduce the cost of query. With a little change to the previous indexing methods, we can find the approximate *k* nearest neighbors fast enough. Thus the *laziness* problem can be overcome by approximate KNN retrieval.

3.2 Curse of Dimensionality

Another practical issue in applying KNN algorithms is that the distance between points is calculated based on *all* attributes (i.e., on all axis in the Euclidean space containing the instances). To see the effect of this policy, consider applying KNN to a problem in which each instance is described by 20 attributes, but where only 2 of these attributes are effectively relevant to determining the output value. In this case, instances that have identical values for the 2 relevant attributes may nevertheless be distant from one another in the 20-dimensional space. As a result, the similarity metric used by KNN depending on all 20 attributes will be misleading. The distance between neighbors will be dominated by the large number of irrelevant attributes. This difficulty, which arises when many irrelevant attributes are present, is sometimes referred to as the *curse of dimensionality*. KNN algorithms are especially sensitive to this problem (see Fig. 3. a).

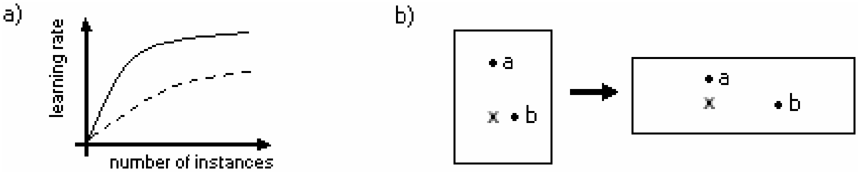


Fig. 3. a) Stippled curve relates to a problem which suffers from curse of dimensionality. **b)** Stretching and shortening axis can affect the value of Euclidian distance between instances. In the left space *b* is closer to *X*, but in the right one *a* is.

One interesting approach to overcoming this problem is to weight each attribute differently when calculating the distance between two points, or we can stretch or shorten the domain space for some axis (see Fig. 3. b). The attributes which are more relevant should be shortened in their axis and vice versa. The amount by which each axis should be stretched can be determined *automatically* using a cross-validation approach. To see how, first note that we wish to stretch (multiply) the jth axis by some factor z_j, where the values $z_1 \dots z_n$ are chosen to minimize the output value error of the KNN algorithm. Second, note that this error can be estimated using cross-validation. Hence, one algorithm is to select a random subset of the available initial training examples, then determine the values of $z_1 \dots z_n$ that lead to the minimum error in estimating the remaining examples. By repeating this process multiple times the estimate for these weighting factors can be made more accurate. This process of stretching the axis in order to optimize the performance of KNN provides a mechanism for suppressing the impact of irrelevant attributes. For more detailed solutions see [7], [8].

The cross-validation algorithm which just described briefly may need a great deal of time, but it is done when the machine is initialized and its output factors are saved. Thus there won't be any effect on the retrieval and the learning time.

3.3 Generalization from Nearest Neighbors

Another important issue in KNN algorithms is generalization from the retrieved neighbors. A few effective and efficient methods have been developed such as distance-weighted algorithm, locally-weighted regression and etc (see [1]). We are not going to talk about these algorithms or maybe a new one. Most of the algorithms are suitable for our purpose, but we should be careful of the algorithm time complexity, which may negate the efforts for reducing the time of retrieving the *k* nearest neighbors.

4 Empirical Results

We have examined the GPLM in different problems. Two of these examinations are presented in this section.

Consider a soccer player shooting a ball. We'd like to estimate *range* (*R*) of the shoot. Suppose that *R* depends on *velocity* (v_x, v_y, v_z) and *acceleration* (a_x, a_y, a_z). *R* can be determined by way of physics, therefore we supplied *training* and *test* sets in this way. The result is displayed in Fig. 4.

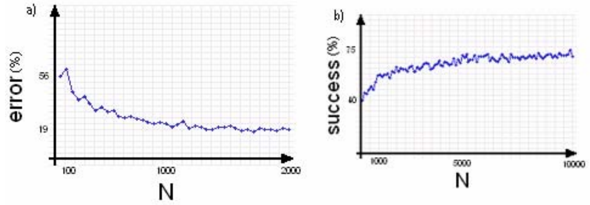


Fig. 4. *Success* and *error* for 6-dimensional shooting range problem, **a)** *error* percentage drops as *N* grows, **b)** *error* less than 20% is considered as *success*

Another practical example which is a challenging problem in RoboCupRescue Simulation League, is estimating the amount of water should be poured on fire to be extinguished in a cycle. It depends on *Area of the burning building* and *Ignition time*. Result is displayed in Fig. 5.

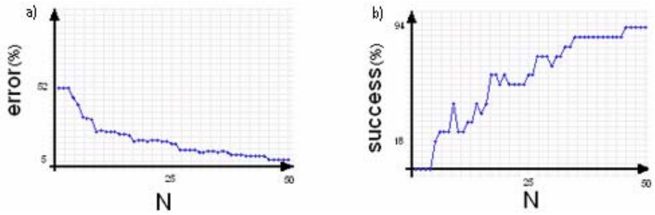


Fig. 5. *Success* and *error* for 2-dimensional extinguishing problem, **a)** *error* drops as *N* grows, **b)** *error* less than 20% is considered as *success*

5 Conclusions

In this paper, we proposed GPLM which is suitable for various learning problems. The features of a "General-Purpose Machine" described in the introduction. We get the GPLM idea from the brain functionality and proposed KNN for the internal system. We also discussed about how to optimize KNN to be efficiently used in GPLM. In the computer simulation, we applied GPLM to several learning problems, and the simulation results showed that we have achieved our scope of generality.

References

1. Mitchell, T. M.: *Machine Learning*, McGraw-Hill (1997)
2. Russell, S. J., Norvig, P.: *Artificial Intelligence A Modern Approach*, Prentice Hall, Englewood Cliffs, New Jersey 07632 (1995)
3. Albus, J. S.: *A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)*, J. Dyn. Syst. Meas. Control, Trans. ASME 97, 220-227 (1975)
4. Gaede, V., Günther, O.: *Multidimensional Access Methods*, ACM Computing Surveys, Vol.30, No. 2 (1998)
5. Samet, H.: *The Design and Analysis of Spatial Data Structures*, Addison-Wesley (1989)
6. Anderson, H. C.: *Why Artificial Intelligence isn't (Yet)*, AI Expert Magazine, July (1987)
7. Moore, A. W., Lee, M. S.: *Efficient algorithms for minimizing cross validation error*. Proceedings of the 11th International Conference on Machine Learning. San Francisco (1994)
8. Berchtold, S., Boehm, C., Kriegel H. P.: *The Pyramid-Technique: Towards Breaking the Curse of Dimensionality*, Proc. ACM SIGMOD (1998)

Improvement of Color Recognition Using Colored Objects

T. Kikuchi[1], K. Umeda[1], R. Ueda[2], Y. Jitsukawa[2], H. Osumi[1], and T. Arai[2]

[1] Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo, Japan
tkikuchi@sensor.mech.chuo-u.ac.jp, {umeda, osumi}@mech.chuo-u.ac.jp
http://www.mech.chuo-u.ac.jp/umedalab/index_e.html
[2] The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
{ueda, jitsukawa, arai-tamio}@prince.pe.u-tokyo.ac.jp
http://www.arai.pe.u-tokyo.ac.jp/

Abstract. Recognition of colored objects is important and practical for robot vision. This paper proposes a color recognition method which is robust to illumination change. A color space of Cb/Y and Cr/Y is introduced where Cb and Cr are color difference and Y is intensity of YCbCr color space. This color space is not affected by the change of the brightness of illumination. And a method to update clusters of color table is proposed. The method can cope with the change of the color of illumination. Experiments show that the proposed method can recognize color more robustly for illumination change. RoboCup four legged robot league is chosen as the research platform.

1 Introduction

Images are widely used by a robot to recognize its environment. Many objects in the environment where a robot works have some colors and they are useful cues for recognizing the environment or objects. Color information observed in images changes according to illumination condition and thus robust object recognition with color information is not easy. Many studies have dealt with the estimation of illumination or object color [1][2].

To recognize color fast and simply, a lookup table that relates observed pixels and color names is effective. Jüngel et al. proposed a method to construct a table dynamically by scanning an image for the RoboCup field [3]. Mayer et al. assumed a known environment to find a target object with a given color table and modify the table [4]. These methods are not applicable to general environment because of the restriction of the environment for constructing their recognition method. Hanek et al. utilized shape information instead of color information for object recognition in RoboCup middle size league [5]. Their method requires much calculation cost. Dahm et al. proposed a new color space in which color clustering becomes simpler [6].

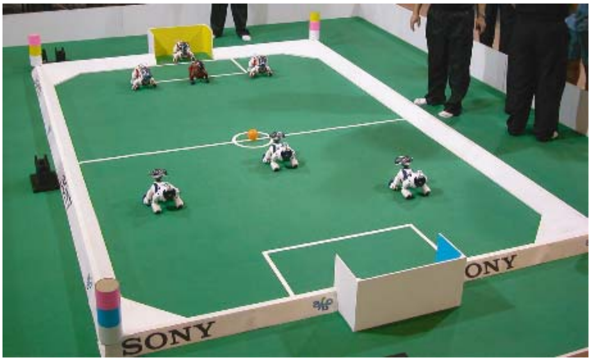
We propose a new method to recognize colored objects robustly. A color space which is simple and robust for illumination change is introduced. A color table is updated using statistical information of original color cluster and observed

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 537–544, 2006.

pixels. The effectiveness of the proposed methods are verified experimentally. RoboCup four legged robot league is chosen as the research platform.

2 Camera and Environment

Sony ERS-7 (AIBO) is the common robot for RoboCup four legged robot league. A CMOS camera is mounted at its nose and used for image acquisition. The image size is small, 208 × 160 pixels . Its color space is YCbCr. A 4200 mm × 2700 mm field is used for the match. Objects in the field are a ball, goals, robots, landmarks and the field itself as shown in Fig. 1. They are colored to help autonomous behaviors of the robots. It is obvious that color reconition is fundamental for tasks as self-localization, measurement of ball position, etc.



Ball	Orange
Goal	Sky blue, Yellow
Field	Green, White
Robot	Red, Dark blue
Landmark	Pink-Sky blue, Pink-Yellow

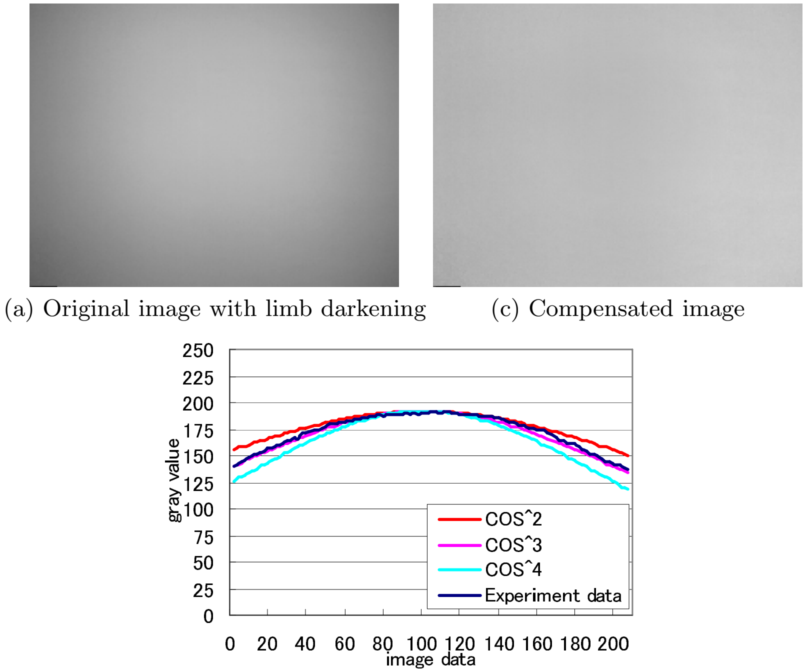
Fig. 1. Field of RoboCup four legged robot league

3 Camera Calibration

Proper camera calibration is essential for measurement with images. Camera parameters to calibrate are, intrinsic parameters such as focal length [7], distortion of the lens, limb darkening etc.

We took advantage of the calibration function of MVTec Halcon image processing software [8] to obtain intrinsic and distortion parameters. The distortion parameter κ is -4683m^{-2}, which is smaller than we expected, and we found that correction of distortion is not mandatory for image processing with the camera.

Image intensity becomes darker at the limb, and it is called limb darkening (or vignetting). Theoretically, it is proportional to the fourth power of cosine of irradiation angle. Fig. 2(a) shows the image of a white drawing paper parallel to the camera. The limb darkening is obviously observed because the lens is wide-angle (56.9deg × 45.2deg). Fig. 2(b) shows the intensity values for a row of image. We can see the limb darkening of the ERS-7's camera is proportional to the third power of cosine, not fourth. Fig. 2(c) shows the compensation by



(a) Original image with limb darkening

(c) Compensated image

(b) Limb darkening for a row of image

Fig. 2. Limb darkening of ERS-7 CMOS camera and its compensation

the third power of cosine. Uniform intensity distribution is obtained. However, compensation of the limb darkening is not mandatory either, for we utilize a color space robust to the brightness change.

4 Realization of Robust Color Table

4.1 Color Table

In real-time image processing, object recognition with a color image is often realized based on color labeling. Color labeling here means to relate each observed pixel to an object's color name such as the ball color. It is realized by thresholding in a color space or by using a lookup table (color table) that relates each pixel to a color name. We take advantage of the color table for color labeling. The table-based method has the characteristics of detailed color labeling with low calculation cost.

4.2 Color Space Robust to Brightness Change

The color space of the ERS-7's camera is YCbCr as mentioned above. Y, Cb and Cr represent intensity and color difference of blue and red respectively. For

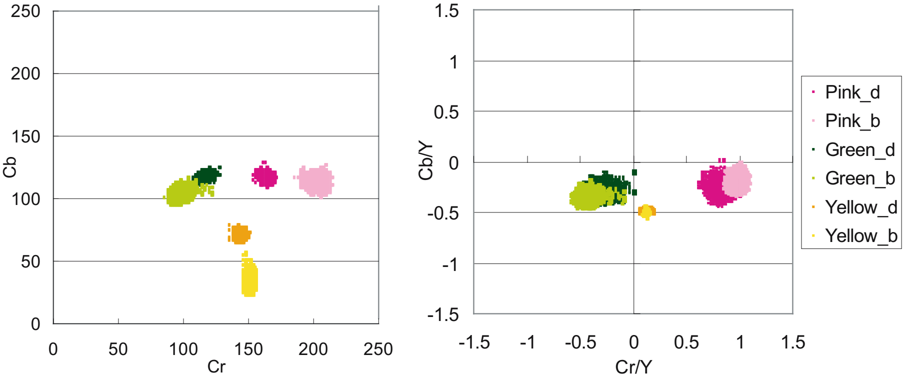


Fig. 3. Color disribution of Cb-Cr space and Cb/Y-Cr/Y space

color recognition, Cb and Cr are important. In principle, Cb and Cr change in proportion to the Y, i.e. Cb and Cr are affected by brightness. Therefore, we propose a color space of the color differences divided by the intensity; Cb/Y-Cr/Y space. Color table is constructed and recognition is performed in the space.

This space is expected to be robust to the brightness change. Fig. 3 shows an example. A Kodak color chart was captured by the camera, and each pixel value is plotted in Cb-Cr space and Cb/Y-Cr/Y space. Pink, yellow and green data of the chart were tested. In Fig. 3, data with suffix “b” and “d” were captured at bright (1200 lx) and dark (800 lx) environment respectively. We can see that the Cb/Y-Cr/Y space is not affected by the brightness change. Additionally, we can see distributions for the dark image tend to become larger. It is because the effect of noise of the intensity value is magnified by the division.

4.3 Update of Color Table

When illumination changes, observed color information also changes. We propose a method to cope with the color change to some extent. The basic idea is to update the color table according to the illumination change.

Fig. 4 outlines the update process. Suppose a reference color table was acquired in advance and average vector and covariance matrix of each cluster of the color table is calculated as shown in Fig. 4(a)DThe ellipses represents the points with the same Mahalanobis distance.

The update process of the color table is as follows.

1. Capture an image under an illumination condition and plot each pixels of the image on Cb/Y-Cr/Y space.
2. Make a cluster with points which have small Mahalanobis distance to the cluster of the color table as shown in Fig. 4(b). The square of Mahalanobis distance obeys the χ^2 distribution with 2 degrees of freedom and statistical evaluation is possible.

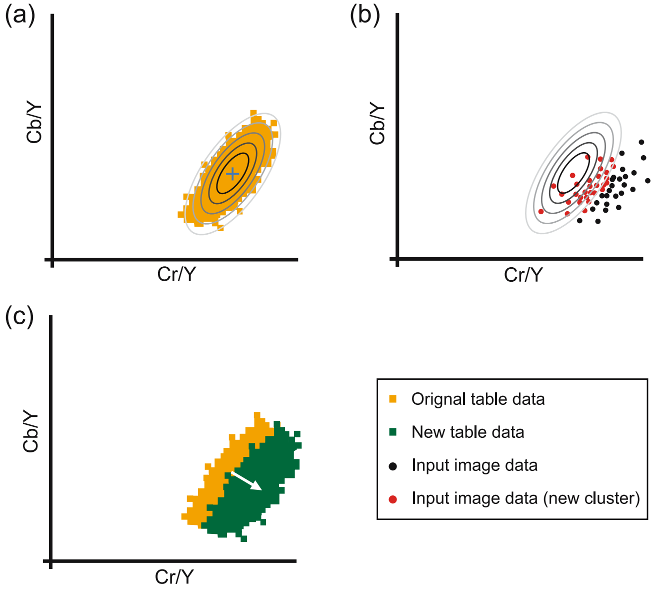


Fig. 4. Update of color recognition table

3. Calculate the offset between the cluster made in 2 and the cluster of the color table.
4. Move the the cluster of the color table by the offset calculated in 3 and make a new cluster of the color table (Fig. 4 (c)).

By applying the above procedure to each cluster of the color table, an updated color table is constructed.

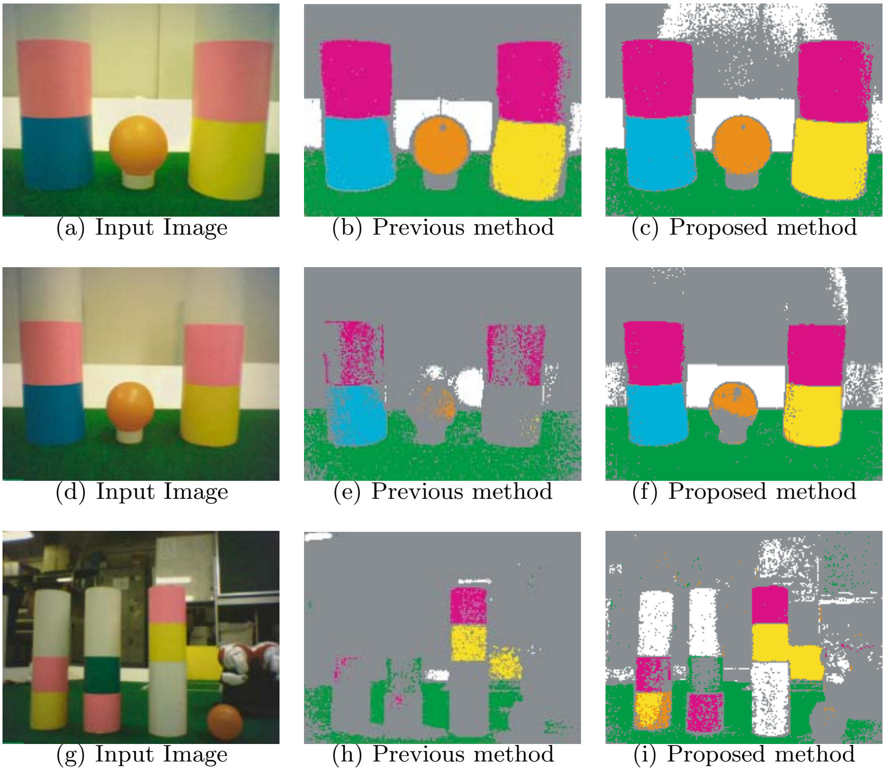
5 Experiments

The lighting was by commercially available fluorecent lights. They were adjusted to 1200 and 800 lx.

5.1 Color Recognition Experiments

First we constructed the reference color table in advance under the 1200 lx illumination. Three images were used for this. For comparison, we prepared a traditional color table in YCbCr space. A color is assigned to each point in the YCbCr space.

Fig. 5 shows the results to recognize the color of objects on the field. Fig. 5(a) is the image used to construct the reference table. Fig. 5(b) by the previous method and Fig. 5(c) by the proposed method both show the sufficient color recognition. Fig. 5(d) shows the image under 800 lx and additional lighting of a incandescent lamp. Brightness itself was changed and color was changed because of the lamp with red-shifted spectrum. In this condition, the previous method failed to detect



(a) Input Image (b) Previous method (c) Proposed method

(d) Input Image (e) Previous method (f) Proposed method

(g) Input Image (h) Previous method (i) Proposed method

Fig. 5. Input image and color recognized image by previous and proposed method

the landmarks (Fig. 5(e)) and on the contrary, the proposed method succeeded in precise color recognition (Fig. 5(f)) . Fig. 5(g) shows another example with the same lighting condition as Fig. 5(d). In this case, the previous method failed again (Fig. 5(h)). The proposed method produces much better results (Fig. 5(i)), but there exist some recognition failures for orange and yellow objects. This is because the cluster for orange was moved to the yellow pixels and consequently a mixtured wrong cluster of the table was produced at the update.

5.2 Ball Recognition and Measurent of Distance

We conducted another experiment of recognizing the ball and measuring distance to it using the produced color table by the proposed color recognition methods. With its head shaking, the robot tried to recognize the ball and measure its distance. The ball was set at 300, 500, 1000 and 1500 mm from the robot. Twenty trials were performed for each distance. The distance to the ball was measured by two methods: using the number of pixels (when the ball is far) and using the geometrical configuration between the robot and the ball (when the ball is near). In both methods, the more precise the color recognition is, the more accurate the distance measurement becomes.

Table 1. Ball recognition (number for 20 trials)

Light	Method	1500	1000	500	300
		Recognition	Recognition	Recognition	Recognition
1200 lx	Previous method	20	20	20	20
	Proposed method	20	20	20	20
1200 lx + blue light	Previous method	20	20	20	20
	Proposed method	20	19	18	19
800 lx	Previous method	0	0	0	0
	Proposed method	19	20	18	19
800 lx + blue light	Previous method	0	0	0	0
	Proposed method	20	18	19	20

Table 2. Measurement of ball distance (unit: mm)

Light	Method	1500		300	
		Average distance	Standard deviation	Average distance	Standard deviation
1200 lx	Previous method	1873.9	57.5	366.7	13.8
	Proposed method	1526.4	38.4	310.8	8.7
1200 lx + blue light	Previous method	1920.9	35.4	385.9	5.5
	Proposed method	1535.6	23.3	325.8	6.4
800 lx	Previous method	×	×	×	×
	Proposed method	1752.5	45.6	316.4	4.7
800 lx blue light	Previous method	×	×	×	×
	Proposed method	1641.1	27.3	364.0	7.5

The illumination condition was 1200 or 800 lx by the fluorecent lights with or without a 100W blue incandescent lamp, i.e. four conditions. Table 1 and 2 show the number of successful experiments and results of distance measurement respectively.

From Table 1, we can observe the following. The previous method only succeeded when illumination was 1200 lx and every trial failed at 800 lx. It is because the pixel values of the ball went out of the range assigned in the reference table. On the contrary, the proposed method almost succeeded in every condition regardless of illumination. There are a few failures, in which an object other than a ball was recognized wrongly. This is because the cluster of the table invaded other color region at the update.

As for the distance measurement in Table 2, the proposed method produced more precise results in every condition. In the previous method, the number of recognized pixels decreases and thus the distance errors increase. The proposed method can recognize the pixels robustly, and consequently distance measurement becomes precise.

6 Conclusion

In this study, we have proposed a color recognition method which is robust to illumination change. A color space of Cb/Y and Cr/Y was introduced where Cb and Cr are color difference and Y is intensity of YCbCr color space. The color space is not affected by the brightness change. And a method to update clusters of color table was proposed. This method can cope with the change of the color of illumination. Experiments showed that the proposed method can recognize color more robustly for illumination change.

One problem is that the shift of the cluster of the table is sometimes not precise and it causes the failure of color recognition. Future work is to cope with this problem. And illumination model should be considered which includes mirror reflectance.

References

1. G. D. Finlayson, B. V. Funt. Color constancy under varying illumination. Proc. of IEEE Int. Conf. on Computer Visoin, pp.720 -725, 1995.
2. M. Sridharan and P. Stone: Toward Illumination Invariance in the Legged League. Proc. of RoboCup2004 Symposium, (CD-ROM), 2004.
3. M. Jüngel, J. Hoffmann, M. Lotzsch. A Real-Time Auto-Adjusting Vision System for Robotic Soccer. Proc. of RoboCup2003 Symposium, pp.214-225, 2003.
4. G. Mayer, H. Utz, and G. Kraetzchmar. Towards Autonomous Vision Self-Calibration for Soccer Robot. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)2002, pp.214-219, 2002.
5. R. Hanek, T. Schmitt, S. Buck and M. Beetz. Fast Image-based Object Localization in Natural Scenes. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems(IROS)2002, pp.116-122, 2002.
6. I. Dahm, S. Deutsch, M. Hebbel, and A. Osterhues. Knowledge-based Autonomous Dynamic Colour Calibration. Proc. of RoboCup2003 Symposium, pp.1-12, 2003.
7. O. Faugeras, Three-Dimensional Computer Vision, MIT Press, 1993.
8. http://www.mvtec.com/

Improving Percept Reliability in the Sony Four-Legged Robot League*

Walter Nisticò[1] and Thomas Röfer[2]

[1] Institute for Robot Research (IRF), Universität Dortmund
walter.nistico@udo.edu
[2] Center for Computing Technology (TZI), FB 3, Universität Bremen
roefer@tzi.de

Abstract. This paper presents selected methods used by the vision system of the GermanTeam, the World Champion in the Sony Four-Legged League in 2004. Color table generalization is introduced as a means to achieve a larger independence of the lighting situation. Camera calibration is necessary to deal with the weaknesses of the new platform used in the league, the Sony Aibo ERS-7. Since the robot camera uses a rolling shutter, motion compensation is required to improve the information extracted from the camera images.

1 Introduction

The perceptive layer in a robotic architecture is the primary source of information concerning the surrounding environment. In case of the RoboCup 4-legged league, due to the lack of range sensors (e.g. laser scanners, sonars), the robot only has the camera to rely upon for navigation and object detection, and in order to be able to use it to measure distances, the camera position in a robot-centric reference system has to be dynamically estimated from leg and neck joint angle measurements, having to deal with noise in these measures. The robot interacts with a dynamic and competitive environment, in which it has to quickly react to changing situations, facing real-time constraints; image processing tasks are generally computationally expensive, as several operations have to be performed on a pixel level, thus with an order of magnitude of $10^5 - 10^6$ per camera frame. The need to visually track fast moving objects (i.e. the ball) in the observed domain, further complicated by the limited camera field of view, makes it necessary for the vision system to be able to keep up with the highest possible frame rate that the camera can sustain: in the case of the robot Sony Aibo ERS-7, 30 fps. As a result, image segmentation is still mainly achieved through static color classification (see [3]).

2 Color Table Generalization

Robustness of a vision system to lighting variations is a key problem in RoboCup, both as a long term goal to mimic the adaptive capabilities of organic systems,

* The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 545–552, 2006.

as well as a short term need to deal with unforeseen situations which can arise at the competitions, such as additional shadows on the field as a result of the participation of a packed audience. While several remarkable attempts have been made in order to achieve an image processor that doesn't require manual calibration (see [5], [12]), at the moment traditional systems are more efficient for competitions such as the RoboCup. Our goal was to improve a manually created color table, to extend its use to lighting situations which weren't present in the samples used during the calibration process, or to resolve ambiguities along the boundaries among close color regions. Thereto, we have developed a color table generalization technique which uses an exponential influence model similar to the approach described in [7], but in contrast to it, is not used to perform a semi-automated calibration from a set of samples. This new approach is based on the assumption of spatial locality of the color classes in the color space, so that instead of the frequency of a set of color samples, it's the spatial frequency of neighbors in the source color map to determine the final color class of a given point, following the idea that otherwise, a small variation in lighting conditions, producing a spatial shift in the mapping, would easily result in classification errors. Thus, a color table is processed in the following way:

- Each point assigned to a color class irradiates its influence to the whole color space, with an influence factor exponentially decreasing with the distance:

$$I_i(p_1, p_2) = \begin{cases} \lambda^{|p_1 - p_2|} & i = c(p_2) \\ 0 & \forall i \neq c(p_2) \end{cases} \tag{1}$$

  where p_1, p_2 are two arbitrary points in the color map, $\lambda < 1$ is the exponential base, $I_i(p_1, p_2)$ is the influence of p_2 on the (new) color class $i \in \{red, orange, yellow, \cdots\}$ of p_1, and $c(p_2)$ is the color class of p_2,
- Manhattan distance is used (instead of Euclidean) to speed up the influence calculation ($\mathrm{O}(n^2)$, where n is the number of elements of the color table):

$$|p_1 - p_2|_{manhattan} = |p_{1y} - p_{2y}| + |p_{1u} - p_{2u}| + |p_{1v} - p_{2v}| \tag{2}$$

- For each point in the new color table, the total influence for each color class is computed:

$$I_i(p_0) = B_i \cdot \sum_{p \neq p_0} I_i(p_0, p) \tag{3}$$

  where $B_i \in (0..1]$ is a bias factor which can be used to favor the expansion of one color class over another
- The color class that has the highest influence for a point is chosen, if:

$$\frac{max(I_i)}{I_{bk} + \sum_i I_i} > \tau \tag{4}$$

  where τ is a confidence threshold, I_{bk} is a constant value assigned to the influence of the background (noColor) to prevent an unbounded growth of the colored regions to the empty areas, and i again represents the color class.

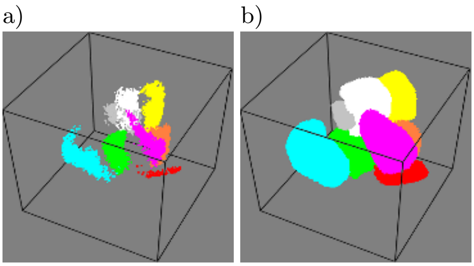


Fig. 1. Effects of the exponential generalization on a color table: (a) original, (b) optimized

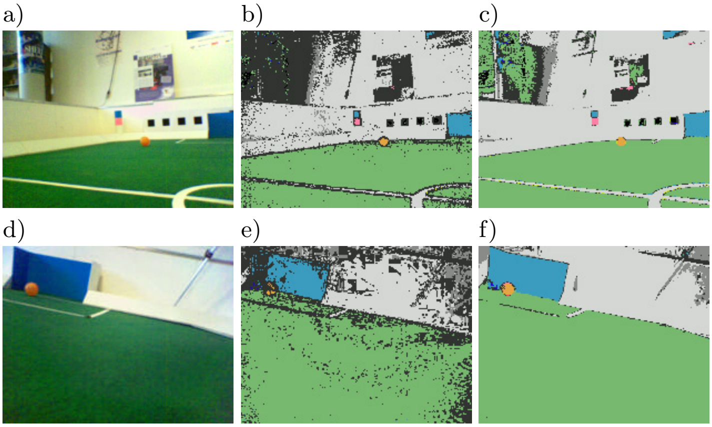


Fig. 2. Exponential generalization. (a) and (d) represent images taken from the same field, but in (d) the amount of sunlight has increased: notice the white walls appearing bluish. (b) and (e) are the result of the classification from the original color table, calibrated for the conditions found in (a); notice that (e) is not satisfactory, as the ball is hard to detect and the goal appears largely incomplete. (c) and (f) are classified using the generalized table, showing that it can gracefully accommodate to the new lighting conditions (f).

The parameters λ, τ, B_i, I_{bk} control the effects of the generalization process, and we have implemented 3 different settings: one for conservative generalization, one for aggressive expansion, one for increasing the minimum distance among neighboring regions. The time required to apply this algorithm, on a 2^{18} elements table, is $\approx 4-7$ minutes on a 2.66GHz Pentium4 processor, while for a table of 2^{16} elements, this figure goes down to only 20-30 seconds.

3 Camera Calibration

With the introduction of the ERS-7 as a platform for the 4-Legged League, an analysis of the camera of the new robot was required to adapt to the new specifications. While the resolution of the new camera is ≈ 31% higher compared to the previous one, preliminary tests revealed some specific issues which weren't present in the old model. First, the light sensitivity is lower, making necessary the use of the highest gain setting, at the expense of amplifying the noise as well; such a problem has been addressed in [10]. Second, images are affected by a vignetting effect (radiometric distortion), which makes the peripheral regions appear darker and dyed in blue.

3.1 Geometric Camera Model

In the previous vision system, the horizontal and vertical opening angles of the camera were used as the basis for all the measurement calculations, following the classical "pinhole" model; however for the new system we decided to use a more complete model taking into account the geometrical distortions of the images due to lens effects, called the *DLT model* (see [1].) This model includes the lack of orthogonality between the image axes s_θ, the difference in their scale (s_x, s_y), and the shift of the projection of the real optical center (principal point) (u_0, v_0) from the center of the image (together called "intrinsic parameters") and the rotation and translation matrices of the camera reference system relative to the robot (R, T, "extrinsic parameters"). In addition to this, we have also decided to evaluate an augmented model including radial and tangential non-linear distortions with polynomial approximations, according to [4] and [8]. In order to estimate the parameters of the aforementioned models for our cameras, we used a Matlab toolbox from Jean-Yves Bouguet (see [2]). The results showed that the coefficients (s_x, s_y, s_θ), are not needed, as the difference in the axis scales is below the measurement error, and so is the axis skew coefficient; the shift between the principal point (u_0, v_0) and the center of the image is moderate and dependent from robot to robot, so we have used an average computed from images taken from 5 different robots. As far as the non-linear distortion is concerned, the results calculated with Bouguet's toolbox showed that on the ERS-7 this kind of error has a moderate entity (maximum displacement ≈ 3 pixel), and since in our preliminary tests, look-up table based correction had an impact of ≈ 3ms on the running time of the image processor, we decided not to correct it.

3.2 Radiometric Camera Model

As the object recognition is still mostly based on color classification, the blue cast on the corners of the images captured by the ERS-7's camera is a serious hindrance in these areas. Vignetting is a radial drop of image brightness caused by partial obstruction of light from the object space to image space, and is usually dependent on the lens aperture size ([9], [6]), however, in this case the strong chromatic alteration seems difficult to explain merely in terms of optics, and we suspect it could be partially due to digital effects. To be able to observe the characteristics of this vignetting effect, we captured images of uniformly colored objects from the robot's camera, lit by a diffuse light source (in order to minimize the effects of shadows and reflections). As can be seen in Figure 3,

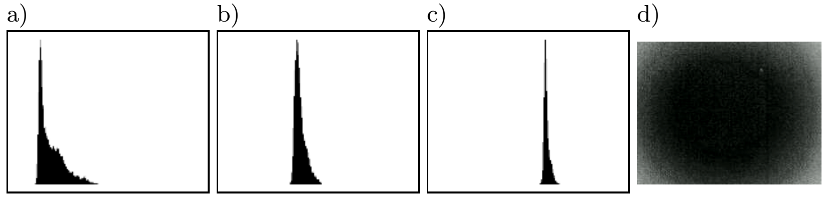


Fig. 3. (a,b,c) Histograms of the U color band for uniformly colored images: yellow (a), white (b) and skyblue (c). In case of little or no vignetting effect, each histogram should exhibit a narrow distribution around the mode, like in (c). (d) Brightness distribution of the U color band for a uniformly colored yellow image.

the radiometric distortion d_i for a given spectrum i of a reference color I is dependent on its actual value (*brightness component*):

$$d_i(I) \propto \lambda_i(I_i) \tag{5}$$

Moreover, the chromatic distortion that applies on a certain pixel (x, y) appears to be also dependent on its distance from a certain point (cf. Fig. 3(d)), center of distortion (u_d, v_d), which lies approximately close to the optical center of the image, the principal point; so, let $r = \sqrt{(x - u_d)^2 + (y - v_d)^2}$, then (*radial component*):

$$d_i(I(x, y)) \propto \rho_i(r(x, y)) \tag{6}$$

Putting it all together:

$$d_i(I(x, y)) \propto \rho_i\left(r(x, y)\right) \cdot \lambda_i\left(I_i(x, y)\right) \tag{7}$$

Now, we derive $\rho_i, \lambda_i, \forall i \in \{Y, U, V\}$ from a set of sample pictures; since both sets of functions are non-linear, we decided to use a polynomial approximation, whose coefficients can be estimated using least-square optimization techniques:

$$\begin{aligned} \rho_i\left(r\right) &= \sum_{j=0}^{n} \varrho_{i,j} \cdot r^j \\ \lambda_i\left(I_i\right) &= \sum_{j=0}^{m} l_{i,j} \cdot I_i^j \end{aligned} \tag{8}$$

In order to do so, we have to create a log file containing reference pictures which should represent different points belonging to the functions that we want to estimate, hence we chose to use uniform yellow, blue, white and green images taken under different lighting conditions and intensities. Then, the log file is processed in the following steps:

- For each image, a reference value is estimated for the 3 spectra Y, U, V, as the modal value of the corresponding histogram ($numOfBins$ = *color Levels* = 256).
- The reference values are clustered into classes, such that series of images representing the same object under the same lighting condition have a single

reference value; this is achieved using a first order linear Kalman filter to track the current reference values for the 3 image spectra, and a new class is generated when:

$$\exists j \in \{Y, U, V\} : \left| r^m_{j,k} - r^p_{j,k-1} \right| > \vartheta \tag{9}$$

where $r^m_{j,k}$ is the reference (for spectrum j) measured at frame k, $r^p_{j,k-1}$ is the reference predicted by the Kalman filter at frame $k-1$, and $\vartheta = 40$ is a confidence threshold.

- Simulated annealing ([11]) is used to derive the coefficients (u_d, v_d), and $\varrho_{i,j}$, $l_{i,j} \forall i \in \{Y, U, V\}$ (in a separate process for each color band).
- In each step, the coefficients $\varrho_{i,j}, l_{i,j}$ are mutated by the addition of zero mean gaussian noise, the variance is dependent on the order of the coefficients, such that high order coefficients have increasingly smaller variances than low order ones.
- The mutated coefficients are used to correct the image, as:

$$I'_i(x, y) = I_i(x, y) - \rho_i\left(r(x, y)\right) \cdot \lambda_i\left(I_i(x, y)\right) \tag{10}$$

- For each image $I_{i,k}$ in the log file (i is the color band, k the frame number), given its reference value previously estimated $r_{i,k}$, the current "energy" E for the annealing process is calculated as:

$$E_i = \sum_{(x,y)} \left(I'_{i,k}(x, y) - r_{i,k}\right)^2 \tag{11}$$

- The "temperature" T of the annealing is lowered using a linear law, in a number of steps which is given as a parameter to the algorithm to control the amount of time spent in the optimization process; the starting temperature is normalized relative to the initial energy.
- The correction function learned off-line is stored in a look-up table for a fast execution on the robot.

Figure 4 shows some examples of corrections obtained after running the algorithm on a log file composed of 8 image classes (representing different colors at different lighting conditions) of 29 images each, decreasing the temperature to

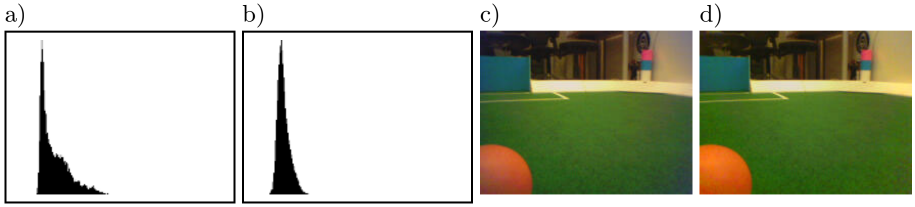


Fig. 4. Color correction in practice: histograms of the U color band of a uniformly yellow colored image before correction (a), and after (b); actual image taken from a game situation, before correction (c) and after (d).

0 in 100 steps, for a total optimization time of 7 minutes (Pentium4 2.66GHz). In case of the image spectra which exhibit the highest distortion (Y, U), the variance after the calibration is reduced by a factor of 10.

4 Motion Compensation

The camera images are read sequentially from a CMOS chip using a *rolling shutter*. This has an impact on the images if the camera is moved while an image is taken, because each scan line is captured at a different time instant. For instance in Figure 5 it can be seen, that the flag is slanted in different directions depending on whether the head is turning left or right. In experiments it was recognized that the timestamp attached to the images by the operating system of the Aibo corresponds to the time when the lowest row of the image was taken. Therefore, features in the upper part of the image were recorded significantly earlier. It is assumed that the first image row is recorded shortly after taking the previous image was finished, i. e. 10% of the interval between two images, so 90% of the overall time is spent to take the images. For the ERS-7, this means that the first row of an image is recorded 30 ms earlier than the last row. If the head, e. g., is rotating with a speed of 180°/s, this results in an error of 5.4° for bearings on objects close to the upper image border. Therefore, the bearings have to be corrected. Since this is a quite time-consuming operation, it is not performed as a preprocessing step for image processing. Instead, the compensation is performed on the level of percepts, i. e. recognized flags, goals, edge points, and the ball. The compensation is done by interpolating between the current and the previous camera positions depending on the y image coordinate of the percepts.

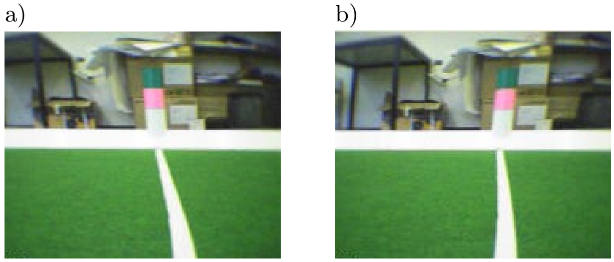


Fig. 5. Images taken while the head is quickly turning. a) Left. b) Right.

5 Results

The algorithms described here have been tested and used as part of the vision system of the GermanTeam which became World Champion in the Four-Legged League at the RoboCup 2004 competitions. The average running time of the whole image processor was 9±2 ms, and the robot was able to keep up with the maximum camera frame rate under all circumstances, i.e. 30 fps. Throughout the competitions, our vision system proved to be robust and accurate, and our robots' localization was widely acclaimed as the best of the league; these techniques have also been used on the vision system of Microsoft Hellhounds, a member of the GermanTeam, achieving the second place in the Variable Lighting Technical Challenge at the Japan Open 2004 competitions.

Acknowledgments

The program code used was developed by the GermanTeam, a joint effort of the Humboldt-Universität zu Berlin, Universität Bremen, Universität Dortmund, and Technische Universität Darmstadt. http://www.germanteam.org/

References

1. H. Bakstein. A complete dlt-based camera calibration, including a virtual 3d calibration object. Master's thesis, Charles University, Prague, 1999.
2. J.-Y. Bouguet. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.
3. J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061–2066, 2000.
4. J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 1106–1112, 1997.
5. M. Jüngel. Using layered color precision for a self-calibrating vision system. In *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2005.
6. S. B. Kang and R. S. Weiss. Can we calibrate a camera using an image of a flat, textureless lambertian surface?, 2000.
7. S. Lenser, J. Bruce, and M. Veloso. Vision - the lower levels. Carnegie Mellon University Lecture Notes, October 2003. http://www-2.cs.cmu.edu/ robosoccer/ cmrobobits/lectures/vision-low-level-lec/vision.pdf.
8. R. Mohr and B. Triggs. Projective geometry for image analysis, 1996.
9. H. Nanda and R. Cutler. Practical calibrations for a real-time digital omnidirectional camera. Technical report, CVPR 2001 Technical Sketch, 2001.
10. W. Nistico, U. Schwiegelshohn, M. Hebbel, and I. Dahm. Real-time structure preserving image noise reduction for computer vision on embedded platforms. In *Proceedings of the International Symposium on Artificial Life and Robotics, AROB 10th*, 2005.
11. S. Russel and P. Norvig. *Artificial Intelligence, a Modern Approach.* Prentice Hall, 1995.
12. D. Schulz and D. Fox. Bayesian color estimation for adaptive vision-based robot localization. In *Proceedings of IROS*, 2004.

Inferring 3D Body Pose from Uncalibrated Video

Xian-Jie Qiu[1,2], Zhao-Qi Wang[1], Shi-Hong Xia[1], and Yong-Chao Sun[1,3]

[1] Institute of Computing Technology, The Chinese Academy of Sciences,
Beijing 100080, China
[2] Graduate School, The Chinese Academy of Sciences, Beijing 100039, China
[3] Institute of Computing Technology, The Chinese Academy of Sciences,
Shenyang 110016, China
{qxj, zqwang, xsh, ycsun}@ict.ac.cn

Abstract. Recovery of 3D body pose is a fundamental problem for human motion analysis in many applications such as motion capture, vision interface, visual surveillance, and gesture recognition. In this paper, we present a new image-based approach to infer 3D human structure parameters from uncalibrated video. The estimation is example based. First, we acquire a special motion database through an off-line motion capture process. Second, given uncalibrated motion video, we abstract the extrinsic parameters and then silhouettes database associated with 3D poses is built by projecting each data of the 3D motion database into 2D plane with the extrinsic parameters. Next, with the image silhouettes abstracted from video, the unknown structure parameters are inferred by performs a similarity search in the database of silhouettes using approach based on shape matching. That is, the 3D structure parameters whose 2D projective silhouette is the most similar to the 2D image silhouette are took as the 3D reconstruction structure. We use trampoline sport motion, an example of complex human motion, to demonstrate the effectiveness of our approach.

1 Introduction

Recovery of 3D body pose is a fundamental problem for human motion analysis in many applications such as motion capture, vision interface, visual surveillance, and gesture recognition. Human body is an articulated object that moves through the 3D world. This motion is constrained by 3D body kinematics and dynamics as well as the dynamics of the activity being performed. Such constraints are explicitly exploited to recover the body configuration and motion in model-based approaches, such as[1, 2, 3], through explicitly specifying articulated models of the body parts, joint angles and their kinematics (or dynamics) as well as models for camera geometry and image formation. Recovering body configuration in these approaches involves searching high dimensional spaces (body configuration and geometric transformation) which are typically formulated deterministically as a nonlinear optimization problem, e.g. [2], or probabilistically as a maximum likelihood problem, e.g. [3]. In this paper we introduce a novel image-based framework for inferring 3D body pose from silhouettes using a single monocular uncalibrated camera.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 553 – 560, 2006.

Compared with other methods proposed previously, the strength of our approach lies in the combination of shape matching approach based on moment invariants and 3D human pose database together for the inference of 3D structure from complex motion video. On the other hand, our approach can process uncalibrated video since camera calibration of intrinsic parameters is unnecessary.

2 Related Work

Human pose estimation from images is an active area of computer vision research with many potential applications ranging from computer interfaces to motion capture for character animation, biometrics or intelligent surveillance. In the last decade there have been extensive researches in human motion analysis.

Inferring 3D pose from silhouettes can be achieved by learning mapping functions from the visual input to the pose space. However, learning such mapping between high dimensional spaces from examples is an ill posed problem. Therefore certain constraints should be exploited. In [4], the problem was constrained using nonlinear manifold learning, where the pose is inferred by mapping sequences of the input to paths of the learned manifold. In [5], 3D structure parameters are inferred from multi-view using a probabilistic model of multi-view silhouettes. Inferring pose can also be posed as a nearest neighbors search problem where the input is matched to a database of exemplars with known 3D pose. In [7] pose is recovered by matching the shape of the silhouette using shape context. Another promising approach, called *model based* [9, 10, 11, 12], relies on a 3D articulated volumetric model of the human body to constrain the localization process in one or several images. In this situation, the goal in human pose estimation applications is to estimate the model's articulation and possibly structural parameters such that the projection of the 3D geometrical model closely fits a human in one or several images.

The approach we use in this paper to infer 3D structure can be posed as a mapping problem through shape similarity between the projection of the 3D geometrical model and the input of image silhouettes based on the 3D pose database.

3 Framework

To infer structure parameters from uncalibrated video, this paper introduces a novel image-based framework (Figure 1). The estimation is example based. At first, motion database in special sport motion, such as trampoline sport, is acquired through an off-line motion capture process and a 3D human pose motion database is built (Section 4.1). Second, given motion video, we abstract the viewpoint (extrinsic parameters) automatically (Section 4.2) and build a 2D silhouettes database associated 3D poses by projecting 3D pose into 2D plane (Section 4.3). At last, given the 2D silhouettes abstracted from video, we infer the 3D structure parameters through performs a similarity search in the silhouettes database associated 3D poses through a approach based on moments (Section 4.4). We use trampoline sport motion, an example of complex human motion, to demonstrate the effectiveness of our approach (Section 5). The overall system diagram can be seen in Fig.1.

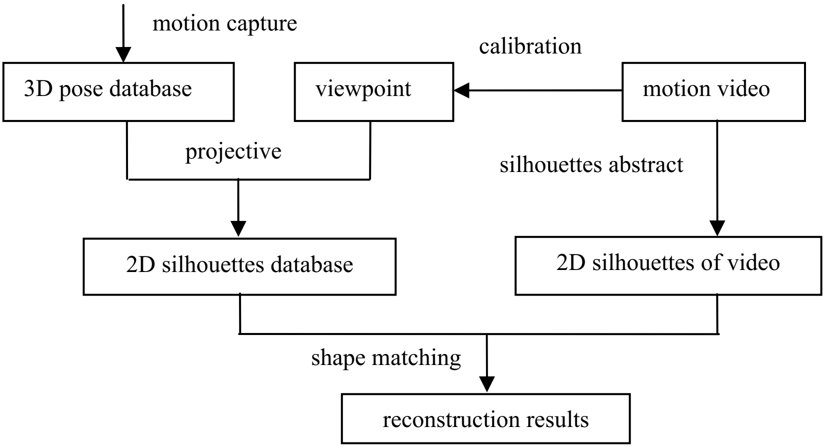


Fig. 1. The overall system diagram

4 3D Pose Reconstruction

4.1 Human Model and 3D Pose Database

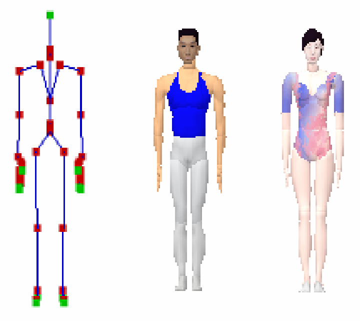


Fig. 2. Human model

Our human body model (Fig.2) based on VRML consists of kinematics 'skeletons' of articulated joints controlled by angular joint parameters ,covered by 'flesh' built from superquadric ellipsoids with additional tapering and bending parameters. We acquire a special motion database through an off-line motion capture process. In our practice, trampoline sport motion, an example of complex human motion, is selected to demonstrate the effectiveness of our approach. We collect the trampoline sport motion data by motion capture and build a standard 3D trampoline motion database, which include all the species of trampoline sport motion.

4.2 Extrinsic Parameters Estimation

3D human pose can be observed from the different viewpoints and the appearances are different. So it is necessary to abstract the viewpoint of the motion video and adjust the appearance of the 3D human model to ensure that the viewpoint of watching 3D model and that of video are the same. Then we can compare the difference between the silhouettes abstracted from video and projective silhouettes of 3D human model clearly.

The method for abstracting viewpoint from trampoline sport video is based on our previous work [13, 14], whose idea is to locate the trampoline in the image frame.

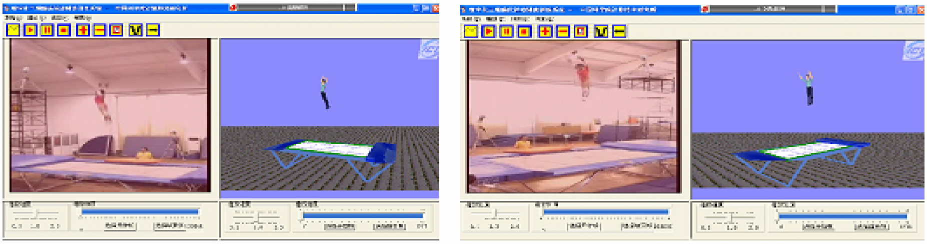


Fig. 3. Viewpoint Abstract and Simulation System Viewpoint Adjustment

4.3 2D Silhouettes Database

Once the viewpoint of the video (extrinsic parameters) has been abstracted and appearance of 3D human model is adjusted automatically, we can project the 3D pose of the human model and build the 2D silhouettes database of human pose.

In our practice, the simple image-based method is adopted to build the 2D silhouettes database of human pose. The particular steps are that: after abstract the viewpoint of the video, all the motion data of 3D database are driven with virtual athlete and displayed in the screen. At the same time, the 2D silhouettes of human pose are collected by capturing the screen and processing with the image processing technique.

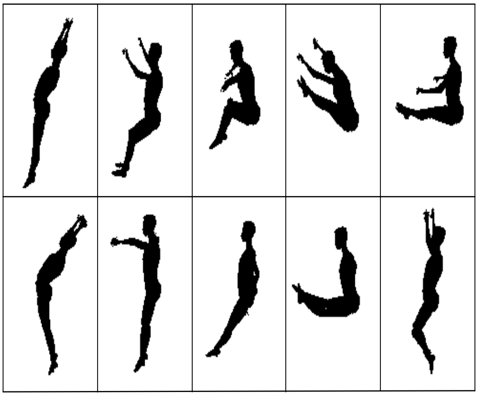


Fig. 4. 2D Silhouettes database built with one viewpoint

4.4 Human Pose Inferring Based on Shape Matching

We use the shape matching approach based on moment which is combination of affine moment invariants [16] and Hu moment invariants [17] to infer the 3D structure.

Moment invariants are features based on statistical moments of characters. They are traditional and widely-used tool for character recognition. Classical moment invariants were introduced by Hu（1962）.

Among seven moment invariants derived by Hu, $\phi_1,...,\phi_6$ are invariant under translation, rotation and scaling of the object and ϕ_7 are invariant under translation and scaling.

The AMIs were derived by means of the theory of algebraic invariants [16]. Supposed $I_1,...,I_4$ are the AMIs derived by Flusser and ϕ_7 is the 7th moment invariants derived by Hu [17]:

$$I_1 = \frac{1}{\mu_{00}^{4}}(\mu_{20}\mu_{02} - \mu_{11}^{2})$$

$$I_2 = \frac{1}{\mu_{00}^{10}}(\mu_{30}^{2}\mu_{03}^{2} - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{03}\mu_{21}^{3} - 3\mu_{21}^{2}\mu_{12}^{2})$$

$$I_3 = \frac{1}{\mu_{00}^{7}}(\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^{2}) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^{2}))$$

$$I_4 = \frac{1}{\mu_{00}^{11}}(\mu_{20}^{3}\mu_{03}^{2} - 6\mu_{20}^{2}\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^{2}\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^{2}\mu_{02}\mu_{12}^{2} + 12\mu_{20}\mu_{11}^{2}\mu_{21}\mu_{03}$$

$$+6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} - 8\mu_{11}^{3}\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^{2}\mu_{30}\mu_{12} +$$

$$9\mu_{20}\mu_{02}^{2}\mu_{21}^{2} + 12\mu_{11}^{2}\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^{2}\mu_{30}\mu_{21} + \mu_{02}^{3}\mu_{30}^{2})$$

$$\phi_7 = (3\upsilon_{21} - \upsilon_{03})(\upsilon_{30} + \upsilon_{12})[(\upsilon_{30} + \upsilon_{12})^2 - 3(\upsilon_{21} + \upsilon_{03})^2] + (3\upsilon_{12} - \upsilon_{30})(\upsilon_{21} + \upsilon_{03})$$

$$[3(\upsilon_{30} + \upsilon_{12})^2 - (\upsilon_{21} + \upsilon_{03})^2]$$

Where μ_{pq} is $p+q$ rank of central moment, υ_{pq} is the normed $p+q$ rank central moment, $\upsilon_{pq} = \frac{u_{pq}}{u_{00}^{w}}, w = \frac{p+q}{2} + 1$.

We defined the vector $(\phi_7, I_1, I_2, I_3, I_4)$ as the shape similarity vector and compute it with every 2D silhouette shape of the 2D pose database. Given raw video, we abstracted the human silhouette from image frame and at the same time, compute the shape similarity vector of the silhouette described as $L_i = (I_1, I_2, I_3, I_4, \phi_7)$. Then we compute the Euclidean distance between L_i and all the shape similarity vectors in 2D pose database. The silhouette in the 2D pose database whose Euclidean distance to L_i be the minimal to be selected and the corresponding 3D structure parameters are took as the structure of image silhouette.

5 Experiments

We have developed a prototype system for inferring 3D pose from video image. And it is implemented on an Intel Pentium IV 2.8GHz 512MB PC running WindowXP.

The experiments results of estimating 3D structure from image frame of trampoline video can be described in Fig.5. The first row are the image frames, the second row is the human silhouettes abstracted from image frames by the background subtraction algorithm. The third row is the silhouettes picked up from the 2D silhouettes database according to shape similarity and the forth row is the 3D reconstruction results described with surface geometrical model.

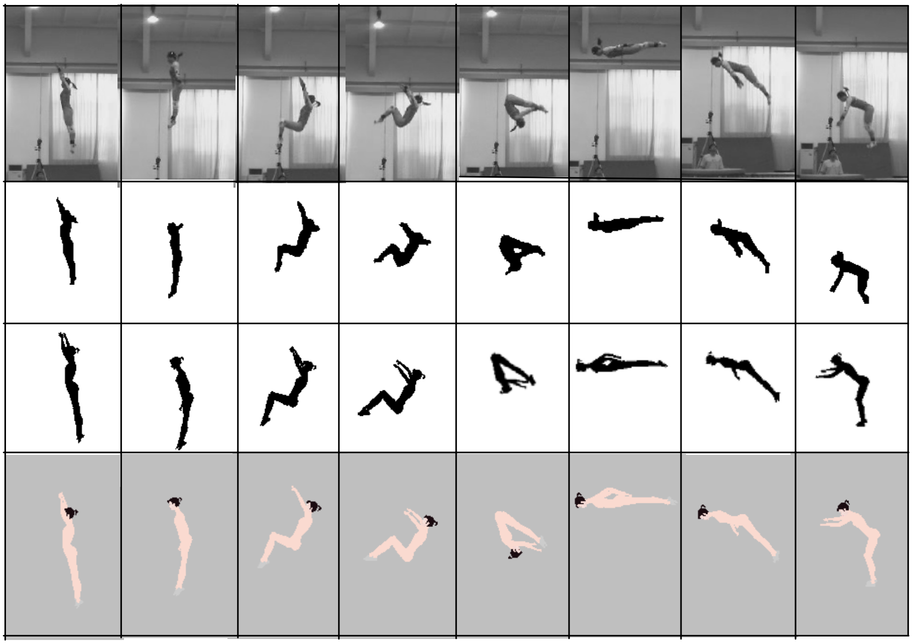


Fig. 5. 3D Pose Inferring from Motion Video

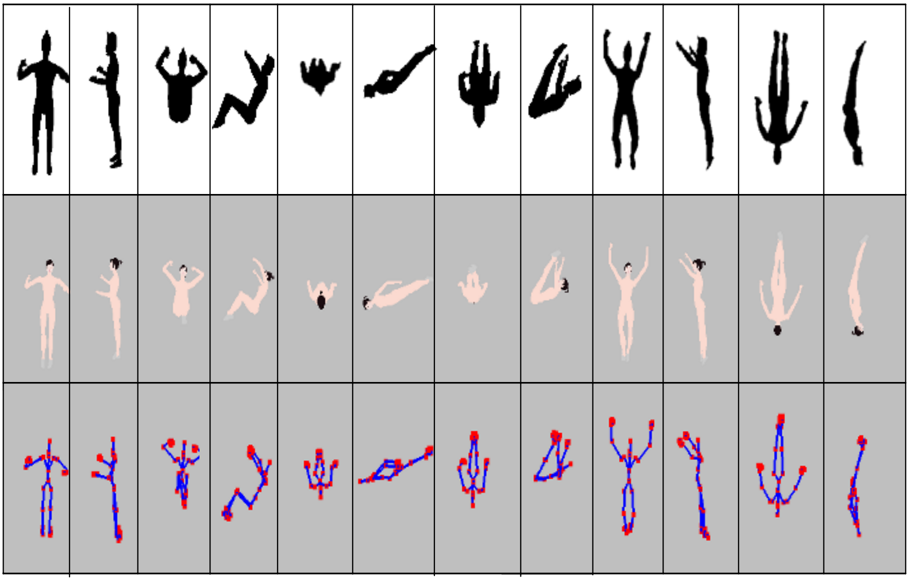


Fig. 6. 3D pose inferring from simulated video silhouette

In order to test the validity of our method in larger data, we have implemented our method on 2000 synthetic instances of human figure contours in random poses that were generated with a computer graphics package called Poser [19]. The result can be seen in Fig.6. The first row is the synthetic human figure contours and the second row are the reconstruction results described with surface geometrical model and the third row are the reconstruction results described with skeleton model.

6 Conclusion

In this paper we introduced a new silhouette-based framework for inferring human pose from video using a single monocular uncalibrated camera. We have verified with a great deal of instances and the experiments showed that the framework is efficient to estimate the 3D structure from complex motion video such as sport video. The major limitation to this approach is that it requires a database that is appropriate for the problem domain. We believe that this is not a serious limitation for many application areas where the likely human behaviors can reasonably be predicted in advance. Another possible weakness of the approach is matching ambiguities, since 2D silhouette can be corresponding to various 3D human pose. But the approach we present can be used to process the multi-view videos, that is, the reconstruction results is the 3D structure whose projective silhouette of geometrical model is most similar with the multiple silhouettes abstracted from multi-video synchronously.

Acknowledgments

This research is supported by NSF of China, Grant (60103007, 60403042,60473002); 863 Plan of China, Grant (2003AA114010, 2004AA115130, 2005AA114010); National Special Item for Olympics, Grant (2001BA904B08, KACX1-04, Z0004024040231); National 973 Project, Grant (2002CB312104, 2002CB312105).

References

1. J.M.Rehg and T.Kanade. Model-based tracking of self-occluding articulated objects. In ICCV, pages612–617,1995.
2. D. Gavrila and L. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In IEEE Conference on Computer Vision and Pattern Recognition, 1996.
3. H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In ECCV (2), pages 702–718, 2000.
4. M. Brand. Shadow puppetry. In International Conference on Computer Vision, volume 2, page 1237, 1999
5. T. D. Kristen Grauman, Gregory Shakhnarovich. Inferring 3d structure with a statistical image-based shape model. In ICCV, 2003.
6. Howe, Leventon, and W. Freeman. Bayesian reconstruction of 3d human motion from single-camera video. In Proc. NIPS, 1999.
7. G. Mori and J. Malik. Estimating human body configurations using shape context matching. In European Conference on Computer Vision, 2002.

8. G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In ICCV, 2003.
9. C. Sminchisescu, and B.Triggs, A Robust Multiple Hypothesis Approach to Monocular Human Motion Tracking, research report INRIA-RR-4208, June 2001.
10. J. Deutscher, and A. Blake, and I. Reid, Articulated Body Motion Capture by Annealed Particle Filtering, Proc. CVPR, vol. 2, pp. 126–133, 2000.
11. C. Sminchisescu, and B. Triggs, Covariance-Scaled Sampling for Monocular3D Body Tracking, Proc. CVPR, pp.447–454, 2001.
12. [Breg98] Bregler, C. and Malik, J. Tracking People with Twists and Exponential Maps, Proc. CVPR, 1998.
13. Qiu Xian-jie, Wang Zhao-qi and Xia Shi-hong. A novel computer vision technique used on sport video. Journal of WSCG, Prague,Czech Republic, 2004,545~554
14. Qiu Xian-jie, Wang Zhao-qi and Xia Shi-hong and Wu Yong-dong. A Virtual-Real Comparison Technique Used on Sport Simulation and Analysis. Journal of Computer Research and Development(Accepted). (In Chinese).
15. C. Sminchisescu and A. Telea, Human Pose Estimation From Silhouettes: A Consistent Approach Using Distance Level Sets. In the Proceedings of WSCG2002, Prague, Czech Republic.
16. Hu M K. Visual Pattern Recognition by Moment Invariants. IRE Trans. IT, 8,1962,179~182.
17. Flusser Jan Suk Tomas.Affine moment invariants: A new tool for character recognition. Pattern recognition Letters, 1994vol15,433–436.
18. W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-Based Visual Hulls. In Proceedings ACM Conference on Computer Graphics and Interactive Techniques, pages 369–374, 2000.
19. Egisys Co.Curious Labs.Poser5:The Ultimate 3D Character Solution.2002.

Integrating Collaboration and Activity-Oriented Planning for Coalition Operations Support

Clauirton Siebra and Austin Tate

Centre for Intelligent Systems and their Applications
School of Informatics, The University of Edinburgh
Appleton Tower, Crichton Street, EH9 1LE, Edinburgh, UK
{c.siebra, a.tate}@ed.ac.uk

Abstract. The use of planning assistant agents is an appropriate option to provide support for members of a coalition. Planning agents can extend the human abilities and be customised to attend different kinds of activities. However, the implementation of a planning framework must also consider other important requirements for coalitions, such as the performance of collaborative activities and human-agent interaction (HAI). This paper discusses the integration of an activity-oriented planning with collaborative concepts using a constraint-based ontology for that. While the use of collaborative concepts provides a better performance to systems as a whole, a unified representation of planning and collaboration enables an easy customisation of activity handlers and the basis for a future incorporation of HAI mechanisms.

1 Introduction

Coalitions, from Latin *coalescere* (*co-*, together + *alescere*, to grow) is a type of organisation where joint members work together to solve mutual goals. The principal feature of any coalition is the existence of a unique global goal, which motivates the actions of all coalition members. However, normally such members are not directly involved in the resolution of this goal, but in sub-tasks associated with it. For example, a search and rescue coalition that aims to evacuate an island (evacuate island is the global goal) has several sub-tasks (e.g., refill helicopters, provide information about weather conditions, etc.) that must be performed to reach the global goal.

The basic idea of this paper is to support coalition members via assistant agents that provide planning facilities. In this scenario each member could have a personal agent that supports his/her activities and delivers, for example, planning information, coordination commands and options to carry out activities. This approach is powerful because while human users have the ability to take decisions based on their past-experiences (case-base reasoning), agents are able to generate and compare a significant number of options, showing both positive and negative points of such options. Projects as *O-Plan* [1] and *TRAINS* [2] explore this fact, providing planning agents that interact in a mixed-initiative style with users during the development of solutions, such as plans.

However the use of standard planning mechanisms is not sufficient to deal with the complexity of problems associated with coalition domains, such as disaster relief

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 561 – 568, 2006.

operations. In these domains, activities cannot consist merely of simultaneous and coordinated individual actions, but the coalition must be aware of and care about the status of the group effort as a whole [3]. The *Helicopters Attack Domain* [4] and the *Guararapes Battle* game [5] are two experiments that corroborate with this affirmation. The principal reason for the problems in such domains is that "collaboration between different problem-solving components must be designed into systems from the start. It cannot be patched on" [6]. Thus, the idea of our approach is to develop the collaborative framework together with the planning mechanism in a unified way, using a constraint-based ontology for that. The properties of constraint manipulation have already been used by several planning approaches as an option to improve their efficiency and expressiveness [7]. Constraints are especially a suitable option to complement the abilities of *Hierarchical Task* Network (HTN) planning as, for example, to represent possible resultant subproblems dependences of the decomposition process [8]. The use of constraints will also facilitate a future expansion of our system toward a better human support. Note that constraints are a declarative way of providing information so that users specify what relationship must hold without specifying a computational procedure to enforce that relationship. Consequently users only state the problem while the computer tries to solve it [9].

The remainder of this document is structured as follows: section 2 presents some initial aspects of our approach associated with hierarchies, planning representation and process. Section 3 summarises the collaborative formalism that our models are based on, explaining how we are modelling its principal concepts using the constraint-based ontology and associated functions. Finally section 4 concludes this work.

2 Hierarchical Coalitions and Conceptualisation

This section introduces three important concepts for our work: the coalition organisation, the planning representation and the activity-oriented planning process.

Hierarchies are a well-known and used structure for organising members of a team. Military organisations, for example, are the most common examples of hierarchical arrangements. One of the principal advantages of hierarchies is that they support the deployment of coordination mechanisms because such mechanisms can exploit their hierarchical organisational structure. This is because the organisation implicitly defines the agents responsibilities, capabilities, connectivity and control flow. In addition, hierarchies also have the following advantages:

- They are compatible with the *divide-and-conquer* idea so that the process of splitting a problem into smaller sub-problems is repeated in each level;
- Hierarchical levels may deal with different granularities of knowledge so that each of them does not need to specify all the details about the problem, and;
- It is possible to enclose problems in local subteams, instead of spreading them along the entire organisation.

We are considering hierarchical organisations arranged into three levels of decision-making: strategic, operational and tactical. However this is not a "must restriction" so that hierarchies can be expanded to *n* levels. A last important aspect of hierarchical coalition systems is that there is a need of customising their agents so that they are able to support specific activities carried out into each decision-making level.

Our planning representation is based on <I-N-C-A> (Issues - Nodes - Constraints - Annotations) [10], a general-purpose constraint-based ontology that can be used to represent plans in the form of a set of constraints on the space of all possible plans in the application domain. Each plan is considered to be made up of a set of Issues and Nodes. *Issues* may state the outstanding questions to be handled and can represent unsatisfied objectives, questions raised as a result of analysis, etc. *Nodes* represent activities in the planning process that may have sub-nodes (sub-activities) making up a hierarchical description of plans. Nodes are related by a set of detailed *Constraints* of diverse kinds such as temporal, resource, spatial and so on. Annotations add complementary human-centric and rationale information to the plan, and can be seen as notes on their components.

The system architecture considers planning as a two-cycle process, which aims to build a plan as a set of nodes (activities) according to the <I-N-C-A> representation. In this way, the first cycle tries to create candidate nodes to be included into the agent's plan, respecting its current constraints. If this process is able to create one or more candidate nodes, one of them can be chosen and its associated constraints are propagated, restricting the addition of future new nodes. The agents' plan contains the net of activities that agents intend to perform. If an agent receives a new activity, it must generate *actions* that include this new node in its plan. Each action is a different way to perform this inclusion so that different actions generate different nodes configurations. We call this process of *activity-oriented planning* because agents provide context sensitive actions (e.g., delegations, Standard Operating Procedures, dynamic plan generation and specific solvers) to perform activities.

In brief, the role of agents is to provide actions to decompose nodes until there are only executable nodes. Each action is implemented by an activity handler, which uses one or more different constraint managers to validate its constraints. For example, the action of applying a SOP is a handler that decomposes an activity according to the SOP specification. For that, the handler uses specific constraint managers that check the pre-conditions in which the SOP can be applied, signalising in case of conflicts.

3 The Collaborative Framework

Planning agents are able to support the performance and coordination of activities, however they do not ensure collaboration between coalition members. For that, coalition systems must also consider notions of collaboration since their phase of conception. Considering such fact, this section summarises the collaborative formalism that we are exploring and how we are implementing its ideas following the <I-N-C-A> approach.

3.1 Requirements of the Teamwork Theory for Collaboration

The *Teamwork Theory* [11] provides a set of formal definitions that lead the design of collaborative systems. Several works have proposed frameworks based on such definitions. *SharedPlans* [12], for example, argues that each collaborative agent needs to have mutual beliefs about the goals and actions to be performed, and about the capabilities, intentions and commitments of other agents. *STEAM* [13] is an implemented model of teamwork, where agents deliberate upon communication necessities during the establishment of joint commitments and coordination of responsibilities. Although these and other works have different approaches to deal with different technical problems, an investigation on these works can distinguish the following principal requirements: (1) agents must commit on the performance of joint activities; (2) agents must report the status of their activities, informing on progress, completion or failure, and; (3) agents must mutually support the activities of other coalition members, sharing useful information or using their own resources for that.

3.2 Activities Commitment

We are implementing both ideas of activities commitment and status reporting through the "AgentCommitment" function. This section analyses the commitment of activities, while the next section (Section 3.3) discusses the status reporting process.

According to <I-N-C-A>, each plan *p* is composed by a set of plan nodes n_i. If a superior agent, that has *p* as goal, sends such nodes n_i to its subordinate agents, then a commitment between the superior and each subordinate must be done. For that end, the function "AgentCommitment", implemented by each agent of the hierarchy, receives as input parameter the node $n_i \in p$ and the *sender* identifier.

```
01. function AgentCommitment($sender$,$n_i$)
02.    $subplan$ ← GenerateNodes($n_i$)
03.    if($\exists$($subplan$))
04.       if(HasNodesToBeDelegated($subplan$)) then
05.          Delegate($subplan$,$subord$) $\wedge$ WaitCommits()
06.          if $\exists s$ ($s \in subord$)$\wedge$($\neg$Commit($s$)) then
07.             go to step 04
08.       Report($sender$,$n_i$,COMMITED)
09.       while($\neg$Complete($subplan$))
10.          if(JustReady($subplan$)$\vee$Changed($subplan$))then
11.             Report(sender,$n_i$,EXECUTING)
12.          if(Violated($subplan$)$\vee$Receive(FAILURE)) then
13.             go to step 04
14.       end while
15.       Report(sender,$n_i$,COMPLETION)
16.    else
17.       Report($sender$,$n_i$,FAILURE)
18.       $\forall s$ ($s \in subord$) $\wedge$ HasCommitment($s$,$subplan_s$),
19.          Report(s,$subplan_s$,FAILURE)
20. end
```

At this point we can discuss some implications and features of this function. First, n_i has a set of constraints associated with it so that the "GenerateNodes" function (step 02) considers such set to return an option (nodes list or *subplan*) to perform n_i. If a subplan is possible (step 03) and it does not depend of anyone more (step 04) then the agent can commit on n_i (step 08). However, if *subplan* depends on the commitments of subordinates, then the agent must delegate the necessary nodes to its subordinates and wait their commitments (step 05). This implies that commitments are done between a superior agent and their subordinates and, starting from the bottom, a "upper-commitment" only can be done if all the "down-commitments" are already stabilised. Note that a delegation on the same *subplan* cancels previous commitments.

Second, an interesting implication is that if some subordinate agent is not able to commit (step 06), the agent returns (step 07) to generate other subplan option rather than sending a failure report to its superior. This approach implements the idea of enclosing problems inside the sub-coalition where they were generated.

Finally if the agent is not able to generate a *subplan* for n_i, it reports a failure to its superior (step 17). In addition, it must also alert their subordinates that n_i has failed and consequently its subnodes can be abandoned (step 18-19).

We can note that the principal advantage of this approach is that commitments are naturally manipulated during the constraint processing as any other constraint. In other words, the failure in a commitment is interpreted as a problem of constraint satisfaction, which can trigger a common process of recovery, such as a replanning.

3.3 Status Reporting

After reporting a commitment (step 08), an agent a_i must monitor and report execution status until the completion/failure of n_i. The principal questions here are when to send a report and which information should be reported. There are two obvious cases in which a_i performing n_i must report: when a_i completes n_i and when happens an execution failure so that a_i is not able to deal with the failure by itself. In the first case a_i only needs to send a completion report, while in the second case a failure report must be sent optionally with the failure reasons (constraints unsatisfied).

Reports associated with progress are a more complex case. In order agents do not have to communicate each step of their execution. Previous works have already identified communication as a significant overhead of risk in hostile environments [13]. Furthermore, in case of progress reports, the information associated with them should be useful to the monitoring process.

Considering these facts, we start from the principle that relevant information, once sent, becomes common knowledge and hence unnecessary to updates. For example, if a_i informs that it has just started the execution of n_i, a_i does not need to send other messages informing that it still executing n_i. From this point we must think on which could be the information generated during activities execution and that is likely to help superior agents during the process of monitoring. For that end, consider the following scenario: when a_i commits on the performance of n_i, it informs which *subplan* it is going to use. The constraints of such *subplan* can have the following classes of values: *concrete*, which expresses known values or estimated values if a_i has a good idea about the process; and *variables* to be used during unpredicted situations, indicating that a_i does not have idea about some specific information.

Using constraints with concrete values, the superior agent of a_i can perform more confident reasoning on the ongoing activities. For example, if it knows that the activity of a_i will take 30 minutes, it can allocate a_i to a new activity after 30 minutes.

Thus, if during the execution of *subplan* one of its concrete values is changed or its variables are instantiated, a progress report must be sent.

Returning to the function, while the planning is not complete (step 09), the changes in *subplan* (step 10) are monitored and sent to superiors as a ongoing execution report (step 11). Constraint violations and failure reports are also monitored (step 12) so that the agent firstly tries to repair the problem by itself (step 13) before sending a failure report. From this discussion we can stand five different plan (or subplan) status: possible, no-ready, impossible, complete and executing. The <I-N-C-A> definition for activities contains a status attribute that can be filled with one of these options.

At last, the principal point of this function is that the reasoning associated with commitments and reports are based on results of constraint processing. This fact is illustrated by the functions "Complete" (step 09), "JustReady" (step 10), "Changed" (step 10) or Violated (step 12), which are all applied on constraints. Thus, we still having the same basis for working, which is also used by the planning mechanisms (activity handlers and constraint managers).

3.4 Mutual Support

The principal idea behind mutual support is to enable that one agent has knowledge about the needs of other agents. For example, an agent a_1 knows that a specific road is clean so that it uses this constraint in its plan. However, as the world is dynamic, the road can be blocked. If any other agent finds out that such road is no longer clear, it must inform this fact to a_1. Thus this informer agent is supporting the performance of a_1. The easier option to implement this feature is to force agents to broadcast any new fact to all coalition. Consequently all agents will have their world state updated and problems like that can be avoided. However, this is not an appropriate approach in terms of communication and agents will also receive several useless information.

Consider now that Θ_x is a coalition and that each agent $a_i \in \Theta_x$ has a plan p_i with a set of conditional constraints C, which a_i desires that hold so that p_i still valid. In this case each $c_i \in C$ is a constraint that a_i believes that is true and hopes that still being true. Then a_i broadcasts C for Θ_x so that other agents of its subgroup know what it needs. Such agents must deal with C according to the function below:

```
01. function MutualSupport($a_i$,$C$,$myBel$)
02.    while($\exists c_i \; c_i \in C$)
03.       if($\exists c_i c_k \; c_i \in C \wedge c_k \in myBel \wedge$ Contrast($c_i$,$c_k$)) then
04.          $newActivity \leftarrow$ CreateActivity(Goal($c_i$))
05.          if($\neg\exists$newActivity) then Inform($a_i$,$c_k$)
06.          Retire($c_i$,$C$)
07.       if($\exists c_i \; c_i \in C \wedge \neg$Valid($c_i$)) then Retire($c_i$,$C$)
08. end
```

According to this function, which is applied by agents that receive C from a_i, agents must compare their believes *myBel*, which are also a set of constraints, with C. If they find some "contrast" (step 03), they must try to create a new activity whose goal is to turn c_i true (step 04). If this is not possible, they must inform a_i that c_i is no longer holding and its new value is c_k (step 05).

The Contrast function (step 03) says that two constraints, which are supposed to be equal, are not equal. However we are also considering as contrast the situation where two constraints have the potential to be equal. For example, ((color Car) = ?x) and ((color Car) = blue). In this case the two constraints are in contrast because they have the potential to be equal if the variable ?x assume the value "blue". This second type of contrast is very useful in the following class of situations. Suppose that one of the activities of a_i is "rescue injured civilians". For that end, a_i firstly needs to find such civilians so that they have the following conditional constraints: ((position ?a) = ?b), ((role ?a) = civilian) and (status ?a) = injured). This set of constraints implies that the variable ?b is the location of an injured civilian ?a. Then if other agents that have or discover a set of constraints that contrast with the set sent by a_i, they must inform a_i (note that in this case no make sense to create a new activity).

The Valid function (step 07) accounts for eliminating the constraints that no longer represent conditions to sender agents. This is important to avoid that agents still sending useless information and also to decrease the number of messages in the coalition. A practical way to do that is to consider that all $c_i \in C$ has a timestamp that indicates the interval that such constraint is valid. Using a timestamp as (t_{initial},t_{final}) and considering that t_{initial} and t_{final} are ground values, the Valid function only needs to compare if the condition (t_{final} < current-time) is true to eliminate the respective constraint. However, the use of timestamps fails if sender agents does not know the interval that their conditional constraints must hold. Note that the use of timestamps tries to avoid that agents need to broadcast messages saying that they no longer need that a group of constraints hold. Rather, timestamps enable that agents reason by themselves on the elimination of such constraints.

One of the principal advantages of the MutualSupport function is that it improves the information sharing because the sending of information is guided by the constraint-based knowledge that each agent has on the activities of its partners. In addition, the function also decreases the number of messages changed among agents.

4 Conclusion and Directions

This work shows the integration of an activity-oriented planning with notions of collaboration via a unified constraint-based framework. This framework enables an easy development of activity handlers, which can be customised according to the tasks of each decision-making level. Ongoing experiments of this proposal are using the RoboCup Rescue simulator as principal domain of evaluation. The first idea is to demonstrate the importance of coordination and collaboration during coalition operations. However, the principal purpose is to show that the development of planning mechanisms can be maintained independent of the collaborative framework.

Acknowledgement

Clauirton Siebra' scholarship is sponsored by CAPES Foundation under process number BEX2092/00-0. This material is based on research within the I-X project sponsored by the Defense Advanced Research Projects Agency (DARPA) and US Air

Force Research Laboratory under agreement number F30602-03-2-0014 and other sources.

The University of Edinburgh and research sponsors are authorised to reproduce and distribute reprints and on-line copies for their purposes not withstanding any copyright annotation here on. The views and conclusions contained here in are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of other parties.

References

1. Tate, A.: Mixed-Initiative Interaction in O-Plan. Proceedings of the AAAI Spring Symposium on Computational Models for Mixed-Initiative Interaction, Stanford, California, USA (1997)
2. Fergunson, G., Allen, J., Miller, B.: TRAINS-95: Towards a Mixed-Initiative Planning Assistant. Proceedings of the Third Conference in AI Planning Systems, AAAI Press, Menlo Park, California, USA (1996) 70-77
3. Levesque, J. Cohen, P., Nunes, J.: On Acting Together. Proceedings of the Eighth National conference on Artificial Intelligence, Los Altos, California, USA (1990) 94-99
4. Tambem, M.: Towards Flexible Teamwork. Journal of Artificial Intelligence Research, Vol. 7 (1997) 83-124
5. Siebra, C., Ramalho, G.: An Architecture to Support Synthetic Actors in Virtual Environments, Second Brazilian Workshop on Artificial Intelligence, Rio de Janeiro, Brazil (1999)
6. Grosz, B.: Collaborative Systems, AI Magazine, Vol. 17-2 (1996) 67-85
7. Nareyek, A., Fourer, R., Freuder,, E., Giunchiglia, E., Goldman, R., Kautz, H., Rintanen, J., Tate, A.: Constraints and AI Planning, Notes from the AAAI Workshop on Constraints and AI Planning, Austin, Texas, USA (2000)
8. Stefik, M.: Planning with Constraints (MOLGEN: Part 1). Artificial Intelligence, Vol. 16-2 (1981) 111-140
9. Freuder, E.: Synthesizing Constraint Expressions. Communications ACM, Vol 21-11 (1978) 958-966
10. Tate, A.: <I-N-C-A>: an Ontology for Mixed-Initiative Synthesis Tasks. Proceedings of the IJCAI Workshop on Mixed-Initiative Intelligent Systems, Acapulco, Mexico (2003)
11. Cohen, P., Levesque, H.: Teamwork, Special Issue on Cognitive Science and Artificial Intelligence, Vol. 25 (1991) 487-512
12. Grosz, B., Hunsberger, L., Kraus, S.: Planning and Acting Together, AI Magazine, Vol. 20-4 (1999) 23-34
13. Tambe, M.: Towards Flexible Teamwork, Journal of Artificial Intelligence Research, Vol. 7 (1997) 83-124

Laser-Based Localization with Sparse Landmarks*

Andreas Strack[1], Alexander Ferrein[2], and Gerhard Lakemeyer[2]

[1] Robotics Group
Department for Mathematics and Computer Science
University of Bremen, Bremen, Germany
strack@informatik.uni-bremen.de
[2] Knowledge-based Systems Groups
RWTH Aachen, Aachen, Germany
{ferrein, gerhard}@cs.rwth-aachen.de

Abstract. Self-localization in dynamic environments is a central problem in mobile robotics and is well studied in the literature. One of the most popular methods is the Monte Carlo Localization algorithm (MCL). Many deployed systems use MCL together with a laser range finder in well structured indoor environments like office buildings with a rather rich collection of landmarks. In symmetric environments like robotic soccer with sparse landmarks which are occluded by other robots, most of the time the standard method does not yield satisfying results. In this paper we propose a new heuristic weight function to integrate a full 360° sweep from a laser range finder and introduce so-called *don't-care* regions which allow to ignore some parts of the environment. The proposed method is not specific to robotic soccer and scales very well outside the soccer domain.

1 Introduction

Self-localization in dynamic environments is a central problem in mobile robotics and is well studied, leading to many satisfying approaches which can be found in the literature such as [5, 2, 1].

Most localization algorithms follow a probabilistic approach. The most popular among these is the Monte Carlo Localization algorithm (MCL) [1]. Many applications of this method use a laser range finder (LRF) for perceiving the environment. MCL with LRF works best in environments with many landmarks.

In environments where landmarks are sparse, on the other hand, the results are far less satisfying. One such domain is the Middle-size league of robotic soccer, where up to ten mobile robots are competing on a field of the size of 12×8 meters. Available landmarks are the goals and the corner posts. To make the task even harder, they are often occluded by other robots.

* This work was supported by the German National Science Foundation (DFG) in the Priority Program 1125, *Cooperating Teams of Mobile Robots in Dynamic Environments* and by a grant of the Ministry of Science and Research of North Rhine-Westphalia, Germany.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 569–576, 2006.

For these reasons, the combination of a LRF and MCL is presently not the method of choice for self-localization in RoboCup. Most teams use vision-based systems for the purpose of localization. Nevertheless, our team, the "AllemaniACs", successfully deploys a Monte Carlo approach with a laser range finder in that environment. In this paper we present two modifications to MCL with LRF which allows it to be a viable and robust method for self-localization in RoboCup.

In principle, the sparsity of landmarks can be dealt with by taking many single measurements in a sweep from the laser scanner. However, this makes the use of the standard MCL algorithm intractable because the range of weights for the samples grows exponentially in the number of single readings. To circumvent these computational problems we propose a heuristic weight function. Furthermore, we introduce so-called *don't-care* regions in maps that ignore the regions outside the field and thus enables incomplete specification of environments.

It turns out that our approach, which was inspired by the RoboCup setting, scales very well for indoor navigation in large environments.

The rest of the paper is organized as follows. In Section 2 we briefly introduce the MCL method and some extensions as well as some related work in the field. In Section 3 we present our modifications to the MCL algorithm. Before we conclude with Section 5, we show some experimental results in Section 4.

2 Related Work

By far the most approaches for the localization task use probabilistic methods. A belief distribution $\mathbf{P}(l)$ describes the probability that the robot is located at pose $l = (x, y, \theta)$, where x and y are Cartesian coordinates, and θ denotes the orientation of the robot.

The belief is updated using the sensory perceptions. $e_{1:t} = ((u_1, z_1) \dots (u_t, z_t))$ denotes a sequence of proprioceptive sensor information u_i, and exteroceptive measurements z_i. The former are given as odometry measurements, and the latter are measurements from, for example, a LRF or a camera.

Together with the Markov assumption that perceptions at time t are statistically independent from former evidence the update is given by

$$\mathbf{P}(\mathbf{L}_{t+1}|e_{1:t+1}) = \alpha\, \mathbf{P}(z_{t+1}|\mathbf{L}_{t+1}) \cdot \sum_{l_t} \mathbf{P}(\mathbf{L}_{t+1}|l_t, u_t)\, p(l_t|e_{1:t}).^1 \qquad (1)$$

$\mathbf{P}(z_{t+1}|\mathbf{L}_{t+1})$ and $\mathbf{P}(\mathbf{L}_{t+1}|l_t, u_t)$ are probability distributions denoting the *perception model* and the *motion model*, respectively. α is a normalization factor ensuring that all probabilities in the resulting distribution sum up to one.

The recursive nature of Eq. 1 allows for updating the belief at time $t + 1$ in terms of the belief at time t. It is called a *recursive Bayes Filter*.

Implementations of the Bayes Filter differ mainly in the representation of the belief. For example, Kalman Filter (KF) based approaches use a unimodal Gaussian distribution (e.g. [5]). In contrast, grid-based Markov Localization (ML)

[1] Here $\mathbf{P}$ denotes a probability distribution whereas p denotes a single probability.

stores the distribution in a discrete grid covering the state space. While the former methods are computationally more efficient and accurate, their restriction to unimodal distributions makes it impossible for them to perform global localization (finding the position without initial knowledge). ML is able to solve this task and also the kidnapped-robot problem, which means finding the correct position again after the filter converged to a wrong position.

A combination of both methods, called ML-EKF ([3]), combines the robustness of ML and the accuracy of the KF. MCL is a further refinement of ML, replacing the probability grid by the already described sampling mechanism. As the filter converges, the samples gather around positions of high probability. An experimental comparison [4] of the described localization methods showed that the ML-EKF approach performs about equally well as MCL.

In this paper, we will make use of the Monte Carlo Localization algorithm [1]. It works by approximating the belief by a set of weighted samples:

$$\mathbf{P}(l_t) \sim \{(l_{1,t}, w_{1,t}), \ldots, (l_{N,t}, w_{N,t})\} = \mathbf{S}_t. \tag{2}$$

Once the initial sample set is given, the Monte Carlo algorithm works in three steps:

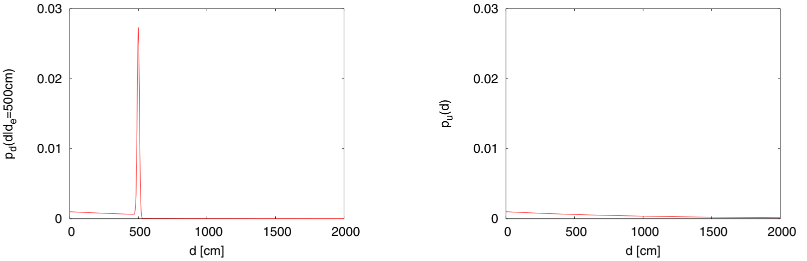
1. In the *prediction step* the samples are moved according to the odometry information u_t. Noise is added to the movement according to the known relative error of odometry.
2. In the *weighting step* the samples are weighted with the perception model using the exteroceptive sensory data z_{t+1}.
3. In the *re-sampling step* a new sample set is drawn. The distribution of samples represents the distribution of weights of the weighting step.

3 A Heuristic Perception Model for MCL

In this section, we present the modifications to the MCL method to be able to localize with a laser range finder in environments with sparse landmarks. First, we briefly discuss *don't-care* regions and their integration into the sensor model. Then we introduce our heuristic weight function.

3.1 Don't Care Regions in Occupancy Grid Maps

We use occupancy grid maps (as in [2]) for representing the environment. Each cell of the grid stores the probability of this cell being occupied by an obstacle. Fig. 2 presents an example of a RoboCup field. Black regions denote an occupancy with a probability of 1 (the goals and the posts), and white regions are free areas. With the help of this information one can determine for each position on the map which distance a laser ray pointing to a certain direction should measure. This value will be referred to as the *expected distance* in the following. The red border around the field in Fig. 2 represent our *don't-care* extension to occupancy grid maps. In these areas simply no information about occupancy is given. This models an incompletely specified environment.



(a) Sensor model with an expected distance $d_e = 500$cm

(b) Sensor model for a don't care–region

Fig. 1. Sensor model for a single distance measurement for known and unknown expected distance

When using a 2D laser range finder, the weighting step of MCL can be performed as described in [2]. The perception model $p(d|l)$ for a single laser beam describes the probability that the laser beam traveling in a certain direction from l will measure the distance d.

We need the environment model to distinguish two cases: (1) the laser beam will hit an obstacle; (2) the laser beam will hit a don't care-region. In the first case we know the expected distance of the measurement. According to [2], this yields the single-beam perception model shown in Fig. 1(a). The peak represents a Gauss curve assigning a high probability to the laser beam being reflected at the expected distance. Note that the probability of the measurement being shorter than expected is significantly higher than the probability of its being longer. This is due to the possibility of dynamic occlusion. Because the probability of occlusion is equal at all distances, it is modeled by a geometric distribution. The merging of the geometric distribution with the Gaussian yields the model displayed in Fig. 1(a).

As one does not have information of an expected distance in the case when a laser ray is hitting a *don't-care* region, the perception model is reduced by the Gaussian part. This yields a purely geometric distribution shown in Fig. 1(b).

3.2 The Heuristic Perception Model

In order to perform the actual weighting of samples one has to combine the probabilities of single-beam perception for a given sample position l and a full 2D-sweep $z = (d_1, \ldots, d_n)$ of the laser range finder by multiplying the weights [2]:

$$p_{mul}(z|l) = \prod_{i=1}^{n} p(d_i|l). \tag{3}$$

The weight range is exponential in the number of single measurements. With 360 readings it is practically impossible to weight the samples in that exponential range. To give an example, suppose that we have two positions l_1 and l_2 with uniform weights of 0.01 for l_1 and 0.025 for l_2. p_{mul} yields the following weights:

$$p_{mul}(l_1|z) = 0.01^{360} = 10^{-720} \text{ and } p_{mul}(l_2|z) = 0.025^{360} \approx 10^{-576}.$$

The re-sampling step of MCL draws samples proportionally to their weights. This means that in this example the sample containing position l_2 has to be drawn 10^{144} times (!) as often as the sample with position l_1. Practically, this is impossible because one cannot handle a sample set as large as this.

Note that using the logarithm of p_{mul} does not provide a solution to this problem. One can handle the weight range by doing so and also sample from the logarithm of the sample distribution. However, the resulting sample distribution would have to represent the proportions of the weights *before* using the logarithm. Thus, the sizes of the sample sets would still be intractable.

It may seem that the problem could be fixed simply by reducing the number of measurements to a manageable size.[1]However, in the RoboCup scenario this does not work since typically up to 90% of the readings are useless due to the sparsity of landmarks and occlusions. Hence even dropping a few readings risks losing the few precious good readings. Instead, we propose to use all readings for re-sampling, but replace the product by the sum of the measurements:

$$p_{add}(z|l) = \sum_{i=1}^{n} p(d_i|l).$$

In contrast to the multiplicative model the weight range is now linear in the number of single measurements of a 2D-sweep. Considering Fig. 1(a) the weights range from about 0 to about $360 \cdot 0.025 = 9$.

Let us consider how the heuristics p_{add} differs from the mathematically correct model p_{mul}. First, it is easy to see that the former changes the proportions of the weights:

$$\frac{p_{add}(z|l)}{p_{add}(z'|l)} \neq \frac{p_{mul}(z|l)}{p_{mul}(z'|l)} \quad \text{for } \textit{most } z, z', l.$$

Furthermore, it does not preserve the order:

$$p_{add}(z|l) > p_{add}(z'|l) \not\equiv p_{mul}(z|l) > p_{mul}(z'|l) \quad \text{for } \textit{many } z, z', l.$$

Thus, the additive perception model may prefer positions to others that would not have been favored by the multiplicative model. A simple example for such a situation is the following: Let $z = (d_1, d_2)$ and l_1, l_2 such that $p(d_1|l_1) = 0.06$, $p(d_2|l_1) = 0.01$, $p(d_1|l_2) = 0.04$, and $p(d_2|l_2) = 0.02$. Now it follows that $p_{mul}(z|l_1) = 0.0006 < 0.0008 = p_{mul}(z|l_2)$ and $p_{add}(z|l_1) = 0.07 > 0.06 = p_{add}(z|l_2)$.

What one can learn from this example is that the additive model tends to assign higher *overall* weights than the multiplicative model to positions with high *single* weights. Expressed in different terms, the low weights do not have such a great impact on the sample weighting as with the multiplicative model.

Having a look at Fig. 1(a), low single-beam weights can have two reasons: (1) The reading is dynamically occluded, or (2) The reading is longer than

[1] Experience shows that a reasonable number is in the order of 40.

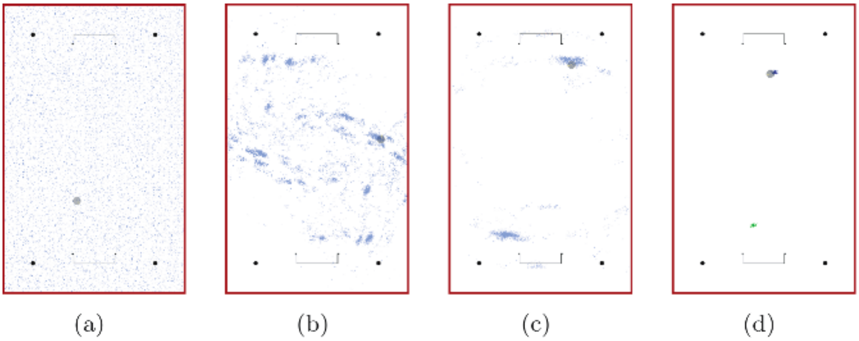


Fig. 2. Simulated global localization on the RoboCup field. The real position of the robot is depicted by the gray circle. Because of the symmetry of the environment model, two clusters have developed.[3]

expected. In the first case it is desirable that a low weight does not pull down the weight of a correct hypothesis. In the second case, however, the weighting function should reduce the weight of a position by the low single-beam weight. p_{add} works well in the first case while it fails in the second. In practice, however, it turned out that even in this case the correct hypotheses were supported and the algorithm converges (cf. Fig. 2). The convergence behavior is subject to further investigation.

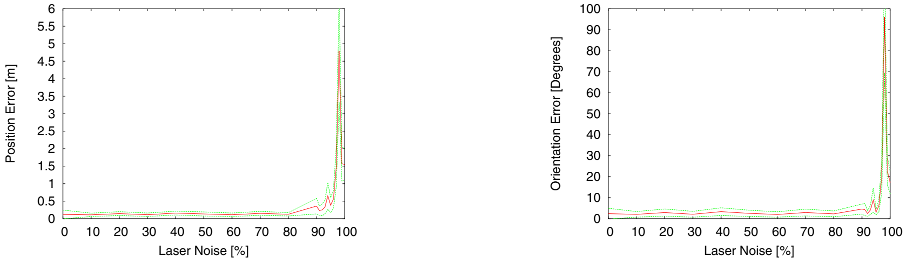
4 Experimental Results

We have tested our method extensively, both in simulation and with real robots at RoboCup events. We will now present results concerning the accuracy and robustness of our approach.

We used a map of a a RoboCup field as shown in Fig. 2 for the evaluation. It contains just the goals and the corner posts of a RoboCup field. We changed the noise level of the LRF in order to gain meaningful results. A noise level of $n\,\%$ means that we set $n\,\%$ of the single readings randomly shorter than the reading from the simulator. This simulates dynamic objects causing too short readings. The distribution for the random shortening was uniform.

During the experiment with movement the robot traveled at an average speed of 1.74 m/s and 34°/s, with a maximum speed of 3 m/s and 225°/s. Fig. 3 shows that below a critical noise level of 90% the accuracy is about 15 cm in the pose and 4° in orientation. Above a noise level of 95% localization is no longer possible. One can get an intuitive understanding of what a loss of 90% of the laser information means by calculating how many laser beams are still useful in that case. For example, if the robot is placed in the middle of the field, about 12%

[3] This ambiguity cannot be resolved by MCL with LRF due to the inherent symmetry of the environment. In practice, it is resolved simply by using the information about the color of nearest goal provided by the camera used for ball tracking.



(a) Pose Error

(b) Orientational Error

Fig. 3. Accuracy of position tracking by the MCL module. The red line shows the mean error, and the green lines represent a one–σ–environment around the mean.

Table 1. Position Losses

RoboCup 2003							
Team	Artisti	Trackies	Eigen	Cops	Persia	ISePorto	AIS
Position Losses	0	1	7	4	2	1	5

German Open 2004						
Team	Paderkicker	Minho	Philips	FUFighters	Persia	AIS
Position Losses	0	0	0	0	1	1

of its laser measurements correspond to usable landmarks. A loss of 90% means that only 1%-2% remain which means 3-7 distance measurements.

We gathered data from two RoboCup events the AllemaniACs took part in in order to gain results about the robustness of our localization approach. In order to do so, we counted dis-localizations during each of the matches. The results are shown in Table 1. All position losses were caused by severe failures of odometry due to slippage caused by collisions. When this happens, the robot senses a movement suggesting that it translated or rotated much farer than it actually did.

5 Conclusion

In this paper we presented two extensions to the MCL algorithm that made it possible to use the approach in connection with a 360° LRF in an environment with sparse landmarks like the Middle-size league of RoboCup soccer.

We adapted the perception model for laser range measurements so that many readings can be used. While usual implementations use about 20–40 measurements per scan, we needed to make use of a whole sweep of 360° at a resolution of 1° because of the sparsity of landmarks and high sensor occlusion in RoboCup. Our heuristic perception model exhibited good performance in this setting. As the only source for dis-localizations was slippage, we suggest to add some kind of passive odometry to the sensory equipment of the robot. This should yield a drastic increase in localization robustness.

We regard the fact as remarkable that MCL is at all able to keep track of the position in the presence of a constant laser noise of 90% and fast movement of the robot. We ascribe the good performance of our MCL Module in the presence of laser noise to the characteristic of the additive perception model (cf. Section 3.2) that it prefers position hypotheses with high individual weights.

The drawback of the additive model is the weak effect weights have on the weighting of a position for readings that are longer than the expected distance. This plays a role primarily in global localization. However, we found out that our approach works well in this case, too, as is shown, for example, in Fig. 2. Although the weights for wrong positions are not pulled down as strongly as with the multiplicative model, it turns out that the correct hypotheses still have a higher overall weight and are thus preferred for convergence.

Concluding, the additive model is a heuristic which turned out to work very well in practice. In the future we want to investigate the mathematical properties of that model and compare it with the multiplicative model in detail.

The second adaption, the *don't-care* regions, represent an extension to occupancy grid maps enabling one to mark regions where the map has no information about occupancy. The results in localization performance showed that the extension works well. An opportunity to test the performance of our approach in large indoor environments was during RoboCup 2004 at Lisbon / Portugal. We won the silver medal in the technical challenge by autonomously driving all the way from our team area to the soccer field. Thus, our approach to localization in RoboCup is not only well suited for this domain but it scales up well in larger indoor environments.

References

1. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. of the Int. Conf. on Robotics and Automation (ICRA)*, May 1999.
2. D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 1999.
3. J.-S. Gutmann. Markov-Kalman localization for mobile robots. In *Proc. of Int. Conf. on Pattern Reconition (ICPR)*, 2002.
4. J.-S. Gutmann and D. Fox. An Experimental Comparison of Localization Methods Continued. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, 2002.
5. J.-S. Gutmann, W. Hatzack, I. Herrmann, B. Nebel, F. Rittinger, A. Topor, and T. Weigel. Reliable self-localization, multirobot sensor integration, accurate path-planning and basic soccer skills: playing an effective game of robotic soccer, 1999.

Lightweight Management - Taming the RoboCup Development Process

Tijn van der Zant and Paul G. Plöger

FhG Institute of Autonomous Intelligent Systems,
Schloss Birlinghoven,
53754 St. Augustin, Germany
tijn@aisland.org,
paul-gerhard.ploeger@ais.fraunhofer.de

Abstract. RoboCup projects can face a lack of progress and continuity. The teams change continuously and knowledge gets lost. The approach used in previous years is no longer valid due to rule changes and specialists leaving the team leave black boxes that no-one understands. This article presents the application of a recent software development technique called eXtreme Programming to the realm of RoboCup. Many common problems typical for teams of students seem to be solvable with this technique. It also gradually spreads out in professional software production companies. Students mastering it are of high use for their further career after having left the university. The strategy is being tested on a real RoboCup Mid-Size and an Aibo league project and produces very promising results. The approach makes it possible to modularize scientific knowledge into software that can be re-used. Both the scientist/expert, who has the knowledge, and the software development team benefit from this approach without much overhead on the project.

1 Introduction

1.1 RoboCup

RoboCup is an extremely difficult to approach problem and the complexity makes it impossible for single individual researchers to tackle it. It seems mandatory to attempt a solution in a team. In many teams members change frequently and the interest in RoboCup may vary substantially. Devotion of all work power to a single well defined topic exclusively is a rare exception. These discrepancies pose some of the following questions which are quite typical for many RoboCup teams:

- How can one avoid starting anew each tournament, i.e. how to support systematically the enhancement of a existing behavior system? Every new team member needs to get acquainted to the source code. The documentation is hard to maintain and can lack the clarity needed to start programming.
- How can one address the problems caused by permanent changes of the rules? Active and regular participation in tournaments requires a never-ending and constant improvement. This is very much akin to try to meet the demands of an external customer, who is not only never satisfied, but constantly changes the requirements.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 577–584, 2006.

– How can one cope with frequent team turn-overs, i.e how to avoid a significant drop in performance or the resign of the team if some key system designer graduates, leaving the team behind. How to either fill or avoid a knowledge gap?
– How can one foster cooperation in the team, i.e. how to lower the danger of a complete dependence on the expertise of a single member?

1.2 Correlations with the Industry

If we compare this to the software producing industry these questions sound familiar and they can easily be paraphrased. The first might read as: how to distribute knowledge about huge bodies of existing code to all members of project teams in order to raise the overall productivity? Secondly, changing of rules can be interpreted as a permanent change of customer specifications resulting in an endless number of engineering change requests. Inherent to this lingers the demand for "faster, better and cheaper", which is known as well from other robotics customers like NASA[11] as from other big SW producing companies. Thirdly the life cycle of the project usually lasts longer than the individual career of a single project engineer. The goal of the whole project will be endangered if the team of programmers lack cooperation. Solving a large problem in common constitutes also a problem amongst *humans*.

1.3 Experience Matters

Both authors have a long experience of more than six years in different RoboCup teams. They have extracted the problems described in this article from experience and conversations with many other teams. The authors thank the openness of members of other teams. It made them realize that many teams face very similar problems. No teams are mentioned specifically because the nature of the problems is a general one. It suggests to investigate all viable solutions and especially to look for successful techniques from software engineering. It is important to write down these experiences for the next generations.

This article is structured as follows: in the next section a problem description is given. It suggests a mapping to some solution. Section 3 constitutes the core of this paper, it defines the favored SW engineering method and reasons why it is very well suited for this problem domain. It also indicates some open problems that are partially addressed but need a more complete solution in near future. Then follows the results on actually applying the suggested methods in running RoboCup projects and closes with some outlook on future extensions.

2 Problem Description

2.1 The Development Process

In late 2003 it was decided to take a move from the RoboCup team formerly known as GMD-Musashi, being rooted at Fraunhofer research institute AIS[6], to become an internship project only operated by students in a masters program at the University of Applied Sciences of Bonn-Rhein-Sieg. During this migration process we faced a number of problems.

Facing the blend of some very elaborate and difficult to program problems students easily get overwhelmed. To keep them motivated one can use early successes in spite of the complexity of the underlying task. Traditional SW production models, such as the waterfall or spiral model, have a too slow turnaround for this, so we chose eXtreme Programming [7] (XP) as an underlying programming paradigm. It allows a jump start to code production and has a high promise of early success. Secondly we introduced a visual programming suite called 'Iconnect'[10]. It contains many different kinds of either standard or user defined modules dedicated to signal processing tasks in real-time systems. The programming paradigm is a synchronous data-flow architecture which eases the programming tasks for a robot very much. Although the first steps are simple, the library reaches all the way up to vision algorithms like scaling, clipping skeleton building, smoothing kernel filters and Lens calibration.

To foster collaboration, trust in the whole code has to be generated. This is established through extended automated testing, so called test case driven design or unit testing [7]. Every student is allowed or even encouraged to change the code of other group members, but to generate the necessary trust to dare so the changed code has to be accompanied by many test cases. The positive completion of all tests ensures that the functionality of the code remained invariant. Collective code ownership is essential and tightly bounded teams becomes possible. These procedures allow for micro architectural code transformation also known as refactoring [9].

2.2 RoboCup

The use of standard platforms, such as the VolksBot[1] and the Aibo[2], eases the development process in RoboCup. There can be a substantial waste of time in the maintenance of shaky robots being constructed from far too many parts, which can be tamed by using these basic robots. This change yields an encapsulation of all micro-controller related issues, so the SW team can concentrate on other issues, sometimes called high level SW. The knowledge of an expert is in the micro-controller, ready to be used.

A good choice is to drastically cut down on the vast many number of choices in the design space. Sacrificing here pays off in a much higher productivity, see section 4. By prescribing a VolksBot as HW and Iconnect as the low level SW exchange of modules becomes possible. Prescribing the SW development methodology of XP adds many advantages which RoboCup seems to demand. It supports small teams optimally with a range of techniques, without giving too much overhead. One integral part of XP, the overall testing of the complete behavior system, is difficult to automate completely for the given case of behavior based robot control programs. But generally it is believed that regular unit testing leaves less than 20% of the whole SW system uncovered (our experience).

The expectations of the management may be met by a fast proof of capabilities achieved by a purchase of some robot, instead of the time-consuming process of building one. The start-up of new teams can focus on its organization and actively building up its structure. The growth and decrease of a team is no problem. Lazy students may get motivated by doing XP since it is fun to do and early running tests are tempting. Finally pair programming makes all people in the team cooperate and distributes the knowledge. This reduces what is known as the 'truck' factor in XP [7], which is the number of people that can be run over by a truck without endangering the project. If this is 1, for any part of the project, the project might be in trouble.

3 Approach

3.1 Top-Down Approach to Bottom-Up Robotics

Some of the methods of XP have to be adapted to the RoboCup environment and some do not really seem to work at all. The authors have been on a few (non-robotic) XP projects and there the technique works quite well, although it is only a starting point and not the holy grail which solves all the problems. A list of practices used or aimed for is given with some explanations why we do it and whether it works as expected or not.

Borrowed Techniques from XP

Short releases: This prevents software-drift. There is always a fully working version in CVS (or SVN). If there might be an integration problem between modules, it is detected in an early stage and easy to solve. A release happens every 4 to 6 weeks, and contains a fully working system, though it does not have all the functionality of the end-product. This works very well and motivates the team and assures the boss (professor or team-leader) that the project is on track. If it is not on track appropriate steps are undertaken without having too much damage.

Simple design: A complex design is hard to change. It is also impossible to explain to new people coming into the project. It slows the speed of development and discourages the exploration of alternative solutions. Worst case scenario is a project that has virtually come to a full stop, nobody dares to change much in the code and programming on the robot means, in practice, that the person is mostly debugging.

Testing: An essential feature to get more certainty that the complex system will actually work. If a test fails because something has changed the mistake is easily found and fixed. The software grows and if it is uncertain if the basic systems work as expected or not, there is no way to predict the behavior of the software in the future.

Pair programming: Pair programming contains immediate code refereeing and learning from each other without too much explicit training. One of the problems is that in a too small group there are several specialists working and a pair can quickly turn into one code-warrior and one viewer without learning.

Collective code ownership: Everybody is allowed to change everything. Waiting (hours, days) for someone to change a piece of code which could be fixed in minutes by oneself is frustrating, slows down progress and gives rise to friction in the team.

Continuous integration: To be certain that errors or incompatible modules do not occur, continuous integration is an essential part of a professional agile way of working. Persons can work for long periods of time on their own island only to find out that during integration something very important was very different than expected.

On-site expert: In XP an on-site customer is preferred as the expert on what the end-product should look like. In scientific projects as RoboCup this is more difficult and usually there is no customer. An on-site expert is recommended to keep the group on track. Questions about algorithms, architectures and planning issues are quickly resolved.

Steering: A complicated project has to be steered. In XP it is compared to driving a car: one cannot point the nose into a certain direction and pay no more attention to the driving. Steering is done continuously with small adjustments all the time. This might result in uncertainty because there is no grand/final plan while working. On the other side, such a plan is usually adjusted many times and only gives superficial certainty and a lot of overhead.

Coding Standards: To be able to read all the code as if it was your own, coding standards are needed. Today this is often an automated procedure in the programming environment.

Coaching: The coach aids persons with the adopted way of working. Frictions will arise due to different working habits, and the coach eases the transfer from a naive approach to a structured one. The coach is not the bogyman, but usually talks in general terms. The coach does not punish a team member if something goes wrong but is instead looking how to solve the problem. Nobody is to blame, instead everybody works on the solutions.

Strategic vs. implementation decisions: A big difference exists between these sort of decisions. They can be made by the same persons but it is good to separate them and explain explicitly what sort of decision is being made. Inexperienced persons can make (some) implementation decisions, but only the experts in the team can make the strategic decisions regarding overall architecture, the algorithms to be used and hardware changes for example.

Practices from Experience in Robot Projects

Self-monitoring of the robots: Together with reliable software one should pay attention to reliable hardware. If one is programming a behavior and the robot starts to shake after running the code, the first thing to do is check the new code. This reasoning is not always valid and can be plainly wrong. There are many possible problems with the hardware and software. Fully automated checks of the important systems should give the user a hint whether it is the new code causing the problems or something else. The monitoring can be automated and on-line, which decreases debugging time. It also helps with the control of a robot, if it can diagnose by itself what is wrong.

Round-trip engineering: Build modules and behaviors according to the specifications, test them in simulation, transfer them to the real robot and test them again. If the result is not close enough to what was expected the simulation has to be adapted or the implementation rechecked. A research question is how to automate the simulator tuning.

Active project management: Most of the people working on RoboCup are students. They join the team for a certain amount of time and leave the project. This is the ideal recipe for a failing project. Most of the knowledge leaves with the students and the new students have to learn everything anew. This causes a lack of progress on the long run.

3.2 Tools

Standardized hard- and software Preferably all the hard- and software should be of-the-shelf. The robot used is a commercial product, unless the robot itself is the research topic. The setup of the robot is standardized. One of the tools we build is an installer for the robot and development software, an automatic update procedure (one(!) button) for the drivers and the fully automated control software update procedures. *Every time* one of these buttons is pressed somewhere between five minutes (driver update) and a few hours (complete installation) is saved. Because it is easy to update the software it is more likely that during the games all the robots have the same version running.

Just a few years ago a standardized, of-the-shelf, component based system was lacking[14] and one of the aims of the robotic community. The standardization of the hardware is progressing appropriately, though on the software side it is still lacking. The approach used in modern software engineering is a visual based, modularized and agile[4] one. The visual approach forces to build modules and to standardize all the components. It becomes easier to test the software. To cite Manuela Veloso, who talked in Padova about RoboCup, she was "getting bored by the lack of progress", it was "time to do something new", the RoboCup community should "surprise her" and "be less conservative". The authors think it is due to the approach used in the different RoboCup teams that the RoboCup community is showing a slow-down, and not due to technological incompetence or another reason. The lack of progress is a social problem, not a technological one!

4 Results

The results of the proposed management approach are convincing but hard to convey. The development speed is very high while the work pressure remains constant. There was a steep learning curve due to the new robot, a new and unknown software tool and a new way of working (visual based and XP). To some of the problems we encountered we have some solutions. For the simple design the solution we use is extreme modularity. Everything we do is programmed in modules (in C++). The interfaces are rigidly defined and strong typing is used. In case of the visual programming environment, the graph built in this way *is* the architecture. It supports hierarchy in cases where it is necessary. This ensures flexibility and design overview. No design documents are used as the design *is* the real-time system itself, in case of the visual tool.

The problems with pair-programming is solved by getting more persons on the project. To be certain that modules are not messed up by unexperienced programmers we work on the problem together with the expert if the change is rather large. The on-site expert works really well. The rapid feedback of the expert ensures that if a mistake is made or a problem has to be solved it is done very swiftly. The steering done by the expert helps to keep the project on track. The coaching eases communication in the team. Nobody has a special place and everybody is treated as equal, which aids in the cohesion of the team and speeds up the development process. The approach of dividing the strategic and implementation decision is used to the maximum and it is a good working practice. Students get certainty about the project and leaders are assured no big mistakes are being made.

The estimated speed-up is hard to measure. In the mid-size league, a period of just a few months, with a handful of people working a couple of days a week resulted in our case in a working team that got through the first rounds. We lost to the European Champion and the second on the world ranking list of that year (2004), which is not a bad result for such a short period of working. In comparison, other projects of us, without the methodologies written down in this article, took one-and-a-half year *(three times as long)* to get the same quality.

5 Future Developments

In the Netherlands a research project is ongoing which tests these methodologies between research groups. Questions arise about how it works when dozens of people are working on a robotic project, in half a dozen different research groups. One group is researching how to make a 'virtual laboratory'. How can we work in different physical places and still have a decent project. One extra practice is a gate keeper on the versioning system (CVS/SVN). There can be many unstable branches, where groups or individuals can work, but there is only *one* stable branch. This branch has a gatekeeper who checks whether the code is according to the standards. If not then it does not go into the stable branch. Another method is nightly updates and builds. All the unstable branches are updated with the stable branch code. If merger problems arise it is reported in the daily morning mail. Also all the branches are compiled from scratch, to check for problems. Everything is automated and the system emails one email in the morning to everybody on the email list with all the details.

6 Summary

In this article we presented the application of a recent SW development technique called eXtreme Programming to the realm of RoboCup. Some of the techniques do not seem applicable to working with robots, but other one are working out very fine. The result is a team that is expandable and has a high development speed. We advise other groups also to pay attention to the social aspects of complex high-tech projects such as RoboCup. This might be the first steps toward the maturation of the robot-industry.

References

1. http://www.volksbot.de.
2. http://www.us.aibo.com.
3. http://www-2.cs.cmu.edu/tekkotsu/.
4. Special report: Software goes extreme.
5. A. Bredenfeld, Th. Christaller, H.Jaeger, H-U. Kobialka, P. Schöll. Robot behavior design using dual dynamics. Technical report, GMD Report, 2000.
6. A. Bredenfeld, V. Becanovic, Th. Christaller, I. Godler,G. Indiveri, K. Ishii, J. Ji, H-U. Kobialka, N. Mayer, H. Miyamoto, A.F.F. Nassiraei, P-G. Plöger, P. Schöll, M. Shimizu . Ais-musashi team description paper.
7. Kent Beck. *Extreme Programming Explained.* Addision Wessley, 2000.
8. Silvia Coradeschi and Jacek Malec. How to make a challenging ai course enjoyable using the robocup soccer simulation system. In *RoboCup-98: Robot Soccer World Cup II.*
9. Martin Fowler. *Refactoring: Improving the Design of Existing Code.* Addision Wessley, 1999.
10. Roland Mandl and Bernhard Sick. *Messen, Steuern und Regeln mit ICONNECT.* Vieweg, 2003.
11. Howard E. McCurdy. *Faster, Better, Cheaper: Low-Cost Innovation in the U.S. Space Program.* Johns Hopkins Univ Press, 2001.
12. Peter Stone. Robocup as an introduction to cs research.
13. Peter Stone. Multiagent competitions and research: Lessons from robocup and tac. 2002.
14. B. Werger. Ayllu: Distributed port-arbitrated behavior-based control, 2000.

Mobile Robot Communication Without the Drawbacks of Wireless Networking

Andreas Birk and Cosmin Condea

School of Engineering and Science
International University Bremen
Campus Ring 1, D-28759 Bremen, Germany
a.birk@iu-bremen.de

Abstract. The default solution for mobile robot communication is RF-networking, typically based on one of the IEEE 802.11 standards also known as WLAN technology. Radio communication frees the robots from umbilical cords. But it suffers from several significant drawbacks, especially limited bandwidth and range. The limitations of both aspects are in addition hard to predict as they are strongly dependent on environment conditions. An outdoor RF-link may easily cover 100m over a line-of-sight with full bandwidth. In an indoor environment, the range often drops to a few rooms. Walls made of hardened concrete even completely block the communication. Driven by a concrete application scenario where communication is vital, namely robot rescue, we developed a communication system based on glassfibre links. The system provides 100MBit ethernet connections over up to 100m in its default configuration. The glassfibres provide high bandwidth, they are very lightweight and thin, and they can take a lot of stress, much more than normal copper cable. The glassfiber links are deployed from the mobile robot via a cable drum. The system is based on media converters at both ends. One of them is integrated on the drum, thus allowing the usage of inexpensive wired sliprings. The glassfibre system turned out to be very performant and reliable, both in operation in the challenging environment of rescue robotics as well as in concrete experiments.

1 Introduction

Though there is some work where even cooperative robots are investigated without communication [Ark92], there are hardly any application scenarios where mobile robots can really operate without being networked. The common technical solution is the IEEE 802.11 family of standards also known as WLAN [OP99]. WLAN has its well-known limitation [PPK^{+}03], especially in respect to bandwidth and range. One simple remedy is to use mobile robots to act as relay stations along a kind of bucket brigade [NEMV02, NPGS03]. In doing so, the relay robots follow a lead robot and they stop when the communication chain is threatened to be broken. The big disadvantage is that rather many robots are needed to cover extended areas and that the majority of the robots is used for nothing but as a communication relay. More complex variations of the relay idea

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 585–592, 2006.

are investigated in the field of ad-hoc networking [Per00] where dynamic links and routing protocols are employed [JMH04, RT99, JW96].

In addition to the severe limitations in respect to bandwidth and reliability, wireless communication solutions have the significant drawback for rescue applications that they block parts of the precious RF space. Rescue missions involve different groups of first responders like firebrigades, police, medical doctors, and so on. Each of these groups has their own communication systems. For many large scale disasters like earthquakes, there are even many of these groups from different countries, each with its own type of communication equipment. The coordination of the usage of RF bands is a known problematic issue at disaster sites. Any additional system like a rescue robot will face difficulties of acceptance if it will block parts of this scarce resource with its wireless network.

Here, an extremely simple new approach is taken that circumvents the core underlying troublemaker, namely the usage of RF as medium for mobile robot communication. Instead, glassfibres are used. Being a cable based medium, the challenge is to find a suited approach to deploy the cables during operation by the robot itself. For this purpose a low-cost cable-drum system was developed, which has proven to be very versatile and stable.

The rest of this paper is structured as follows. Section 2 gives an overview of the system. Experiments and results are presented in section 3. Section 4 concludes the paper.

2 System Overview

As mentioned before, the quality of RF-communication strongly depends on environmental conditions. We are interested in a particularly harsh domain, namely rescue missions where robots are operating in urban disasters scenarios ranging from earthquakes to gas or bomb explosions [RMH01, Sny01]. Distortions or even complete failure of RF-communication is a known problem in the according

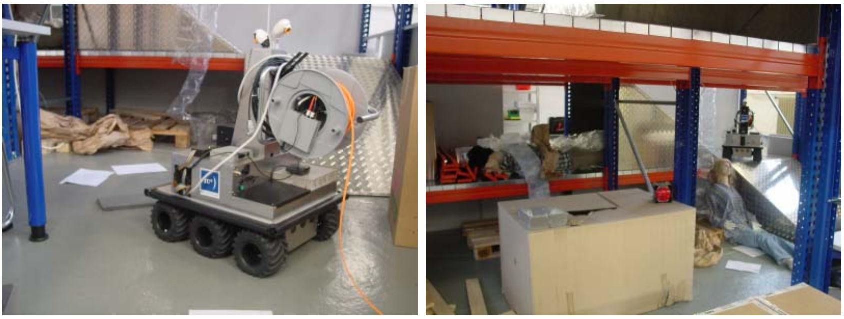


Fig. 1. A rescue robot with the glassfibre drum on its back (left). It has to operate in an environment where high mobility is needed and the glassfibres are experiencing a lot of stress through obstacles (right). Nevertheless, the glassfibres never failed in over two years of operation.

scenarios when RF-transceivers are moved through partially or fully damaged buildings. In addition, RF bands are a scarce resource, which may only be used under very special permissions at large scale disasters like earthquakes. We therefore developed an alternative solution to RF to allow for high bandwidth communication of mobile devices.

The goal of the IUB rescue robots team is to develop fieldable systems within the next years. Since the beginning of its research activities in this field in 2001, the team has participated in several RoboCup competitions to test its approaches [Bir05, BCK04, BKR$^+$02]. In addition to work on mapping [CB05] and adhoc-networking [RB05], the development of the robots themselves based on the so-called CubeSystem [Bir04a] is an area of research in the team [BKP03, BK03].

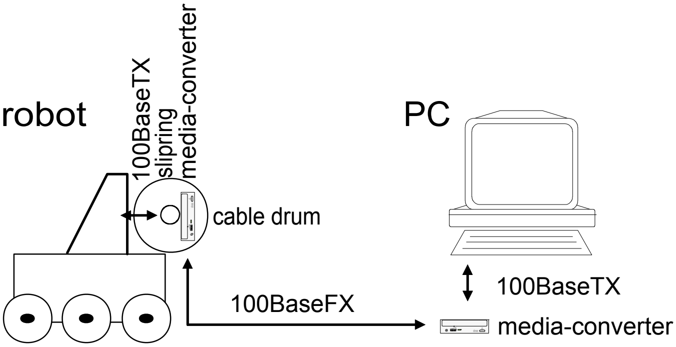


Fig. 2. The components of the deployable glassfibre communication system. The overall system behaves much like a standard 100BaseTX FastEthernet connection between the robot and an endpoint like a PC (cross-cable connection) or a network bridge (straight-cable connection).

Figure 2 shows the main components of the overall system:

- two Allied Telesyn AT-MC100 media converter
- one IDM Electronics H6 slipring
- 30m to 100m of monomode glassfibres

Glassfibres are preferable over copper as cable medium for several reasons. First, they are lightweight. For our application purposes in the order of a factor two to three when compared to copper cable. Second, the bandwidth/distance parameter is much higher [YZ01]. Third, glassfibers are much less vulnerable to physical stress than normal CAT5 cables. This holds especially in respect to the minimum bending radius. This parameter is of quite some importance in application scenarios where high agility is a must. This robustness of the glassfibres in our system has been proven in uncountable testruns of our robots in the IUB rescue arena [Bir04b] as well as in various RoboCup competitions including RoboCup 2003 in Padua, RoboCup American Open 2004 in New Orleans, RoboCup 2004 in Lisbon and the RoboCup German Open 2005 in Paderborn.

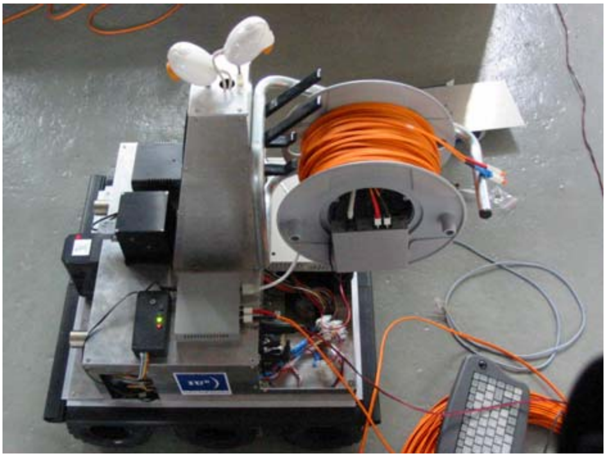


Fig. 3. A rescue robot with the newly designed cable deployment system

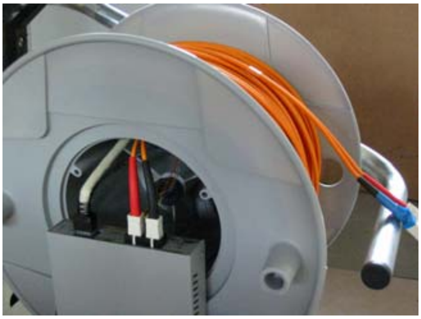


Fig. 4. A close-up of the cable deployment system

In none of the testruns or competitions did the glassfibre based communication system ever fail.

As the cable is to be deployed from the robot, a rotating joint is required. The usage of very expensive optical sliprings could be prevented by a simple trick, namely using standard media converters. On both end points of the communication system, 100BaseTX FastEthernet connections via a standard RJ45 connector are provided. 100BaseTX uses Category 5 cabling, or simply Cat5. Cat5 is a type of cable designed for high signal integrity - it is tested to insure a clean transmission of 100Mhz signals. The size of each wire is 22 gauges and each pair of wires is twisted within the exterior cladding, thus the name "twisted pair" which refers to this type of cabling. As there is no shielding around the four twisted pairs, Cat5 is generally referred by the term "unshielded twisted pair", or simply UTP.

100BaseTX uses only two of the four available pairs of UTP cable. One pair(TX) is used for transmission and the other(RX) is used for reception. The

TIA-568B wiring standard defines the color-coding and, most important, the order of wires' connection in a RJ-45 8-pin modular jack. The so-to-say spare wires on the cable are used in our system to power the media converter on the cable drum.

So, the overall system mainly consists of a conventional 100BaseFX glassfibre communication part, which is converted at both end points to a 100BaseTX copper cable. The unconventional part is the 100BaseTX link on the robot, which connects its network card with the media converter on the cable drum via a wire slipring.

3 Performance of the Cabledrum

The potentially error-prone part of our system is the unconventional 100BaseTX cabling involving a lowcost slipring. There are several crucial parameters for CAT5 cable, namely

– *Attenuation* is the decrease in signal strength along the transmission line. Since digital signal processing cannot significantly compensate for signal degradation, ensuring low levels of attenuation is crucial.
– *Attenuation to crosstalk ratio(ACR)* is the difference between attenuation and near-end crosstalk(NEXT). ACR is a crucial calculation with regard to network transmissions. Its positive values ensure that a signal transmitted along a UTP cable is stronger than near-end crosstalk.
– *Near-end crosstalk(NEXT)* measures the undesired signal coupling between adjacent pairs at the transmit end.
– *Far-end crosstalk(FEXT)* measures the undesired signal coupling among adjacent pairs at the receive end.
– *Equal level far-end crosstalk(ELFEXT)* is obtained by subtracting attenuation from the far-end crosstalk. Poor ELFEXT levels can result in increased bit error rates and/or undeliverable signals.
– *Propagation delay* is the amount of time the signal travels from the transmit end to the receive end.
– *Delay skew* represents the difference between the pair with the highest propagation delay and the pair with the lowest propagation delay.

There exist very strict limitations for these parameters [KBs, Ryb99]. As the low level electrical properties of the slipring are neither documented nor easy to measure, it is hence necessary to make a more high level investigation of the properties of this link. Note that the usage of standard network test equipment is not necessary helpful as the low level parameter of the slipring strongly depend on its mode of operation, i.e., on its rotation rate. So, there is the need to evaluate the deployable glassfibre system in respect to its compliance with the ethernet standard.

The study of network transmission quality is a significant field of research dealing with various metrics [Dre02, Fer90, Dre03]. Here, we simply measure the round trip times of network packets to test the quality of the cabledrum

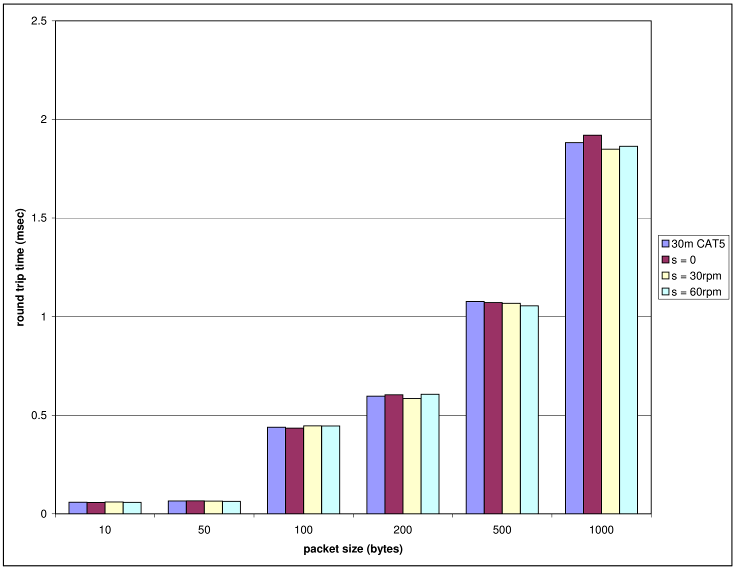


Fig. 5. The average round trip times (RTT) of 1 million packets measured over a standard 30 meter CAT5 cable as well as over the IUB cabledrum rotating at no (s = 0 rpm), medium (s = 30 rpm) and high speed (s = 60 rpm). The RTT only depend on the packet size and slight random variations. No significant differences between the standard cable and the drum rotating at different speeds can be measured.

system. To measure the potential problems caused by the slipring, the short 100BaseTX link from the robot card to the media converter via the slipring is compared to a 30m standard CAT5 cable, i.e., a standard medium length copper cable compliant to 100BaseTX. The cabledrum is furthermore rotated at different speeds to test whether the transmission quality is influenced by this. As shown in figure 5, a difference between the standard patch cable and the cabledrum can not be measured in any of the experiments. This holds in respect to reliability, which is always perfect with 0% packet loss, as well as in respect to average round trip times (shown in figure 5) and jitter. The overall system with 100m glassfibres behaves thus like a direct 100BaseTX FastEthernet connection between the robot and an endpoint with completely neglectable delays from the media converters.

4 Conclusion

When it comes to mobile robot communication, wireless networks are the overwhelming standard. We have shown that a cable based approach can be a serious alternative, especially for RoboCup Rescue. A deployable glassfibre system was presented which has proven to be reliable in the field as well as in experiments. Glassfibres are lightweight, thin, and very robust. Furthermore, they carry high data rates with low delay and high reliability. The need for an expensive optical slipring is circumvented in our system by using media converters. One of the converters is on the cable drum. This allows a 100BaseTX copper wire connection from the robot's network card via a simple wire slipring to this media converter.

References

[Ark92] R. C. Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[BCK04] Andreas Birk, Stefano Carpin, and Holger Kenn. The IUB 2003 rescue robot team. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer, 2004.

[Bir04a] Andreas Birk. Fast robot prototyping with the CubeSystem. In *Proceedings of the International Conference on Robotics and Automation, ICRA'2004*. IEEE Press, 2004.

[Bir04b] Andreas Birk. The iub rescue arena, a testbed for rescue robots research. In *Second IEEE International Workshop on Safety, Security, and Rescue Robotics, SSRR'04*, 2004.

[Bir05] Andreas Birk. The IUB 2004 rescue robot team. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*. Springer, 2005.

[BK03] Andreas Birk and Holger Kenn. A control architecture for a rescue robot ensuring safe semi-autonomous operation. In Gal Kaminka, Pedro U. Lima, and Raul Rojas, editors, *RoboCup-02: Robot Soccer World Cup VI*, volume 2752 of *LNAI*, pages 254–262. Springer, 2003.

[BKP03] Andreas Birk, Holger Kenn, and Max Pfingsthorn. The iub rescue robots: From webcams to lifesavers. In *1st International Workshop on Advances in Service Robotics (ASER'03)*. 2003.

[BKR$^+$02] Andreas Birk, Holger Kenn, Martijn Rooker, Agrawal Akhil, Balan Horia Vlad, Burger Nina, Burger-Scheidlin Christoph, Devanathan Vinod, Erhan Dumitru, Hepes Ioan, Jain Aakash, Jain Premvir, Liebald Benjamin, Luksys Gediminas, Marisano James, Pfeil Andreas, Pfingsthorn Max, Sojakova Kristina, Suwanketnikom Jormquan, and Wucherpfennig Julian. The IUB 2002 rescue robot team. In Gal Kaminka, Pedro U. Lima, and Raul Rojas, editors, *RoboCup-02: Robot Soccer World Cup VI*, LNAI. Springer, 2002.

[CB05] Stefano Carpin and Andreas Birk. Stochastic map merging in rescue environments. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*, page p.483ff. Springer, 2005.

[Dre02] F. Dressler. QoS considerations on IP multicast services. *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, and Medicine on the Internet (SSGRR 2002w), L'Aquila, Italy*, 2002.

[Dre03] F. Dressler. A metric for numerical evaluation of the QoS of an internet connection. *Proceedings of 18th International Teletraffic Congress (ITC18)*, 5b:1221–1230, 2003.

[Fer90] D. Ferrari. Client requirements for real-time communication services; RFC-1193. *Internet Request for Comments*, (1193), 1990.

[JMH04] D. B. Johnson, D. A. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr), July 2004. IETF Internet Draft, draft-ietf-manet-dsr-10.txt.

[JW96] D. B. Johnson and D. A. Waltz. Dynamic source routing in ad-hoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[KBs] http://www.xs4all.nl/ sbp/knowledge/concab/utpether.htm.

[NEMV02] H. G. Nguyen, H. R. Everett, N. Manouk, and A. Verma. Autonomous mobile communication relays. In *SPIE Proc. 4715: Unmanned Ground Vehicle Technology IV*, 2002.

[NPGS03] H. G. Nguyen, N. Pezeshkian, M. Gupta, and J. Spector. Autonomous communication relays for tactical robots. In *11th Int. Conf. on Advanced Robotics (ICAR 2003)*, 2003.

[OP99] Bob O'Hara and Al Petrick. *The IEEE 802.11 Handbook: A Designer's Companion.* Standards Information Network IEEE Press, 1999.

[Per00] C. E. Perkins. *Ad Hoc Networking.* Addison Wesley Professional, 2000.

[PPK$^+$03] Jin-A Park, Seung-Keun Park, Dong-Ho Kim, Pyung-Dong Cho, and Kyoung-Rok Cho. Experiments on radio interference between wireless lan and other radio devices on a 2.4 ghz ism band. In *The 57th IEEE Semiannual Vehicular Technology Conference*, volume 3, pages 1798 – 1801, 2003.

[RB05] Martijn Rooker and Andreas Birk. Combining exploration and ad-hoc networking in robocup rescue. In Daniele Nardi, Martin Riedmiller, and Claude Sammut, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages pp.236–246. Springer, 2005.

[RMH01] M. Micire R. Murphy, J. Casper and J. Hyams. Potential tasks and research issues for mobile robots in robocup rescue. In Tucker Balch Peter Stone and Gerhard Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Lecture keywordss in Artificial Intelligence 2019. Springer Verlag, 2001.

[RT99] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[Ryb99] Valerie Rybinski. De-mystifying category 5, 5e, 6, and 7 performance specifications. *The Simeon Company*, 1999. http://www.siemon.com/us/whitepapers/99-12-17-demystifying.asp.

[Sny01] Rosalyn Graham Snyder. Robots assist in search and rescue efforts at wtc. *IEEE Robotics and Automation Magazine*, 8(4):26–28, December 2001.

[YZ01] Shakil Akhtar Youlu Zheng. *Networks for Computer Scientists and Engineers.* Oxford University Press, December 2001.

Mosaic-Based Global Vision System for Small Size Robot League

Yuji Hayashi, Seiji Tohyama, and Hironobu Fujiyoshi

Dept. of Computer Science, Chubu University, Japan
yuji@vision.cs.chubu.ac.jp, sei@vision.cs.chubu.ac.jp, hf@cs.chubu.ac.jp
http://www.vision.cs.chubu.ac.jp/

Abstract. In the RoboCup F180 Small Size League, a global vision system using multiple cameras has been used to capture the whole field view. In the overlapping area of two cameras' views, a process to merge information from both cameras is needed. To avoid this complex process and rule-based approach, we propose a mosaic-based global vision system which produces high resolution images from multiple cameras. Three mosaic images, which take into account the height of each object such as our robots, opponent robots, and the ball on the field, are generated by pseudo corresponding points. Our system archives a position accuracy of better than 14.2 mm(mean: 4 mm) over a 4 × 5.5 m field.

1 Introduction

Recently, a global vision system using multiple cameras has been used in the RoboCup F180 Small Size League(SSL), since the field size was changed to 5,500 × 4,000mm to create more space. In the case of using two cameras as a global vision system, one is mounted over each half of the field to capture the image which has enough resolution for object recognition. However, there will be some problems with the use of two cameras. For example, in the overlapping area of both cameras' views, information from both cameras should be merged; however, this is considered a very complex process.

A possible solution to avoid this complex process is to employ a planar perspective transform known as "image mosaicing," which generates a high resolution image from two images. To obtain the mosaic image, a homography between a reference image and the other image is computed by correspondences. Although the mosaic image is generated from two camera images, the vision algorithm already in use in the SSL can be easily applied to the mosaic image without any changes. And any additional process such as merging information from both cameras is not needed. Since the mosaic image is registered as a planar image, there will be a blur around the object such as a robot, when the object has a height from the plane used for calculating the homography. This causes errors in the object identification, which processes regions to find the ball and robots and identifies our robots.

In this paper, we propose a mosaic-based global vision system, which generates mosaic images taking into account the height of each object. We will show that our system is capable of high accuracy in position estimation.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 593–601, 2006.

This paper is structured as follows. The second section points the problems in the use of a two camera system. The next section describes the proposed system in detail. Section 4 discusses the experimental results in position estimation and processing time. Section 5 concludes the paper.

2 Problems with a Two Camera System

In order to capture the image which has a pixel representiong 5 millimeters on the field, one is camera mounted over each half of the field as shown in Figure 1. Each camera image is processed by color segmentation and followed by object identification. Finally, the object's position is converted to real world coordinates for controlling the robots. After the vision processes described above, merging information from both cameras is required, because of the overlapping area of the two camera images. To merge results from both cameras in the overlapping area, the following four methods are employed.

A. Updating recent result. When a recent result is obtained from either camera, the final decision of the robot's position is updated.

B. Hard decision boundary. The boundary for each camera is decided manually in advance the final decision of the robot's position is updated if the position is inside the boundary.

C. Hysteresis. When a robot enters the overlap area, the camera that has been tracking the robot continues to track it, and the other camera ignores the data.

D. Fusion. Estimated positions by both cameras are merged to world coordinates by weighting.

In method A, C, and D, robot1 and robot2 on the camera1 image shown in Figure 1 are not identified, because half of the robot is outside of one camera's image and the other half of the robot is outside of the other camera's image. The camera cannot correctly identify the robot because the marking pattern is not completely recognized in one camera's image. In method B, the boundary

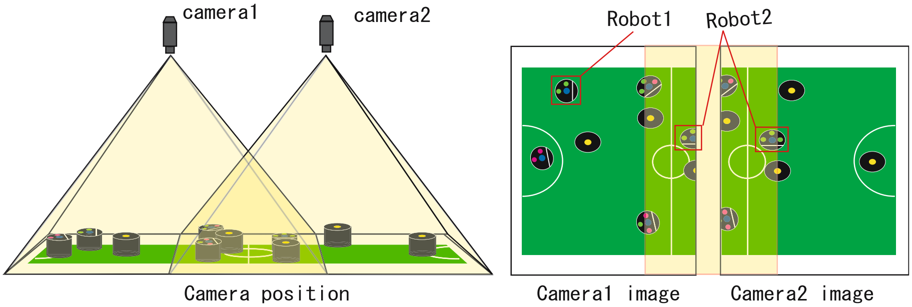


Fig. 1. Global Vision system using two cameras

requires continual adjustment when both cameras are mounted. To solve these problems, our approach generates a high resolution image from two cameras, and then processes the mosaic image by the vision algorithm which is used in a single camera system.

3 Mosaic-Based Global Vision System

To obtain the mosaic image from both cameras, a homography is computed by correspondences between the two images. Since the mosaic image is registered as a planar image, there will be a blur on the image around the object which has a height from the plane used for calculating the homography. Our approach is generating three mosaic images which take into account the height of our robot, the opponent's robot and a ball. Each mosaic image is processed by color segmentation and object identification which usually are used in the global vision system of a single camera. By generating the mosaic image considering the height of each object, a highly accurate position estimate can be obtained.

3.1 Generating a Mosaic Image of the Field Plane

Figure 2(a) shows the relationship of projective geometry between both cameras and the base plane (field). The planar perspective transform based on a homography warps an image into another image using 8 parameters of the matrix $\boldsymbol{H}$ [7] [8]. The homography between the two images of a planar surface is expressed as

$$\boldsymbol{p'} = \boldsymbol{H_1 p_1} = \begin{bmatrix} h_1 \; h_2 \; h_3 \\ h_4 \; h_5 \; h_6 \\ h_7 \; h_8 \; \; 1 \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \tag{1}$$

where $\boldsymbol{p_1} = [u_1, v_1, 1]^T$ on the camera1 coordinate and $\boldsymbol{p'} = [u', v', 1]^T$ on the mosaic image coordinate are corresponding points of two images. Mosaic image

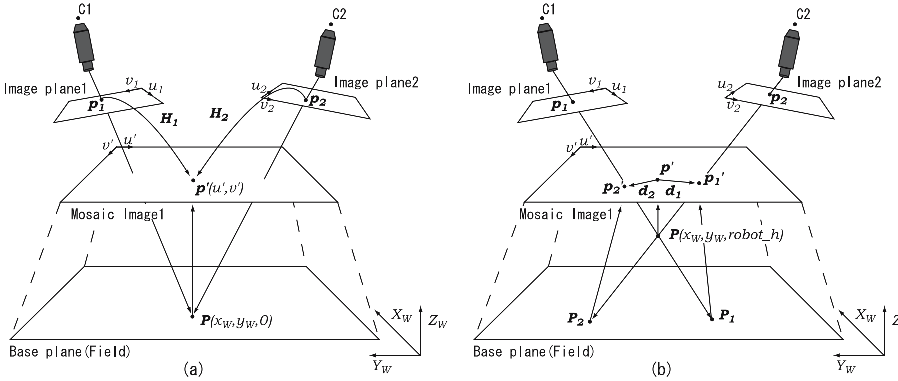


Fig. 2. Projective geometry between both cameras and a base plane

coordinate $\boldsymbol{p'}$ corresponds to $\boldsymbol{P} = [x_W, y_W, z_W]^T$ on the world coordinate. The homography $\boldsymbol{H_2}$, which projects from the point $\boldsymbol{p_2} = [u_2, v_2, 1]^T$ on the camera2 to $\boldsymbol{p'}$ on the mosaic image, is computed inthe same way $\boldsymbol{H_1}$ was calculated. These relations are expressed as the following equations:

$$\begin{aligned} \boldsymbol{p'} &= \boldsymbol{H_1 p_1} \\ \boldsymbol{p'} &= \boldsymbol{H_2 p_2} \\ \boldsymbol{P} &= \alpha \boldsymbol{p'}. \end{aligned} \tag{2}$$

Note that the mosaic image coordinate $\boldsymbol{p'}$ corresponds to the world coordinate $\boldsymbol{P}$ on the field by linear mapping. In our vision system, the mosaic image, in which a pixel represents 5 mm on the field, is generated using bilinear interpolation. Therefore, it is easy to obtain the world coordinate $\boldsymbol{P}$ from the mosaic image coordinate $\boldsymbol{p'}$ by

$$\boldsymbol{P}[\text{mm}] = 5[\text{mm/pixel}] \times \boldsymbol{p'}[\text{pixel}]. \tag{3}$$

The process for generating a mosaic image of the field is described as follows:

Step1. By Choosing the landmark points on the field plane (such as rectangle's corner), which are observed from camera1, corresponding points of $\boldsymbol{p_1}(u, v)$ on the camera1 image coordinate and $\boldsymbol{P}(X_w, Y_w, 0)$ on the world coordinate for the landmark are measured manually.

Step2. $\boldsymbol{P}$ is converted to the mosaic image coordinate $\boldsymbol{p'}$ by using equation (3).

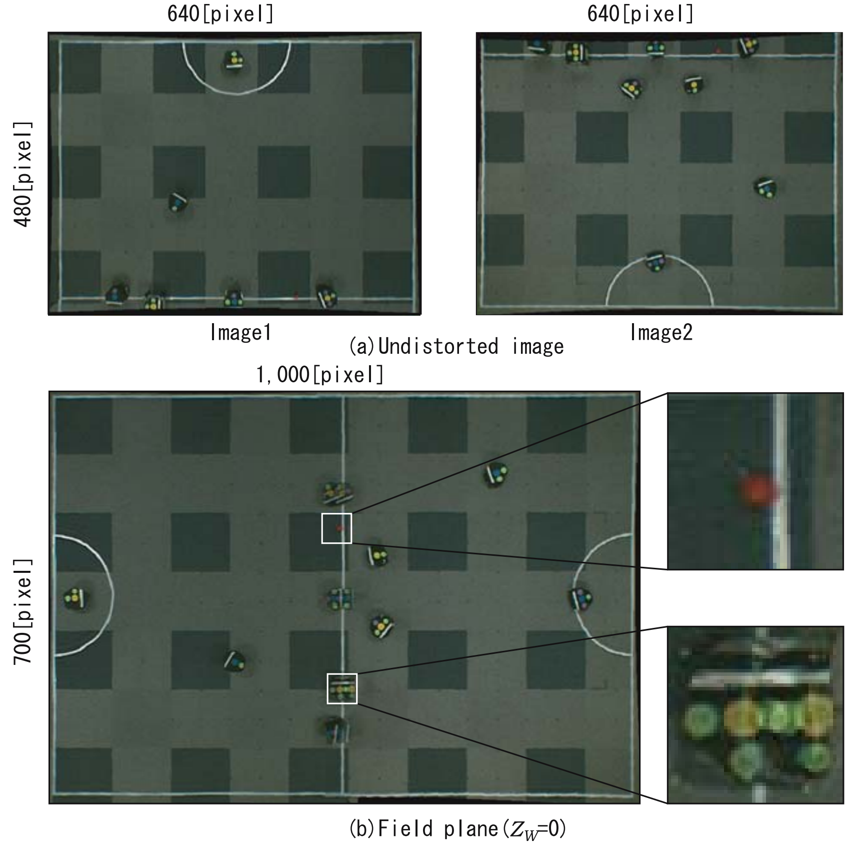
Step3. The homography $\boldsymbol{H_1}$ is computed by using the correspondences of at least 4 points. The homography $\boldsymbol{H_2}$ is also computed in the same way $\boldsymbol{H_1}$ was calculated.

Step4. Blending is performed around the area where both images are overlapped. Finally, the mosaic image is generated using $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$, from both camera images.

Figure 2(b) shows a mosaic image and both camera images. We can see that the ball on the field plane is very clear, but the markers on the top of the robot are not clear. Using the homography calculated from correspondences on the field plane, $\boldsymbol{P}$ viewed from camera1, which is located at a height of *robot_h* from the base plane (field), is projected to $\boldsymbol{P_1}$ on the world coordinate as shown in Figure 2(b). This causes errors denoted as $\boldsymbol{d_1}$ and $\boldsymbol{d_2}$ on the mosaic image as shown in Figure 2(b). For this reason, a blur shown in Figure 3(b) will be observed in the overlapping area on the mosaic image.

3.2 Generating a Mosaic Image of the Virtual Plane

Using the homography computed by correspondences on the field plane, pixels on the top of the robot on the camera image are not correctly projected to the mosaic image coordinate. In order to obtain the homography of any plane in 3D space, correspondences on the plane should be measured. However, it is impossible to measure the feature points on any plane in 3D space in a small



(a)Undistorted image

(b)Field plane(Z_W=0)

Fig. 3. Mosaic image of field plane

amount of setup time. Our approach generates pseudo feature points on the image coordinate for the top of the robot, and the homography is computed as shown in Figure 4.

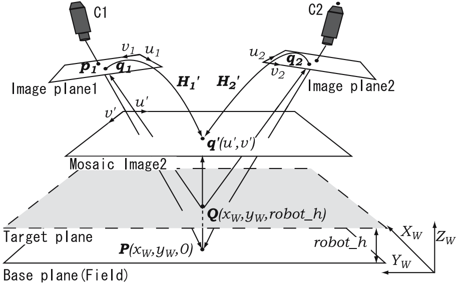


Fig. 4. Computation of homography of target plane

The process for generating a mosaic image taking into account a virtual plane is described as follows:

Step1. By choosing the landmark points on the field plane. Corresponding points of $\boldsymbol{p_1}(u, v)$ on the camra1 image coordinate and $\boldsymbol{P}(X_w, Y_w, Z_w)$ on the world coordinate are measured manually.

Step2. Measuring the height of the robot($robot\_h$), and $\boldsymbol{P}$ is converted to the point $\boldsymbol{Q}(X_w, Y_w, robot\_h)$ on the virtual plane.

Step3. $\boldsymbol{Q}$ is projected to camera 1 by reverse projection using the intrinsic and extrinsic camera parameters. The pseudo corresponding point $\boldsymbol{q_1}$ is calculated by

$$\boldsymbol{q_1} = \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \frac{1}{robot\_h'} \begin{pmatrix} fk_u & fs & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} (R \mid -RT) \begin{pmatrix} x_W \\ y_W \\ robot\_h \\ 1 \end{pmatrix} \tag{4}$$

where $\boldsymbol{R}$ is a rotation matrix, $\boldsymbol{T}$ is a translation matrix, f is a focal length, s is shearing factor, and k_u, k_v are unit length of each axis.

Step4. The homography $\boldsymbol{H_1'}$ is computed by using the correspondences of $\boldsymbol{q'}$ and $\boldsymbol{q_1}$. The homography $\boldsymbol{H_2'}$ is also computed by the same way of $\boldsymbol{H_1'}$.

Step5. Blending is performed around the area where both images are overlapped. Finally, the mosaic image taking into account the height of the object is generated using $\boldsymbol{H_1'}$ and $\boldsymbol{H_2'}$.

Figure 5 shows a mosaic image of the virtual plane which has a height of the robot ($robot\_h$ = 150 mm). Although the ball on the field plane is not clear, we

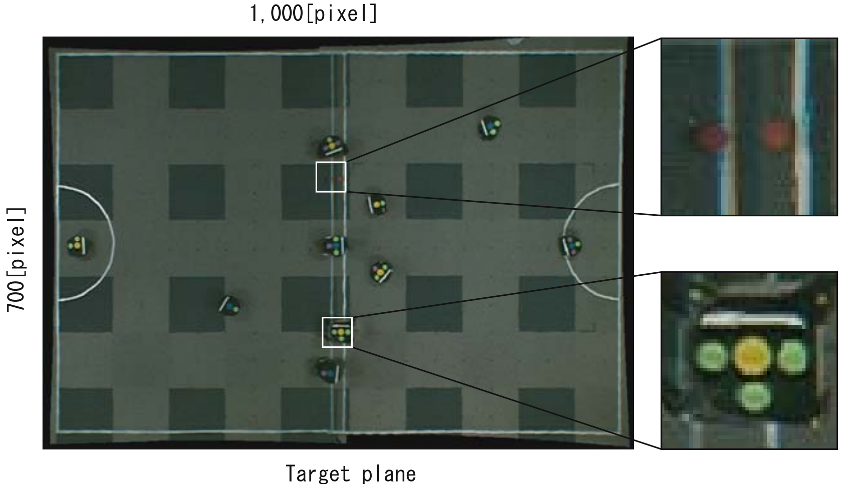


Fig. 5. Mosaic image of the virtual plane

can not see any changes around the markers on the top of the robot. We also realized that the mosaic image of the virtual plane can absorb the change in the height within ± 2cm.

The global vision system we proposed has two merits. One of them is to remove the necessity for process of merging the information from both cameras. The other is to facilitate the application of the vision algorithm which we have used commonly in single camera system such as CMVision[10] and robot identification method described in [9].

4 Experimental Results

To determine the accuracy of the proposed method, we measured the robot's position as ground truth and compared it to the position given by our global vision system.

4.1 Configuration of Vision System

Two cameras are mounted at a height of 3,000 mm, and each camera has a view of each area of 4,900 × 3,400 mm (overlapping area is 300 × 3,400 mm). Both cameras are calibrated using over 40 feature points on the field plane to estimate camera parameters.

4.2 Results

We evaluated the proposed method by location testing for 61 locations spread over the field. Table 1 shows results in regard to location of the robot (height is 150mm) by the two mosaic images of the virtual plane and the field plane. The mean of the position accuracy by the virtual plane (robot height) is 4mm. Note that a pixel represents approximately 5 millimeters on the field in our camera setting. This table shows that our vision system is able to correct to real world locations with a high degree of accuracy. The system achieves a position accuracy of better than 14.2 mm over a 4 m × 5.5 m field.

Table 1. Error of the estimated positions [mm]

	Average		SD		Max	
@	x	y	x	y	x	y
Field plane	43.5	34.3	24.9	20.7	95.7	77.1
Target plane	3.7	4.3	2.8	2.8	12.0	14.2

* SD : Standard deviation

Figure 6 shows the distribution of the estimated positions. By the mosaic image of the field, it is clear that the error becomes larger regarding the distance from the optical center of the camera as shown in Figure 6(b). On the other hand, the mosaic image of the virtual plane is able to estimate real world locations very accurately, since the mosaic image was generated by taking into account the height of the robot.

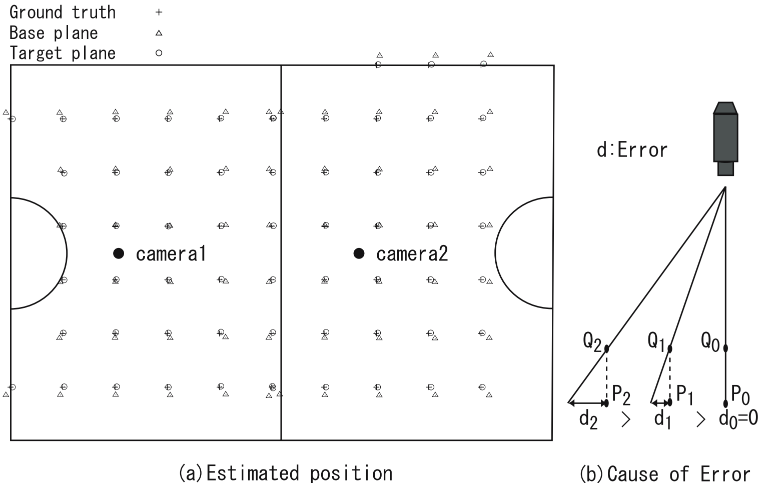


Fig. 6. Distribution of estimated positions

4.3 Processing Time

The proposed global vision system takes 81 ms for all processes including generating three mosaic images for our robots, opponent robots and field plane. (39 ms for generating three full mosaic images, 38 ms for color segmentation, and 4 ms for object identification). This processing time is not sufficient for controlling the robot in real-time using visual feed-back. To solve this problem, we generate mosaic images of only 30 × 30 pixels around each object position, which was detected in a previous frame, in order to run in real-time. The range of 30 × 30 pixels is about 15 × 15 cm in the real field. This search range works to reduce total processing time to 13 ms (generating three mosaic images takes 5.5 ms, color segmentation takes 5.5 ms, and object identification takes 1.1 ms for each process).

5 Conclusion

We proposed a mosaic-based global vision system using multiple cameras, which generates high resolution images taking into account the virtual plane of an object's height. Our system achieves a position accuracy of better than 14.2 mm(mean: 4 mm) over a 4 × 5.5 m field, and does not need any additional process of merging information from both cameras.

References

1. David Ball, Gordon Wyeth and Stephen Nuske, "A Global Vision System for a Robot Soccer Team", Proceedings of the 2004 Australasian Conference on Robotics and Automation (ACRA), Canberra, Australia, 2004.

2. Anna Egorova, Alexander Gloye, Cuneyt Goktekin, Achim Liers, Marian Luft,Raul Rojas, Mark Simon, Oliver Tenchio, and Fabian Wiesel, "FU-Fighters Small Size 2004", RoboCup 2004 Symposium, Small Size League Team Description, 2004.
3. Ng Beng Kiat, "LuckyStar 2004", RoboCup 2004 Symposium, Small Size League Team Description.
4. David Ball, Gordon Wyeth, "UQ RoboRoos 2004:Getting Smarter", RoboCup 2004 Symposium, Small Size League Team Description, 2004.
5. R. Y. Tsai. "A verstile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the Shelf TV Cameras and Lenses", In IEEE Journal of Robotics and Automation, Vol.RA-3, Num.4, pp. 323-344, 1987.
6. Z. Zhang. "A flexible new technique for camera calibration." IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.
7. R. Szeliski, "Image mosaicing for tele-reality applications", Rroc.IEEE Workshop on Applications of Computer Vision, pp.44-53, 1994.
8. Hartley A. and Zisserman A. "Multiple View Geometry in Computer Vision, Cambridge University Press", 2000.
9. Shinya Hibino, Yukiharu Kodama, Yasunori Nagasaka, Tomoichi Takahashi, Kazuhito Murakami, and Tadashi Naruse, "Fast Image Processing and Flexible Path Generation System for RoboCup Small Size League", RoboCup2002, pp.53-64, 2002.
10. James Bruce, Tucker Balch and Manuela Veloso, "Fast and Inexpensive Color Image Segmentation for Interactive Robots", In Proceedings of IROS-2000, Japan, October 2000.

Multi Power Multi Direction Kicking System

Mohammad Azim Karami

Department of electrical & computer engineering
Shahid Beheshti University, Tehran-Iran
azimkarami@gmail.com

Abstract. In this paper a multi power multi direction kicking system is developed. This versatility is achieved without a need of either changing the power supply of robot or the direction of whole chassis of robot .The main factors for designing a suitable solenoid were studied and the casual agents to obtain maximum velocity of ball were introduced. A main circuit is used for getting different currents for solenoid to have different power of shooting. An arrangement of solenoids is introduced to have multi direction kicking system.

1 Introduction

One of the main parts of robots which play in middle and small size leagues of Robocup matches is Kicker system. This system must kick the ball when a specific control command comes from processor of robot. There are usually two main different systems used for ball-kicking in robots.

The first system consists of a couple of rode which rotate about an axis for kicking the ball. These rods get their force from a DC motor, controlled by the control unit of robot. The second system, used in mentioned type of robots is a solenoid system [1],[2],[3]. When a specific command comes from the control unit, a large amount of electrical current flows in solenoid windings and by producing a strong magnetic field, the bar will move. Almost all robots which designed based on this system are connected to one constant power for shooting ball. They only shoot at the end point of their playing algorithm to opponent goal and they don't use their kicking system for passing to other robots. The reason of this behavior is that the power of solenoid kicking system is calibrated for maximum velocity of ball for shooting to the opponent goal. Almost all robots use this system only in one direction. So when robot wants to shoot toward a different direction from its chassis, it must rotate. Rotating is a time consuming movement and robots of opponent team can figure out where robot aims to shoot. So they distort the process of shooting by closing the front of robot.

In this paper a multi power multi direction kicking system is developed. This versatility is achieved without a need of either changing the power supply of robot or the direction of whole chassis of robot.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 602 – 607, 2006.

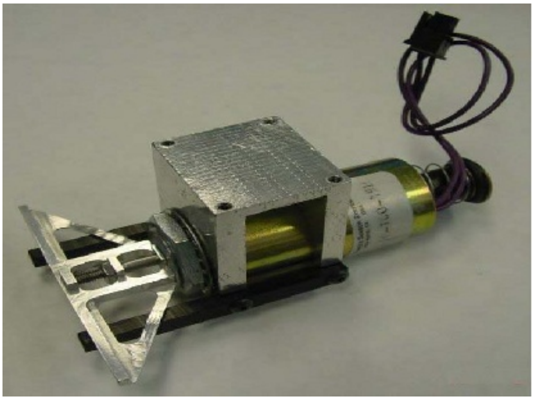


Fig. 1. Kicking system of Cornell Big Red team

2 Collision Analysis

When a collision is occurred between kicker system and ball, the velocity of ball is changed. The summation of momentum of ball and kicker before and after collision is equal. Consider to the equation of conservation of momentum (1) and consider that velocity of kicker vanishes after occurrence of collision because of opponent direction of magnetic force of solenoid.

$$m_{ball} \times \vec{v}_{ball} + m_{bar} \times \vec{v}_{bar} (beforecollision) = m_{ball} \times \vec{v}_{ball} + m_{bar} \times \vec{v}_{bar} (aftercollision)$$

$$m_{bar} \times \vec{v}_{bar} (beforecollision) = m_{ball} \times \vec{v}_{ball} (aftercollision) \tag{1}$$

So, to obtain greater velocity of ball, the mass of solenoid system or the velocity of solenoid must be greater. The mass of solenoid is constant and its increase may cause many faults in motivation of robot in a match. So to get the maximum velocity for ball, the velocity of bar (core) of solenoid should be increased, in other words in order to be able to change the ball velocity we must be able to modify the bar speed. So the efforts should be concentrated on reaching the maximum bar velocity and for obtain the ability of changing velocity the bar.

3 Calculating Transmission Velocity

The velocity of a bar in the final position of solenoid is calculated from equation (2). In this equation the initial velocity of bar is considered to be zero.

$$\vec{v} = \int \vec{a}.dt \tag{2}$$

So, to obtain greater velocity, the acceleration of solenoid bar should be greater and for changing the velocity the acceleration should be changed. Now consider to the equation (3) for changing the acceleration of solenoid bar.

$$\vec{F} = m \times \vec{a} \tag{3}$$

To obtain a greater acceleration, greater a force to the solenoid bar is needed by using greater magnetic field, and for changing the magnitude force which is forced to the bar, the force should be changed. Now consider to the equation (4) for finding the force.

$$\vec{F} = -\vec{\nabla} w \tag{4}$$

The above equation shows that the force is equal to gradient of work. So the work of each position of solenoid bar must be calculated at two positions. First, position is the start point and then at the end of movement. These works could be calculated by Electromagnetic laws (5).

$$\begin{aligned} w_1 &= \tfrac{1}{2} L_1 I^2 \\ w_2 &= \tfrac{1}{2} L_2 I^2 \end{aligned} \tag{5}$$

Which “W1” equals work in start position, and “W2” equals work in the end position of movement. Using the above equations the force could be found by the equation (6).

$$\vec{F} = -\frac{d(\tfrac{1}{2} L_2 I^2 - \tfrac{1}{2} L_1 I^2)}{dx} \tag{6}$$

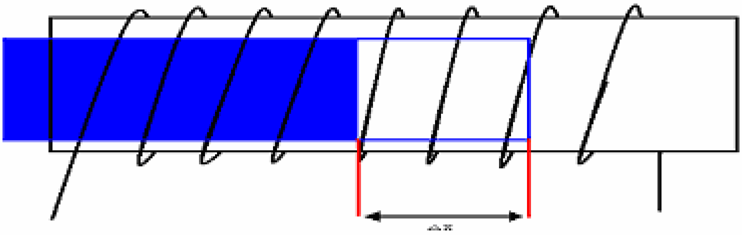


Fig. 2. A solenoid motivation

Inductance “L” could be calculated by the equation (7).

$$L = \frac{N^2}{R} \tag{7}$$

Which “N” is turns of wire of solenoid for producing the electromagnetic field for movement of bar and "R" is reluctance. By equation (8) the Reluctance of the magnetic circuit for solenoid could be calculated.

$$R = \frac{l_{avr}}{\mu.A} \tag{8}$$

The force could be calculated from the above equations. These calculations are presented in equations (9).

$$F = -{}^{1}\!/_{2}\, I^2 \frac{d(L_2 - L_1)}{dx} = -{}^{1}\!/_{2}\, I^2 N^2 \frac{d({}^{1}\!/_{R_2} - {}^{1}\!/_{R_1})}{dx} \tag{9}$$

$$\Rightarrow F = -{}^{1}\!/_{2}\, I^2 N^2 \mu A \frac{d(l_{avr2} - l_{avr1})}{dx}$$

So, to obtain maximum force for shooting ball “I” (current flows in wires of solenoid), “N” (number turns of wire of solenoid), Permeability of core of solenoid and surface area of solenoid must be maximum. The only Value which could be changed to have different power of shooting is the current which flows in the solenoid.

So we must change the value of current to have different powers. The resistance of wires of solenoid is constant (for DC currents), so the voltage which forces the solenoid must be changed.

4 Producing Different Voltages from a Battery

In this section a main circuit for producing the sufficient voltages for solenoid is introduced. This circuit should feed the solenoid to have maximum velocity or have different velocity of ball. There are several circuits for producing different voltages from a DC power supply introduced in Power Electronics.

In this paper we use a Boost regulator. In a boost regulator, the output voltage is greater than the input voltage. A boost regulator is shown in figure 3.

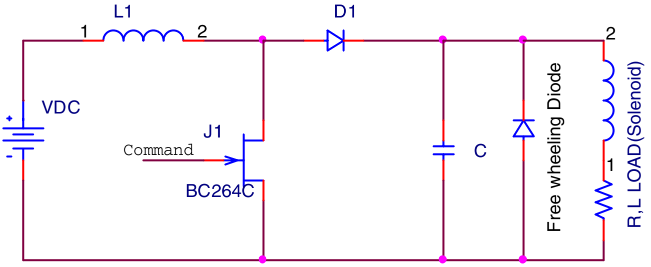


Fig. 3. Circuit of a Boost regulator

The circuit operation can be divided into two modes. Mode 1 begins when transistor switched on. The input current, which rises, flows through inductor "L" and transistor. Mode 2 begins when transistor is switched off. The current which was flowing through the transistor would now flow through "L","C", load and "D1". The inductor current falls until transistor is turned on again in the next cycle. The energy stored in the inductor "L" will be transferred to load .The output voltage of the regulator could be found by using the volt second law and is shown in equation (10).

$$V_{out} = (\frac{1}{1-d})V_{in} \tag{10}$$

The waveform for voltages is shown in figure 4 for three different duty cycles of transistor. So, to have several voltages from Boost regulator, the Duty cycle of transistor must be calculated and changed. A PWM system must be designed for changing Duty cycle of feed for transistor. To design a PWM circuit a timer must be used. This timer could be an independent timer or a timer of a microcontroller.

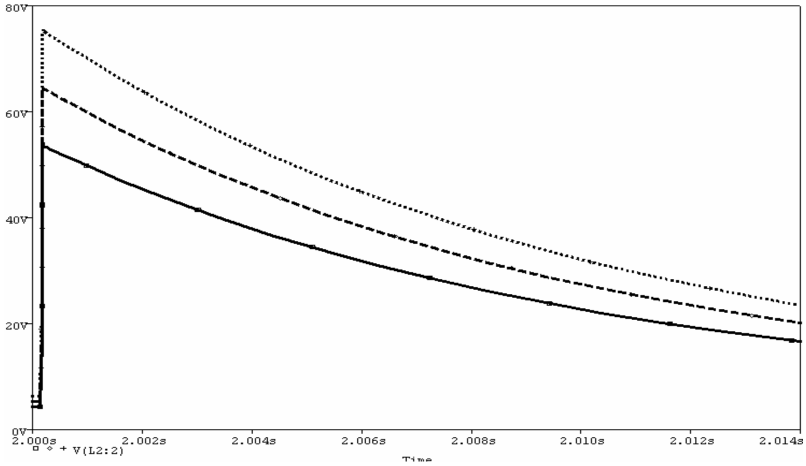


Fig. 4. Wave form for 3 different duty cycles

5 Multi Direction Kicking System

A multi direction kicking system consists of three different independent solenoids which could shoot the ball independently. The arrangement of three different solenoids in three different angles could be implemented for different directions. For this arrangement of solenoid systems, they must be small and light enough. They must be small to produce different angles and they must be light because the weight of robot play the most important role in amount of time for reaching the ball in the match and this time is an important factor in the small size Robocup matches.

To reach each direction the velocity of each of three solenoids should be calculated. The combination of velocity of solenoids produce a vector for shooting. The pattern of this vector is the pattern of shooting, and the magnitude of vector is power of shooting.

Usage of three independent solenoids could increase the power of shooting, too. The figure 5 shows the maximum power of shooting for different angles of two solenoids respect to central solenoid. In this figure “X” is the first solenoid angle with respect to central solenoid and “Y” is the second solenoid angle with respect to central solenoid.

As the figure 5 shows, maximum velocity of ball will be resulted in the situation which, the angles of two beside solenoids are equal to each other. This velocity will be in the maximum value when these angles vanish, but to obtain several directions, these angles must be nonzero. By setting the angle of solenoids and the voltages which feed to the boost regulator, multi power and multi direction of shooting will be obtained.

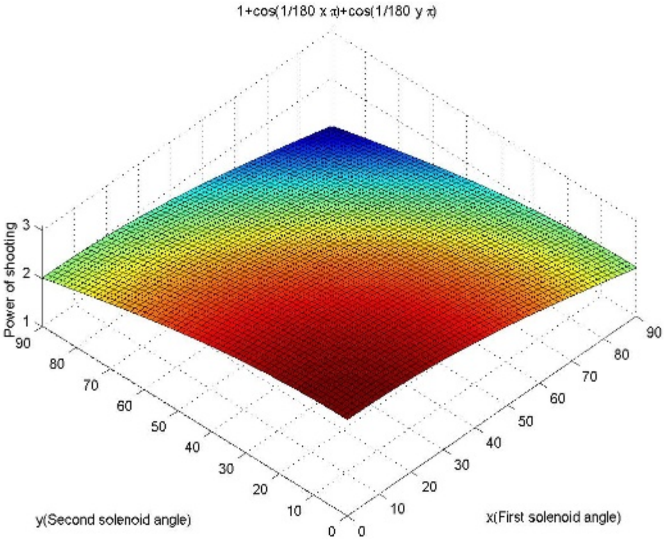


Fig. 5. maximum Power of shooting

References

[1] O.Hashemipour, N.Noori, M.A.Karami, B.Dashtbozorg "SBURT team Description paper" Shahid Beheshti Robocup Small size team (2004)

[2] Patrick Dingle, Jieun Kim” 2004 Cornell university Robocup Documentation” Mechanical group Final documentation (2004)

[3] ”Robocup 2003” a system engineering award, Cornell university Robocup small size team (2003)

[4] M.H.Rashid “Power electronics” circuit devices and applications

[5] Stephen. J .Chapman ”Electric machinery fundamentals”

Particle-Filter-Based Self-localization Using Landmarks and Directed Lines[*]

Thomas Röfer[1], Tim Laue[1], and Dirk Thomas[2]

[1] Center for Computing Technology (TZI), FB 3, Universität Bremen
roefer@tzi.de, timlaue@tzi.de
[2] Fachgebiet Simulation und Systemoptimierung, Fachbereich Informatik,
Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany
dthomas@rbg.informatik.tu-darmstadt.de

Abstract. The paper presents the self-localization approach used by the World Champion in the Sony Four-Legged Robot League 2004. The method is based on a particle filter that makes use of different features from the environment (beacons, goals, field lines, field wall) that provide different kinds of localization information and that are recognized with different frequencies. In addition, it is discussed how the vision system acquires these features, especially, how the orientation of field lines is determined at low computational costs.

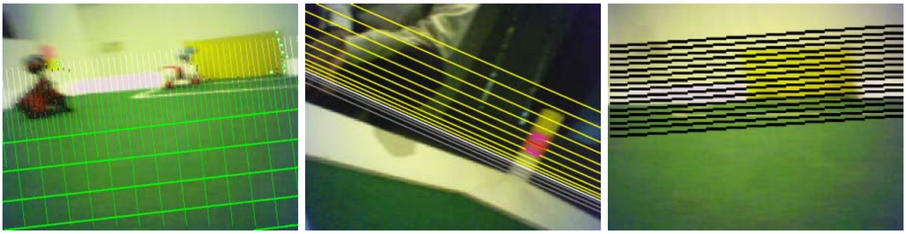
1 Introduction

The Sony Four-Legged Robot League (SFRL) is one of the official leagues in RoboCup, in which a standardized robot platform is used, namely the Sony Aibo. The robots act completely autonomously. The main sensor of the Sony Aibo is the camera located in its head. The head can be turned around three axes (2× tilt, 1× pan), and the camera has a field of view of approximately 57° by 42°. The soccer field in the SFRL has a size of approximately 5m×3m. As the main sensor of the robot is a camera, all objects on the RoboCup field are color coded. There are two-colored beacons for localization (pink and either yellow or skyblue), the two goals are of different color (yellow and skyblue), the field is green, and the field lines as well as the field wall are white.

During actual RoboCup games, the beacons and goals are rarely perceived, especially by the robot that is handling the ball. Therefore it is advantageous if also the field lines and the field wall can be employed for localization. However, different features in the environment are recognized with different frequency and they provide different kinds of information usable for localization. Lines only provide localization information perpendicular to their orientation. The *field lines* are mostly oriented across the field, but the side lines of the penalty area also provide important information, especially for the goalie. The field lines are seen less often than the *field wall* that is surrounding the field. Therefore the latter provides information in both Cartesian directions, but it is often quite far

[*] The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 608–615, 2006.



a) b) c)

Fig. 1. Different scanlines and grids. a) The main grid which is used to detect objects on the field. b) The grid lines for beacon detection. c) The grid lines for goal detection.

away from the robot. Therefore, the distance information is less precise than the one provided by the field lines. The field wall is seen from nearly any location on the field. *Goals* and *beacons* are the only means to determine the orientation on the field, because the field lines and the field wall are mirror symmetric. Goals and beacons are seen only rarely. It turned out that the vision system is reliably able to determine the orientation of field lines, while the orientation of the edge between field wall and field is not as stable. Therefore, it is distinguished between field lines along the field and field lines across the field. This especially improves the localization of the goalie, because it sees both types of lines surrounding the penalty area.

2 Acquiring Localization Information

The vision system of the GermanTeam processes images of a resolution of 208 × 160 pixels, but looking only at a grid of less pixels. The grid is aligned to the so-called horizon, i. e. the plane that is in parallel to the field plane, but on the height of the camera. The idea is that for feature extraction, a high resolution is only needed for small or far away objects. In addition to being smaller, such objects are also closer to the horizon. Thus only regions near the horizon need to be scanned at a relative high resolution, while the rest of the image can be scanning using a wider spaced grid.

Each grid line is scanned pixel by pixel from top to bottom and from left to right respectively. During the scan each pixel is classified by color. A characteristic series of colors or a pattern of colors is an indication of an object of interest, e. g., a sequence of some orange pixels is an indication of a ball, a sequence of some pink pixels is an indication of a beacon, an (interrupted) sequence of sky-blue or yellow pixels followed by a green pixel is an indication of a goal, a sequence of white to green or green to white is an indication of an edge between the field and the border or a field line, and a sequence of red or blue pixels is an indication of a player. All this scanning is done using a kind of state machine; mostly counting the number of pixels of a certain color class and the number of pixels since a certain color class was detected last. That way, beginning and

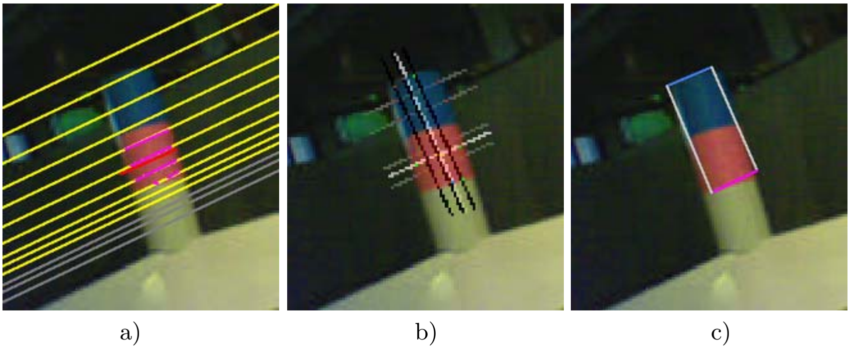


Fig. 2. Three steps in beacon detection: a) Scanlines searching for pink runs. The pink segments are the detected pink runs, the red segment is the result of clustering. b) The specialist detects the edges of the beacon. c) The generated percept.

end of certain object types can still be determined although some pixels of the wrong class are detected in between.

To speed up the object detection and to decrease the number of false positives, essentially three different grids are used. The main grid covers the area around and below the horizon. It is used to search for all objects which are situated on the field, i. e. the ball, obstacles, other robots, field borders, field lines, and the lower borders of the goals (cf. Fig. 1a). A set of grid lines parallel to and in most parts over the horizon is used to detect the pink elements of the beacons (cf. Fig. 1b and Fig. 2). The goal detection is also based on horizontal grid lines (cf. Fig. 1c).

As a first step towards a more color table independent classification, points on edges are only searched at pixels with a big difference of the Y channel of the adjacent pixels. An increase in the Y channel followed by a decrease is an indication of an edge. If the color above the decrease in the Y channel is sky-blue or yellow, the pixel lies on an edge between a goal and the field. The detection of points on field lines and borders is still based on the change of the segmented color from green to white or the other way round.

The differentiation between a field line and the border is a bit more complicated. In most cases, the border has a bigger size in the image than a field line. But a far distant border might be smaller than a very close field line. Therefore the pixel, where the segmented color changes back from white to green after a green-to-white change before, is assumed to lie on the ground. With the known height and rotation of the camera, the distance to that point is calculated. The distance leads to expected sizes of the field line in the image. For the classification, these sizes are compared to the distance between the green-to-white and the white-to-green change in the image to determine if the point belongs to a field line or a border. The projection of the pixels on the field plane is also used to determine their relative position to the robot.

In addition, for every point classified as being on the edge of a field line or the field wall, the gradient of the Y channel is computed (cf. Fig. 3a,b). This

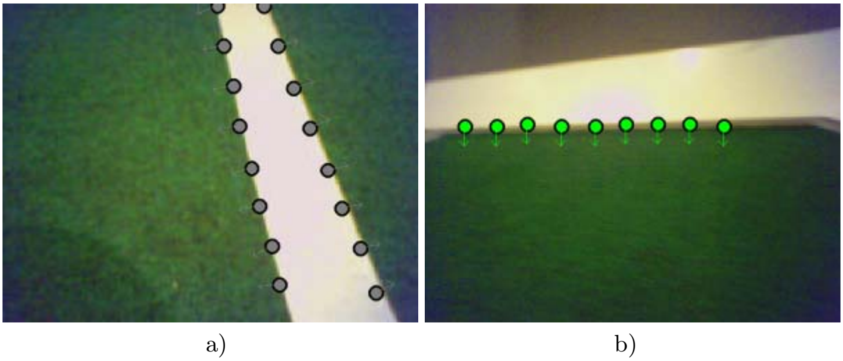


Fig. 3. Points on the edges, including computed gradients. a) On a line. b) On a field wall.

gradient is based on the values of the Y channel of the edge point and three neighboring pixels, using a Roberts operator ([3]):

$$\begin{array}{l} s = I\left[x+1\ y+1\right] - I\left[x\ y\right] \\ t = I\left[x+1\ y\right] - I\left[x\ y+1\right] \\ |\nabla I| = \sqrt{s^2+t^2} \\ \angle\nabla I = \arctan\left(\frac{s}{t}\right) - \frac{\pi}{4} \end{array} \qquad (1)$$

where $|\nabla I|$ is the magnitude and $\angle\nabla I$ is the direction of the edge. $\angle\nabla I$ is afterwards projected to the field plane, resulting in a direction in field coordinates.

3 Self-localization

A Markov-localization method employing the so-called Monte-Carlo approach [1] is used to determine the position of the robot. It is a probabilistic approach, in which the current location of the robot is modeled as the density of a set of particles. Each particle can be seen as the hypothesis of the robot being located at this posture. Therefore, such particles mainly consist of a robot pose, i. e. a vector $pose = (x, y, \theta)$ representing the robot's x/y-coordinates in millimeters and its rotation θ in radians. A Markov-localization method requires both an observation model and a motion model. The observation model describes the probability for taking certain measurements at certain locations. The motion model expresses the probability for certain actions to move the robot to certain relative postures. The one used is described in [2].

The localization approach works as follows: first, all particles are moved according to the motion model of the previous action of the robot. Then, the probabilities for all particles are determined on the basis of the observation model for the current sensor readings, i. e. bearings on landmarks calculated from the actual camera image. Based on these probabilities, the so-called *resampling* is performed, i. e. moving more particles to the locations of samples with a high probability. Afterwards, the average of the probability distribution is determined, representing the best estimation of the current robot pose. Finally, the process repeats from the beginning. Since the general approach has already been described in [2], this paper focuses on how to combine the perceptions of beacons, goals, and (directed) edge point in a way that results in a stable self-localization.

Observation Model. The observation model relates real sensor measurements to measurements as they would be taken if the robot were at a certain location. Instead of using the distances and directions to the landmarks in the environment, i. e. the beacons and the goals, this localization approach only uses the directions to the vertical edges of the landmarks. However, although the points on edges determined by the image processor are represented in a metric fashion, they are also converted back to angular bearings. The advantage of using landmark edges for orientation is that one can still use the visible edge of a landmark that is partially hidden by the image border. Therefore, more points of reference can be used per image, which can potentially improve the self-localization.

The utilized percepts are bearings on the edges of beacons and goals, and points on edges between the field and the field lines, the field wall, and the goals. These have to be related to the assumed bearings from hypothetical postures. As has been pointed out in [2], the different percepts contain different kinds of localization information and are seen with different frequencies. Therefore, it is required to represent separate probabilities for beacons and goals, horizontal field lines, vertical field lines, field walls, and goal edges for each particle.

As the positions of the samples on the field are known, it can be determined for each measurement and each sample, where the measured points would be located on the field if the position of the sample was correct. For each of these measured points in field coordinates, it can be calculated, where the closest point on a real field line of the corresponding type is located. Then, the horizontal and vertical angles from the camera to this model point are determined. These two angles of the model point are compared to the two angles of the measured point. The smaller the deviations between the model point and the measured point from a hypothetic position are, the more probable the robot is really located at that position. Deviations in the vertical angle (i. e. distance) are judged less rigidly than deviations in the horizontal angle (i. e. direction).

Calculating the closest point on an edge in the field model for a small number of measured points is still an expensive operation if it has to be performed for, e. g., 100 samples. Therefore, the model points are pre-calculated for each edge type and stored in two-dimensional lookup tables with a resolution of 2.5 cm. That way, the closest point on an edge of the corresponding type can be determined by a simple table lookup. Figure 4 visualizes the distances of measured points to the closest model point for the four different edge types.

Probabilities for Beacons and Goals. The observation model only takes into account the bearings on the edges that are actually seen, i. e., it is ignored if the robot has *not* seen a certain edge that it should have seen according to its hypothetical posture and the camera pose. Therefore, the probabilities of

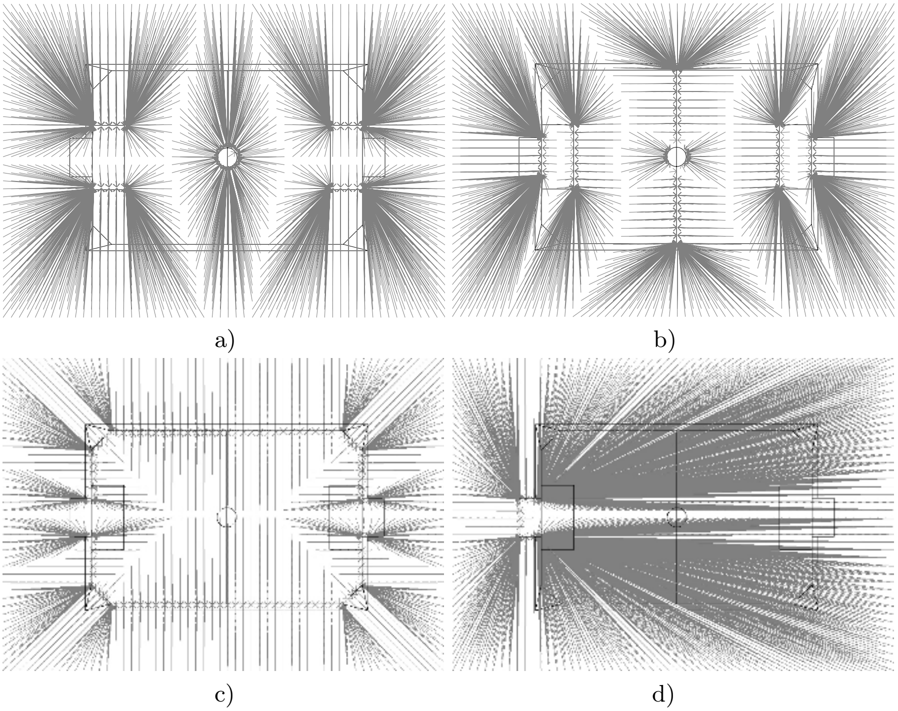


Fig. 4. Mapping of positions to closest edges. a) Field lines along the field. b) Field lines across the field. c) Field wall. c) A goal.

the particles are only calculated from the similarities of the measured angles to the expected angles. Each similarity s is determined from the measured angle $\omega_{measured}$ and the expected angle $\omega_{expected}$ for a certain pose by applying a sigmoid function to the difference of both angles:

$$s(\omega_{measured}, \omega_{expected}) = \begin{cases} \mathrm{e}^{-50d^2} & \text{if } d < 1 \\ \mathrm{e}^{-50(2-d)^2} & \text{otherwise} \end{cases} \qquad (2)$$
$$\text{where } d = \frac{|\omega_{measured} - \omega_{expected}|}{\pi}$$

The probability $q_{landmarks}$ of a certain particle is the product of these similarities:

$$q_{landmarks} = \prod_{\omega_{measured}} s(\omega_{measured}, \omega_{expected}). \qquad (3)$$

Probabilities for Edge Points. The probabilities of the particles are calculated from the similarities of the measured angles to the expected angles. Each similarity s is determined from the measured angle ω_{seen} and the expected angle ω_{exp} for a certain pose by applying a sigmoid function to the difference of both angles weighted by a constant σ:

$$s(\omega_{seen}, \omega_{exp}, \sigma) = \mathrm{e}^{-\sigma(\omega_{seen}-\omega_{exp})^2} \tag{4}$$

If α_{seen} and α_{exp} are vertical angles and β_{seen} and β_{exp} are horizontal angles, the overall similarity of a sample for a certain edge type is calculated as:

$$q_{edge\ type} = s(\alpha_{seen}, \beta_{seen}, \alpha_{exp}, \beta_{exp}) = s(\alpha_{seen}, \alpha_{exp}, 10 - 9\frac{|v|}{200}) \cdot s(\beta_{seen}, \beta_{exp}, 100) \tag{5}$$

For the similarity of the vertical angles, the probability depends on the robot's speed v (in mm/s), because the faster the robot walks, the more its head shakes, and the less precise the measured angles are.

Calculating the probability for all points on edges found and for all samples in the Monte-Carlo distribution would be a costly operation. Therefore, only three points of each edge type (if detected) are selected per image by random. To achieve stability against misreadings, resulting either from image processing problems or from the bad synchronization between receiving an image and the corresponding joint angles of the head, the change of the probability of each sample for each edge type is limited to a certain maximum. Thus misreadings will not immediately affect the probability distribution. Instead, several readings are required to lower the probability, resulting in a higher stability of the distribution. However, if the position of the robot was changed externally, the measurements will constantly be inconsistent with the current distribution of the samples, and therefore the probabilities will fall rapidly, and resampling will take place.

The filtered probability q' for a certain type is updated ($q'_{old} \rightarrow q'_{new}$) for each measurement of that type:

$$q'_{new} = \begin{cases} q'_{old} + \Delta_{up} & \text{if } q > q'_{old} + \Delta_{up} \\ q'_{old} - \Delta_{down} & \text{if } q < q'_{old} - \Delta_{down} \\ q & \text{otherwise.} \end{cases} \tag{6}$$

For landmarks, $(\Delta_{up}, \Delta_{down})$ is $(0.1, 0.05)$, for edge points, it is $(0.01, 0.005)$.

Overall Probability. The probability p of a certain particle is the product of the three separate probabilities for bearings on landmarks, edges of field lines along and across the field, the field wall, and goals:

$$p = q'_{landmarks} \cdot q'_{longitudinal\ lines} \cdot q'_{latitudal\ lines} \cdot q'_{field\ walls} \cdot q'_{goals} \cdot \tag{7}$$

4 Results

[2] presented quantitative results on the precision of the localization approach on an empty field using only lines and goals. The recent improvements clearly target to achieving a good localization during actual RoboCup games, i. e. in situations in which the main focus is on perceiving the ball and localization information is recognized rather rarely. Therefore, the games of the GermanTeam performed in Lisbon (videos at `http://www.tzi.de/4legged`) represent a good

evaluation of how well the localization system works, because many parts of the behavior description of the GermanTeam rely on correct localization, e. g. which kick is selected at which position, and the placement of the defensive players, especially the goal keeper. At RoboCup 2003, it was also demonstrated that the GermanTeam can play soccer without the beacons.

5 Conclusions

This paper presents how beacons, goals, as well as points on edges between the field and field lines or field walls are determined, namely features that are required to localize on a RoboCup field. It is also shown how the edge points are augmented with the direction of the edge using a computationally cheap operation. All these features are used by a particle filter to determine the position of the robot. Here, separate probabilities for different features are used per particle, because the features provide different information about the position of the robot, and they are recognized with different frequencies.

Acknowledgments

The authors thank all members of the GermanTeam for providing the basis for this research.

References

1. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.
2. T. Röfer and M. Jüngel. Fast and robust edge-based localization in the sony four-legged robot league. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, number 3020 in Lecture Notes in Artificial Intelligence, pages 262–273, Padova, Italy, 2004. Springer.
3. L.G. Roberts. Machine perception of three dimensional solids. *Optical and Electro-Optical Information Processing*, pages 159–197, 1968.

Performance Evaluation of an Evolutionary Method for RoboCup Soccer Strategies

Tomoharu Nakashima[1], Masahiro Takatani[1], Masayo Udo[1],
Hisao Ishibuchi[1], and Manabu Nii[2]

[1] Osaka Prefecture University, Gakuen-cho 1-1, Sakai, Osaka 599-8531, Japan
{nakashi, takatani, udo, hisaoi}@ie.osakafu-u.ac.jp
http://www.ie.osakafu-u.ac.jp/~hisaoi
[2] University of Hyogo, Shosha 2167, Himeji, Hyogo 671-2201, Japan
nii@eng.u-hyogo.ac.jp

Abstract. This paper proposes an evolutionary method for acquiring team strategies of RoboCup soccer agents. The action of an agent in a subspace is specified by a set of action rules. The antecedent part of action rules includes the position of the agent and the distance to the nearest opponent. The consequent part indicates the action that the agent takes when the antecedent part of the action rule is satisfied. The action of each agent is encoded into an integer string that represents the action rules. A chromosome is the concatenated string of integer strings for all agents. We employ an ES-type generation update scheme after producing new integer strings by using crossover and mutation. Through computer simulations, we show the effectiveness of the proposed method.

1 Introduction

RoboCup soccer [1] is a competition between soccer robots/agents and its ultimate purpose is to win against the human soccer champion team by the year 2050. Developing RoboCup teams involves solving the cooperation of multiple agents, the learning of adaptive behavior, and the resolution to noisy data handling. Many researchers have been tackling with these problems. An example of them is the application of soft computing techniques [2].

In general, the behavior of the soccer agents is hierarchically structured. This structure is divided into two groups. One is low-level behavior that performs basic information processing such as visual and aural information. Basic actions such as dribble, pass, and shoot are also included in the low-level behavior. On the other hand, high-level behavior makes a decision from the viewpoint of global team strategy such as cooperative play among the teammates.

For low-level behavior, Nakashima et al.[2] proposed a fuzzy Q-learning method for acquiring a ball intercept skill. In [2], it is shown that the performance of the agent gradually improves through trial-and-error.

Evolutionary computation has been used to evolve strategies of games. For example, Chellapilla [3, 4] proposed a method based on the framework of evolutionary programming to automatically generate a checker player without incorporating human expertise on the game. An idea of coevolution is employed

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 616–623, 2006.

in [3, 4]. For RoboCup soccer agents, a genetic programming approach has been applied to obtain the soccer team strategy in [5]. In [5], the idea of coevolution is also employed. The evolution of team strategy from kiddy soccer (i.e., all players gather around the ball) to formation soccer (i.e., each player maintains its own position during the game) is reported.

In this paper, we propose an evolutionary method for acquiring the strategy of RoboCup soccer teams. High-level behavior of soccer teams can be obtained by the proposed method. The characteristic feature of the proposed method is that the behavior of the soccer teams is represented by an integer string. The integer strings is a concatenation of the behavior of ten players. A set of action rules is encoded into the integer strings. The actions of soccer agents are specified by the action rules when they keep a ball. The antecedent part of the action rules is the position of the agent and its nearest opponent agent. The action to be taken is indicated by the consequent part of the action rules that best describes the agent's condition. We examine the effectiveness of the proposed method through computer simulation. The future research direction is also described in this paper.

2 Team Setup

2.1 UvA Trilearn: Base Team

In this paper, we use UvA Trilearn for our evolutionary computation method. UvA Trilearn won the RoboCup world competition that was held in Padua, Italy in 2003. The source codes of UvA Trilearn are available from their web site [7]. Low level behaviors such as communication with the soccer server, message parsing, sensing, and information pre-processing are implemented. Basic skills such as player's positioning, ball intercept, and kick are also implemented in the available source codes. High level behaviors such as strategic teamworks, however, are omitted from the source codes.

UvA Trilearn players take rather simple actions as high level behaviors are not implemented in the released source codes. We show the action tree of the UvA Trilearn players in Fig. 1. There are two action modes: One is ball handling mode, and the other is positioning mode. Each of the players uses one of these two modes every time step depending on its situation in the soccer field. When a player is the nearest one to the ball among the team, the ball handling mode is carried out. On the other hand, when a player is not the nearest one to the ball among the team, it takes the positioning mode. The following subsections explain these two modes in detail. Note that this action is common for all players. Thus, players follow the same action tree to determine the action every time step. Since their conditions and home positions are different from each other, the action taken at a time step is not necessarily the same for all players. Due to the limitation of space, we describe the ball handling mode only in the following section.

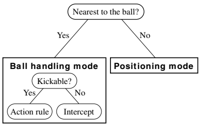


Fig. 1. Action tree of UvA Trilearn players

2.2 Ball Handling Mode

The ball handling mode is employed when a player is the nearest one to the ball among the team, In this mode, the player checks if it can kick the ball or not. A kickable margin is defined by the RoboCup soccer server. A player can kick the ball if the ball is in the kickable area of the player. Otherwise it is impossible to kick the ball as the ball is out of its kickable area. In this case, the player moves towards the ball until the ball is within its kickable area.

In the original UvA base team, the player always shoot the ball to the opponent goal if the ball is kickable. We modified this behavior for our evolutionary computation. We use a set of action rules for determining the action of a player that is kickable the ball. The action rule set represents the strategy of a soccer team. In this paper, we evolve action rule sets to find a competitive soccer team strategy.

The action rules of the following type are used in this paper:

$$R_j : \begin{array}{l} \text{If Agent is in Area } A_j \text{ and the nearest opponent is } B_j \\ \text{then the action is } C_j, \quad j = 1, 2, \ldots, N, \end{array} \tag{1}$$

where R_j is the rule index, A_j is the antecedent integer value, B_j is the antecedent linguistic value, C_j is the consequent action, and N is the number of action rules.

The antecedent integer value $A_j, j = 1, \ldots, N$, refers to a subarea of the soccer field. We divide the soccer field into 48 subareas as in Fig. 2. Each subarea is labaled an integer value. The antecedent value A_j of the action rule R_j is an

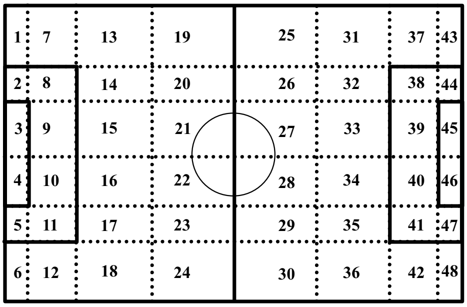


Fig. 2. Soccer field

integer value in the interval $[1, 48]$. The action of the soccer agent also depends on the distance between the agent and its nearest opponent. The antecedent B_j takes one of two linguistic values *near* or *not near*. The player that is kickable the ball examines whether the nearest opponent is near the agent or not. We use two linguistic values for the antecedent part B_j. The nearest opponent is regarded as *near* if the distance between the agent and its nearest opponent is less than a prespecified value. If not, the nearest opponent is regarded as *not near* the player. The consequent action C_j represents the action that is taken by the agent when the two conditions in the antecedent part of the action rule R_j (i.e., A_j and B_j) are satisfied. In this paper, we use the following ten actions for the consequent action C_j.

1. Dribble toward the opponent side. The direction is parallel to the horizontal axis of the soccer field.
2. Dribble toward the opponent side. The direction is the center of the opponent goal.
3. Dribble carefully toward the opponent side. The direction is the center of the opponent goal. The dribble speed is low so that the agent can avoid the opponent agent.
4. Pass the ball to the nearest teammate. If the nearest teammate is not ahead of the agent, the agent does not kick to the nearest teammate. Instead, it clears the ball toward the opponent side.
5. Pass the ball to the second nearest teammate. If the second nearest teammate is not ahead of the agent, the agent does not kick to the second nearest teammate. Instead, it clears the ball toward the opponent side.
6. Clear the ball toward the opponent side.
7. Clear the ball toward the nearest side line of the soccer field.
8. Kick the ball toward the penalty area of the opponent side (i.e., centering).
9. Perform a leading pass to the nearest teammate.
10. Shoot the ball toward the nearest post of the opponent goal.

Note that each player has a set of action rules. Since there are 48 subareas in the soccer field and *near* and *not near* are available for the second antecedent part in action rules (i.e., B_j), the number of action rules for a single player is $48 \times 2 = 96$. There are $96 \times 10 = 960$ action rules in total for a single team with ten field players. Action rules for a goal keeper are not considered in this paper.

There is a special case where players do not follow the action rule. If a player keeps the ball within the penalty area of the opponent side (i.e., if the agent is in Areas $38 \sim 41$ or $44 \sim 47$ in Fig. 2), the player checks if it is possible to shoot the ball to the opponent goal. The player decides to shoot the ball if the line segment from the ball to either goal post of the opponent side is clear (i.e., there are no players on the line segment). If the line is not clear, the player follows the action rule whose antecedent part is compatible to the player's condition.

If the ball is not kickable for a player that is in the ball handling mode, the player's action is to intercept the ball, that is, the player moves to catch the ball. In the intercept process, the player determines whether it dashes forward or turn its body angle based on the relative distance of the ball to the player.

3 Evolutionary Computation

In this paper, we use an evolutionary method to obtain team strategies for soccer agents that are effective for playing soccer. Specifically, our aim is to find the best action rule sets for ten soccer players. Each player has its own set of action rules that are used when it is in the ball handling mode (see Subsection 2.2). We encode an entire team into an integer string. Note that we do not optimize player's individual behavior but a team strategy as a whole. Thus, we evaluate the performance of a team strategy only from its match result, not from player's individual tactics. We show in our computer simulations that the performance of team strategies successfully improves through the evolution. The following subsections explain our evolurionary method in detail.

3.1 Encoding

As described in Section 2.2, an action of the agents is specified by the action rules in (1) when it keeps the ball. Considering that the soccer field is divided into 48 subfields (see Fig. 2) and the position of the nearest opponent agent (i.e., it is *near* the agent or *not near*) is taken into account in the antecedent part of the action rules, we can see that there are $48 \times 2 = 96$ action rules for each player. In this paper, we apply our evolutionary method to ten soccer agents excluding a goal keeper. Thus, the total number of action rules for a single team is $96 \times 10 = 960$. We use an integer string of length 960 to represent a rule set of action rules for ten players. The task of our proposed evolutionary method is then to evolve the integer strings of length 960 to obtain team strategies with high performance. We show in Fig. 3 the first 96 bits of an integer string in our evolutionary method. This figure shows an integer string for a single agent. In Fig. 3, the first 48 bits represent the actions of an agent when the nearest opponent agent is *near* the agent. The order of the integer bits is based on the position of the agent in Fig. 2. On the other hand, the actions of the agent in the case where the nearest opponent agent is *not near* the agent are shown in the remaining 48 bits. The value of each integer bit ranges from an integer interval $[1, 10]$. These integer values correspond to the index number of ten actions described in Section 2.2.

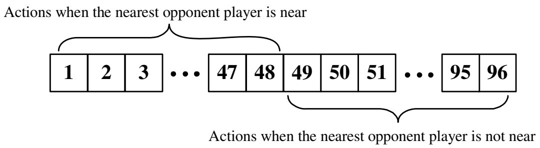

Actions when the nearest opponent player is near
1
2
3
• • •
47
48
49
50
51
• • •
95
96
Actions when the nearest opponent player is not near

Fig. 3. Integer string for a single agent

3.2 Evaluation of Integer Strings

Generally, the main idea of evolutionary methods is to exploit the information of those individuals whose performance is highly evaluated. In this paper, we evaluate the performance of integer strings through the results of soccer games. Specifically, we use the scores of the soccer games as performance measure in our evolutionary method. We first check the scored goals by the soccer teams that are represented by the integer strings. The more goals a soccer team scores, the higher the performance of the integer string for the soccer team is. When the number of the goals is the same among multiple soccer teams, the conceded goals are used as a second performance measure. The soccer teams with lower conceded goals are evaluated as better teams. We do not consider the conceded goals at all when the goals are different between soccer teams to be evaluated.

3.3 Evolutionary Operation

We use one-point crossover, bit-change mutation, and ES-type selection as evolutionary operations in our evolutionary method. New integer strings are generated by crossover and mutation and selection is used for generation update.

In the crossover operation, first we randomly select two integer strings. Then latter part of both strings are exchanged with each other from a randomly selected cut-point. Note that we do not consider any evaluation results when two integer strings for the crossover operation are selected from the current population. In the mutation operation, the value of each integer bit is replaced with a randomly specified integer value in the interval $[1, 10]$ with a prespecified mutation probability. It is possible that the replaced value is the same as the one before the mutation operation.

Generation update is performed by using ES-type selection in our method. By iterating the crossover and the mutation operations, we produce the same number of new integer strings as that of the current strings. Then the best half integer strings from the merged set of the current and the new strings are chosen as the next generation. The selection is based on the match results as described in Subsection 3.2. This generation update is similar to the $(\mu + \lambda)$-strategy of evolution strategy [6]. Note that the current strings are also evaluated in this selection process. Thus, it is possible that a current integer string with the best performance at the previous generation update is not selected in the next generation update because the performance of the integer string in the next performance evaluation is the worst among the merged strings.

To summarize, our proposed evolutionary method is written as follows:

[Procedure of the proposed evolutionary method]

Step 1: Initialization. A prespecified number of integer strings of length 960 are generated by randomly assigning an integer value from the interval $[1, 10]$ for each bit.

Step 2: Generation of new integer strings. First randomly select two integer strings from the current population. Then the one-point crossover and the bit-change mutation operations are performed to generate new integer strings. This process is iterated until a prespecified number of new integer strings are generated.

Step 3: Performance evaluation. The performance of both the current integer strings and new integer strings generated by Step 2 is evaluated through the results of soccer games. Note that the performance of current integer strings is also evaluated every generation because the game results are not constant but different game by game.

Step 4: Generation update. From the merged set of the current integer strings and new ones, select best integer strings according to the performance evaluation in Step 3. The selected bit strings form the next generation.

Step 5: Termination of the procedure. If a prespecified termination conditions are satisfied, stop the procedure. Otherwise go to Step 2.

4 Computer Simulations

The following parameter specifications were used for all the computer simulations in this paper:

- The number of integer strings in a population: 5,
- The probability of crossover: 1.0
- The probability of mutation for each bit: 1/96,
- Generation of initial integer strings: one hand-coded and the others randomly generated.

The initial population was created by randomly assinging 1 $\sim$ 10 for each integer bit. To evaluate the performance of integer strings, we use UvA Trilearn base team [7] as a fixed opponent team. Thus, each integer string plays a soccer game against the UvA Trilearn team for our ES-type generation update.

We performed the proposed evolutionary computation for 300 generations. Table 1 shows the simulation results. We examined the scores of the best integer string at 0-, 100-, 200-, and 300-th generations. Each elite integer string played a soccer game against the UvA Trilearn basic team ten times. From Table 1, we can see that the performance of integer strings becomes better as the number of generations increases.

Table 1. Simulation results with five strings

Generation	Win	Lost	Draw	Average goals	Average goals lost
0	1	9	0	0.3	2.8
100	2	4	4	1.1	1.5
200	3	5	2	1.1	1.2
300	7	2	1	1.5	1.0

5 Conclusions

In this paper, we proposed an evolutionary method for acquiring team strategies of RoboCup soccer players. The action of soccer players that keep the ball is determined by the proposed method. The antecedent part of the action rules includes the positions of the agent and its nearest opponent. The soccer field is divided into 48 subareas. The action of the agent is specified for each subarea. The candidate actions for the consequent part of the action rules consist in a set of ten basic actions such as dribble, kick, and shot. The strategy of a soccer team is represented by an integer string of the consequent actions. In the evolutionary process, one-point crossover, bit-change mutation, and ES-type generation update are used as evolutionary operators. The generation update is performed in a similar manner to the $(\mu+\lambda)$-strategy of evolution strategy. That is, best integer strings are selected from a merged set of current integer strings and new integer strings that are generated from the current integer strings by the mutation operation.

In the computer simulations in this paper, we examined the performance of our proposed method. The results of the computer simulations showed that the offensive performance is improved during the execution of the proposed evolutionary method.

References

1. RoboCup official page, http://www.robocup.org
2. T. Nakashima, M. Udo, H. Ishibuchi, "A Fuzzy Reinforcement Learning for a Ball Interception Problem," RoboCup 2003: Robot Soccer World Cup VII, in press.
3. K. Chellapilla, and D.B. Fogel, "Evolving Neural Networks to Play Checkers Without Relying on Expert Knowledge", *IEEE Trans. on Neural Networks*, Vol.10, No.6, pp.1382–3191, 1999.
4. K. Chellapilla, and D.B. Fogel, "Evolving an Expert Checkers Playing Program Without Using Human Expertise", *IEEE Trans. on Evolutionary Computation*, Vol.5, No.4, pp. 422–428, 2001.
5. S. Luke and L. Spector, "Evolving Teamwork andCoordination with Genetic Programming", *Proceedings of the First Annual Conference on Genetic Programming*, pp.150–56, 1996.
6. Bäck, T.: *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, NY (1996)
7. UvA Trilearn team, http://carol.science.uva.nl/~jellekok/robocup/index_en.html

Practical Extensions to Vision-Based Monte Carlo Localization Methods for Robot Soccer Domain

Kemal Kaplan[1], Buluç Çelik[1], Tekin Meriçli[2],
Çetin Meriçli[1], and H. Levent Akın[1]

[1]Boğaziçi University
Department of Computer Engineering
34342 Bebek, Ístanbul, Turkey
{kaplanke, buluc.celik, cetin.mericli, akin}@boun.edu.tr
[2]Marmara University
Department of Computer Engineering
Göztepe, Ístanbul, Turkey
tmericli@eng.marmara.edu.tr

Abstract. This paper proposes a set of practical extensions to the vision-based Monte Carlo localization (MCL) for RoboCup Sony AIBO legged robot soccer domain. The main disadvantage of AIBO robots is that they have a narrow field of view so the number of landmarks seen in one frame is usually not enough for geometric calculation. MCL methods have been shown to be accurate and robust in legged robot soccer domain but there are some practical issues that should be handled in order to maintain stability/elasticity ratio in a reasonable level. In this work, we presented four practical extensions in which two of them are novel approaches and the remaining ones are different from the previous implementations.

Keywords: Monte Carlo localization, Vision based navigation, mobile robotics, robot soccer.

1 Introduction

Monte Carlo Localization (MCL) [1], [2] or *particle filtering* is one of the common approaches to this problem. This approach has been shown to be a robust solution for mobile robot localization; especially for unexpected movements such as ”kidnapping” [3]. The practical steps needed to make MCL reliable and effective on legged robots using only vision-based sensors, which have a narrow field of view, are presented in this paper. Although it has been applied to legged robots using vision-based sensors in the past [4], [5], [6], the works described in this paper contribute novel enhancements that make the implementation of particle filtering more practical. This work is done as a part of the Cerberus Project of Bogazici University [7] in the Sony four legged league which is one of the subdivisions of the RoboCup [8] in which two teams each consisting of four Sony AIBO robotic dogs compete against each other. The game area is 6m by 4m and four unique bi-colored beacons are placed in order to provide information for localization. In this work, we have used Sony AIBO ERS-210 robots with 200 MHz processor as our testbed. The organization of the rest of the paper is as follows: Brief information about MCL is given in Section 2. In Section 3, the proposed approach is explained in detail. Section 4 contains the results and we conclude with the Section 5.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 624–631, 2006.

2 Monte Carlo Localization

The idea behind MCL is to represent the probability distribution function for posterior belief about the position $Bel(l)$ as a set of samples drawn from the distribution. The samples are in the format $(x, y, \Theta), p$ where (x, y, Θ) is the position and orientation of the sample and p is the discrete probability associated with the sample denoting the likelihood of being at that position of the sample. Since $Bel(l)$ is a probability distribution, sum of all p_i should be equal to 1. Belief propagation is done in terms of two types of updates. When a motion update is performed, the new samples, based on both the old ones and the provided motion estimation, are generated to reflect the change in robot's position. In the motion update, the samples are displaced according to the relative displacement fed by odometry while taking the odometric error into account. For the observation update, the p values of each particle (l, p) is multiplied by $P(s|l)$ which is the probability of receiving sensor reading s assuming that the robot is located at l. Then, updated p values are normalized to maintaining $\sum p_i = 1$.

The second phase of MCL is the so-called resampling phase. In this phase, a new sample set is generated by applying fitness-proportionate selection or the survival of the fittest rule. The number of instances of a particle in the next generation is determined by the formula

$$K = \frac{N.p_i}{\sum_j p_j} \tag{1}$$

The particles with higher p values are more likely to be selected and copied to the new sample set. As a result of this, the particles will eventually move to the locations where the robot is more likely to be located at. In the second step of resampling, particles in the new sample set are moved according to their probabilities. The amount of movement for a particle instance is determined with the formula

$$x_i^{T+1} = x_i^T.(1 - p_i^T).Rnd(-1, 1).\Delta_{trans} \tag{2}$$

$$y_i^{T+1} = y_i^T.(1 - p_i^T).Rnd(-1, 1).\Delta_{trans} \tag{3}$$

$$\Theta_i^{T+1} = \Theta_i^T.(1 - p_i^T).Rnd(-1, 1).\Delta_{rot} \tag{4}$$

where, Rnd returns a uniform random number in the range $[-1, 1]$ and Δ_{trans} and Δ_{rot} are translational and rotational constants, respectively. The amount of movement is inversely proportional to the probability so the particles with higher probabilities will have a smaller move than particles with small probabilities.

3 Proposed Extensions

3.1 Considering the Number of Landmarks Seen

The number of landmarks used in the localization process has an important role in determining the actual position and orientation of the robot accurately. The accuracy of the estimated position and orientation of the robot increases with the increasing number of landmarks seen in a single frame. When calculating the confidence on each particle, each landmark contributes to the confidence by its angle and distance similarity. However, this approach results in an undesired output as the number of landmarks increases. For example, seeing a single landmark having a probability of 0.75 seems to provide a better estimation than four landmarks each having 0.9 probability which results in 0.9 x 0.9 x 0.9 x 0.9 = 0.6561 confidence. In order to avoid this misinterpretation, confidence is calculated in such a way that increasing number of landmarks increases the confidence. The formula used for calculating the confidence is

$$confidence = p^{(5-N_{percepts})} \tag{5}$$

where, $N_{percepts}$ is the number of the percepts seen. Since the maximum number of landmarks that can be seen in a single frame is 4, $p^{5-4} = p$ assigns the current value of the probability to the confidence which is the highest possible value.

3.2 Using Inter-percept Distance

In the Figure 1 w_1 and w_2 are the perceived width (in pixels) of the static objects. Similarly h_1 and h_2 are the heights of the static objects. The distances d_1 and d_2 can be calculated from these values by using a special function. This special function is calculated by fitting one polynomial or partial polynomials on the top of the experimental data. When the distances are known, the calculation of the orientation is relatively simple. The angles α and β can be used to find the orientation of the robot. Under ideal conditions, where there is no noise and the vision is perfect, d_1, d_2, α and β values are enough to find the current configuration of the robot. However, there is noise and it affects the distance calculations dramatically. In our case, where the resolution is 176x144, two-pixel error in the width of the static objects causes less than one-centimeter distance error for near objects. But, for far objects, two-pixel error may cause up to 50 cm distance error. As a solution, the distances between the perceived static objects are used. This measure both reduces the sensitivity to noise and provides additional information for localization. To use the distance between the static objects, the difference of estimated and expected distances cannot be used directly, because of perspective.

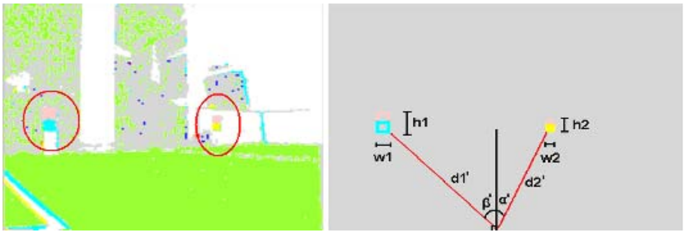


Fig. 1. Relative distances and orientations. (a) Classified image. (b) Recognized static objects.

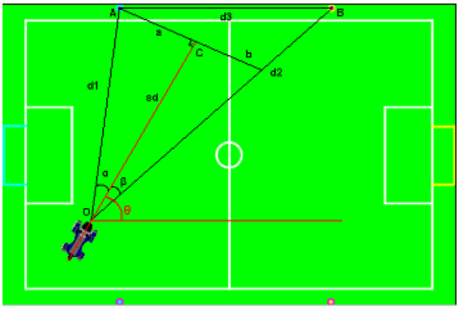


Fig. 2. Calculation of Distance Projection

As shown in Figure 2 the estimated distance should be compared with $a + b$, but not d_3. As an example, suppose that the static objects are at (s_x^1, s_y^1) and (s_x^2, s_y^2). For a given robot configuration $(x, y, \varTheta)$, the following equations are used to calculate a and b.

$$d_1 = \sqrt{(sx_1 - x)^2 + (sy_1 - y)^2} \tag{6}$$

$$d_2 = \sqrt{(sx_2 - x)^2 + (sy_2 - y)^2} \tag{7}$$

$$\alpha = arcTan(\frac{sy_1 - y}{sx_1 - x}) - \theta \tag{8}$$

$$\beta = arcTan(\frac{sy_2 - y}{sx_2 - x}) - \theta \tag{9}$$

$$s_d = cos(\beta).d_2, d_1 > d_2; cos(\alpha).d_1, o/w \tag{10}$$

$$a = tan(\alpha).s_d \tag{11}$$

$$b = tan(\beta).s_d \tag{12}$$

where s_d is the scene distance, which is the projection line (or plane in 3D) distance to the robot. Furthermore, $(a + b)$ is the distance between the two static objects on the projection plane at s_d. After those calculations, we have two values to compare. The first one is the distance between two static objects, calculated from the captured image, which is in pixels. The other one is $(a + b)$ which is in mm. In our work, we used the ratio of each distance to the width of the image, instead of, converting the units. The first ratio is trivial.

$$visionRatio = \frac{d_s}{w_{img}} \tag{13}$$

where, d_s is the distance between static objects and, w_{img} is the captured image width. However the second ratio depends again on s_d. In addition, horizontal field of view (FoV) of the camera is also used to calculate the width of the projection line or plane. For AIBO ERS-210's horizontal FoV is 57.6 degrees. Finally, the effect of the distance between the static objects to the overall configuration probability is calculated as follows,

$$expectedRatio = \frac{(a+b)}{tan(FoV/2).2.s_d} \tag{14}$$

$$p_d = \frac{1}{1+e^{(-40.\Delta_{ratio}.0.875)}} \tag{15}$$

$$\Delta_{ratio} = \frac{|min\{visionRatio, expectedRatio\}|}{|max\{visionRatio, expectedRatio\}|} \tag{16}$$

At the end of each iteration of MCL, for each configuration these p_d values, which are calculated from distances, orientations and other possible sources, are multiplied to find the final confidence of the configuration. Since the effects of rotation of the camera and the orientation of the robot are handled in the object recognition subsystem, this representation works well. However, we assume that the heights of the static objects and the robot are equal. In addition, we only consider the horizontal difference while estimating the distance between static objects.

3.3 Variable-Size Number of Particles

In MCL the number of particles, which are candidate configurations for current position, is one of the most important parameters. If the unnecessarily large amount of particles used, the process slows down dramatically. On the other hand, if the number of particles is too small, the system converges to a configuration very slowly, or cannot converge at all. One possible solution is to fix the number of particles to a constant for which the processing speed is moderate and the localization converges in a reasonable amount. But in this scenario, when the localization starts to converge, which means the candidate configurations are similar to each other, most of the processing is unnecessary.

In our localization system, the number of particles is assigned to the maximum number of particles allowed. This maximum number is the lowest particle count for which the localization system converges in a reasonable time, for nearly all cases. But still it is a large number. During the execution of the localization system, the number of particles is reduced if the confidence about the position of the robot increases. Which means, the system searches fewer configurations if it is certain about its position, as shown in Figure 3. Similarly, when the confidence about the current pose of the robot decreases, the number of particles increases, which means the search for a better configuration speeds up. This oscillation continues if the confidence for the current pose is above a

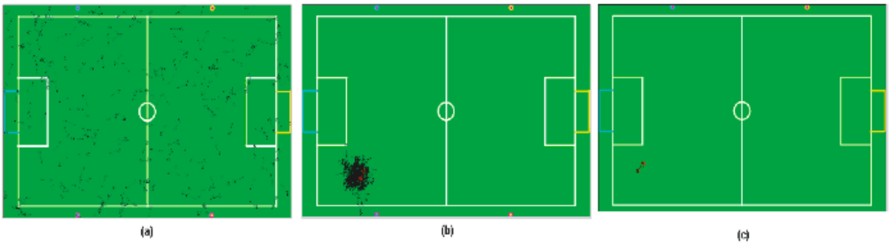


Fig. 3. Number of particles changes while (a) searching, (b) converging and (c) converged

specific constant. Otherwise, which means the robot is lost, the confidence is set to zero and the entire process is restarted. The overall process can be modeled by the following equation,

$$N_{par} = K.p, p > T; K, o/w \tag{17}$$

where N_{par} is the number of particles, which are candidate configurations, K is the maximum number of particles and p is the confidence of the current pose. Finally, T is the threshold value for reseting the system to the lost state.

3.4 Dynamic Window Size Based on Pose Confidence for Resampling

In the earlier approaches, when the confidence on the current position decreases below a threshold value, the positions and orientations of each particle are reset and the particles are distributed over the entire field randomly. However, each reset operation requires processing a large number of particles over a large area. In order to solve this problem, a window, in which the particles will be distributed, is constructed around the representative particle. The size of this window is inversely proportional with the confidence value of the representative particle, and the number of particles that will be distributed in this window is directly proportional to the size of the window. That is, when the confidence on the representative is high, the window constructed around the representative and the number of particles that will be used is small. If a reset operation fails to include particles having significant probability values, the size and the number of particles that will be used in the next reset operation are gradually increased. This approach provides a faster convergence since a smaller number of particles in a smaller area are processed. Figure 4 illustrates the situation in which the robot is kidnapped from the blue goal into the yellow goal. Convergence is seen after 5 frames, and after 11 frames the robot finds its position and orientation.

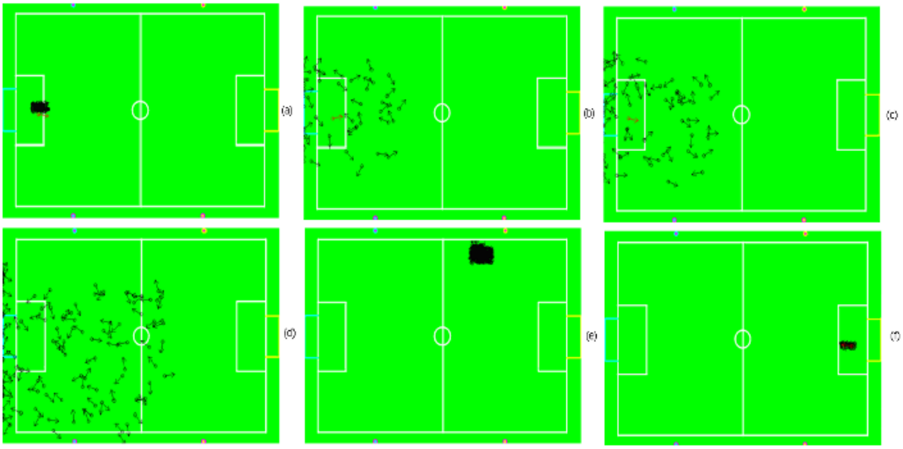


Fig. 4. (a) Just after being kidnapped, (b) first frame after kidnapping, (c) after 2 frames, (d) after 3 frames, (e) after 4 frames, and (f) after 11 frames the robot finds its position

4 Results

To test the efficiency of the extensions, we have performed two different experiments. In both experiments, extended MCL is compared with the original implementation. To test both the converge time and the estimation error, the robot is placed to the upper left corner of the field in order to provide enough number of visual percepts. Localization process is terminated if 95 percent confidence is achieved or the process iterates 200 times. The iteration counts for converging to a point and distance error of that point to the actual position are given in Table 1.

Table 1. Convergence time and error ratios for Extended MCL vs. Standart MCL

	Number of Iterations	**Distance Error**
Standart MCL	69.80±89.90	20.47±10.17
Extended MCL	37.90±37.23	10.83±6.51

According to the results, extended MCL reduces the iteration count to converge a configuration and the error of this configuration nearly by 50 percent. Since the standard MCL fails to converge for some iteration, the iteration count is high than extended MCL in average.

In the second experiment, we tested the convergence speed of the original and extended implementations in case of kidnapping. The robot is moved from where the localization system is converged to a point farther away and the re-convergence time is logged. The results can be seen in Table 2.

Table 2. Re-convergence time (number of frames) in case of kidnapping for Extended MCL vs. Standart MCL

Kidnapping Distance (mm)	**Extended MCL**	**Standart MCL**
$\tilde{3}550$	26	35
$\tilde{2}280$	9	15
$\tilde{1}760$	11	39

5 Conclusions

Autonomous self localization is one of the key problems in mobile robotics research and have been addressed many times with proposed many different approaches. Robot soccer is a good test bed for many aspects of the mobile robotics research such as multi-agent systems, computer vision, self localization and effective locomotion with its highly dynamic and partially observable nature.

In this work, we have proposed four practical extensions to the vision-based MCL for legged robots. Using variable number of particles is not a new approach, but our implementation has no extra computational requirement as the other implementations (i.e. determining the number of particles proportional to the variance of confidences in sample set). Using inter-percept distance in addition to distances and bearings to the percepts is a novel approach and the results are quite satisfactory. Also, considering the number of percepts seen while calculating the pose confidence is a novel approach and allows the observations with high number of percepts have higher effect on the confidence, in other words, the more number of percepts seen, the more reliable the observation is. Again, using a subset of the state space for resampling when our belief about the current pose is over a threshold is not a new idea but the our way of window size and position determination for resampling is novel.

Acknowledgments

This project is supported by Boğaziçi University Research Fund project 05A102D.

References

1. S. Thrun, D. Fox, W. Burgard, and F. Dellaert, ”Robust Monte Carlo Localization for Mobile Robots”, Journal of Artificial Intelligence, 2001.
2. S. Thrun, ”Particle Filters in Robotics”, In The 17th Annual Conference on Uncertainty in AI (UAI), 2002.
3. D. Fox, W. Burgard, and S. Thrun, ”Markov Localization for Mobile Robots in Dynamic Environments” *Journal of Artificial Intelligence Research*, 11, pp. 391-427, 1999.
4. C. Kwok and D. Fox, ”Map-Based Multiple Model Tracking of a Moving Object” *RoboCup 2004: Robot Soccer World Cup VIII. LNCS 3276*, pp.18-33,2005.
5. S. Lenser and M. Veloso, ”Sensor Resetting Localization for Poorly Modelled Mobile Robots” *Proceedings of ICRA 2000, IEEE,* pp. 1225-1232 2000.
6. T. Rofer and M. Jungel, ”Vision-Based Fast and Reactive Monte-Carlo Localization” *Proceedings of IEEE International Conference on Robotics and Automation, Taipei, Taiwan,* pp.856-861, 2003.
7. H. L. Akin, et al., ”Cerberus 2003” Robocup 2003: Robot Soccer World Cup VII, The 2003 International Robocup Symposium Pre-Proceedings, June 24-25, 2003, Padova, pp.448.
8. Robocup Organization ”http://www.robocup.org”

Robocup Rescue Simulation Competition: Status Report

Cameron Skinner and Mike Barley

Department of Computer Science
The University of Auckland
New Zealand

Abstract. This is the fifth anniversary of the Robocup Rescue Simulation Competitions and the tenth anniversary of the disaster that inspired the Competitions. This is a good time to take stock of what milestones have been achieved and what milestones we should be aiming for. Specifically, this paper looks at the goals that led to the establishment of the competition, the current status of the simulation platform and infrastructure, and finally suggests areas of the current simulation platform which should be improved and parts of the Robocup Rescue technical and social infrastructure which should be extended.

1 Introduction

2005 is the tenth anniversary of the devestating Hanshin-Awaji earthquake in Japan which killed more than 6000 people in Kobe city. The Robocup Rescue project was established in 2000 in response to this disaster in order to provide a system for facilitating research into disaster mitigation and search and rescue (SAR) problems [1]. The goal of Robocup Rescue is to ultimately help save lives and provide an important public good, especially in the light of recent natural disasters such as the Boxing Day tsunami, the 2004 Bam earthquake, and even minor disasters such as the recent flooding in parts of New Zealand.

This paper is organised as follows. Section 2 revisits the goals stated when the Robocup Rescue Simulation Competition was established. In Section 3, we review the progress made towards those goals in the last five years and whether the competition is moving in the right direction. We then consider extensions that could be considered in order to move the project towards achieving those goals. Section 4 discusses development of the simulator software. Section 5 considers ways to improve the collaborative research effort of the Robocup Rescue community and to link this research with real-world emergency services and related industries. Section 6 concludes.

2 Goals

The Robocup Rescue Simulation project was started with several goals in mind. The high-level objectives of the project are [1]:

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 632–639, 2006.

1. To apply agent technologies to social problems in a way that contributes to human social welfare.
2. To provide a practical problem for development of novel research in multi-agent systems and artificial intelligence.
3. To promote international research collaboration via the Robocup competition.

The Robocup Rescue Simulation project "aims to be a comprehensive urban disaster simulator" [1] by modelling the state of roads, buildings and individuals in a city after a disaster. Ultimately, this simulator could be used for training, testing of emergency management plans and for real-time command and control in a real emergency situation.

These are ambitious goals so it is unreasonable to expect them to have been achieved in only five years, but we need to know how far we have come and where to go next.

3 Progress to Date

The Robocup Rescue Simulation project has successfully implemented an agent-based urban disaster simulator that allows researchers to experiment with different strategies for responding to the disaster. The software simulates an earthquake in an urban environment and handles building collapse, road blockage, traffic flow, fire and civilian behaviour, as well as the response of the emergency services [1]. Unfortunately, the granularity of the simulation is still rather coarse. Buildings are represented as polygons and classified as either wood, concrete or steel rather than having different types of buildings (such as office blocks, residential housing, hazardous buildings (e.g. petrol stations), factories, warehouses to name a few) with different features. Every fire brigade agent has exactly the same capabilities, rather than having some fire trucks with ladders, some with bigger water tanks and so on.

The scale of the simulation is also quite small at present. Maps are generally limited to less than 1000 buildings, and there are usually no more than 100 civilians and 50 emergency services agents. The communication model is simplistic and does not reflect important characteristics of real-life communications.

Some of these limitations are due to computer processing requirements - in order to add more civilians we need to be able to simulate their behaviour - which will become less of a problem over time as computer power increases. However, there are also some design limitations that need to be addressed. These are listed in Section 4.4.

Despite the small scale and low level of detail in the simulation a great deal of progress has been made in the last five years. The competition has been succesfully run every year since it began and the size of the community is growing steadily.

[1] Currently limited to fire, police and ambulance teams.

3.1 Simulation Platform

Development of the simulation platform has been fairly rapid, given the distributed nature of the community. The simulator kernel is fast and stable, and simulator modules for traffic flow, fire spread, building collapse, road blockages and civilian behaviour have all been implemented. Development is continuing in most of these areas.

In 2004 a new fire simulator was developed by the ResQ Freiburg team [2]. This simulator is a vast improvement over the old one and models the spread of fire much more accurately than before. It also increases the level of realism by allowing fire brigades to “pre-extinguish” buildings by pouring water on them to prevent a nearby fire spreading. In addition, buildings will re-ignite if left next to another fire for too long.

Development of a library of communication functions and other useful tools has started and will make maintenance of the system much easier in future.

3.2 Infrastructure

The most significant achievements have been in the community infrastructure. Organising and technical committees run each year’s competition and steer the direction of development of the league, and a manual [1] has been written which describes the simulator and provides a guide for new developers.

In 2004 a secondary competition was added, the infrastructure competition, to promote development of new simulator implementations and development toolkits. This has already resulted in a new fire simulator developed by the ResQ Freiburg team [2] and 11 teams have pre-registered for the 2005 infrastructure competition.

An open source development model has been adopted for simulator development and all simulator code is controlled by CVS [3] on sourceforge.net [4]. The open source model allows for rapid development of interacting simulator components and also gives teams the ability to verify that simulator components perform as specified.

Finally, several development kits have been written that provide toolkits for developers that want to write agents for use in the Robocup Rescue Simulation competition. These include a C/C++ toolkit (the Agent Development Kit (ADK) [5]), and two java implementations: YabAPI [6] and Rescuecore [7].

4 Proposed Future Simulation Platform Developments

The simulator platform still requires a significant amount of development before it becomes a fast, fine grained, realistic simulation of an urban disaster environment. We have broken the simulator development into four modes: scale, detail, information and communication, and design.

4.1 Scale

Currently the simulator operates on a small scale, in the order of 1km^2 of urban space composed of approximately 1000 buildings and a similar number of road segments. The number of civilians and emergency services are limited to around 100 and 50 respectively. This obviously does not simulate reality very closely. The major difficulty with increasing the scale of the simulation is the increased computation time required to perform calculations.

The scale of the simulation needs to be increased in future, in terms of the size of the area simulated, the number of entities in the simulation, and in the time scales involved.

4.2 Detail

The level of detail in the simulation is also somewhat limited. Currently there are only three types of rescue agent - fire brigades, ambulance teams and police forces - and every instance of a type is identical to every other agent of the same type. In reality there are several types of fire truck, and agents have a diverse set of capabilities.

The representation of the world is at a low level of detail. For example, there is no way to specify that any building is more important than any other building. In real life it is clearly more important to save the hospital from fire than a single house. Similarly, there is no way to specify specific hazards such as petrol stations. Since, in real life, we would want to prevent a large fire from engulfing a petrol station it would be beneficial if this could be modelled in the simulation.

4.3 Information and Communication

The competition should attempt to move towards more realistic modelling of the knowledge that can be expected to be available to emergency service planners. Currently in the simulation, the emergency services have no *a priori* knowledge about the distribution of the population at the time of the disaster. In real life, there is *a priori* knowledge about how the population will be distributed at particular times. For example, the business district will likely be heavily populated on a Tuesday afternoon and almost deserted early on a Sunday morning. This sort of *a priori* knowledge would be useful for planning how to respond to disasters and would be available to real-life emergency service planners.

Similarly, the competition should attempt to move towards more realistic modelling of the communication environment. For example, in many large cities there are large microwave towers that handle much of the emergency services' communication channels. If those towers go out, much of the wireless communication disappears. Like hospitals, these towers would be important to save from fire, etc. If our simulation platform is to be useful for reasoning about how to respond to disasters, capturing these communicational dependencies would be important.

Finally, additional available sources of information need to be considered. Most modern cities have a large network of CCTV cameras for security or traffic monitoring, fire alarms and (in industrial buildings) hazardous material sensors and alarms. The addition of these and similar sources of information would increase the realism of the simulation.

4.4 Design

Compounding the limitations on scale, detail, information and communication are issues with the design of the software. The architecture has some limitations that make it difficult to add new features, and the quality of the code is quite poor in many cases. This is slowly improving as people in the community replace older modules with new ones, but there is a strong need for more quality control and better managed development. The adoption of an open source model using CVS [3] as a source control mechanism will hopefully improve matters in the future.

A full analysis of the existing software and a detailed code review would be highly beneficial as a large amount of code appears to be duplicated in each module and could easily be put into a seperate library. This process has started but is a long way from completion.

In the longer term, it would be useful to extend the software design in such a way that descriptions of the world could be made using a modelling language of some kind. Currently the abilities that agents have, the communication model and the organisational structure are hard-coded. It would be beneficial if it was possible to specify at runtime what kind of organisational structure to use, or to allow dynamic structures, for example to allow the formation and dissolution of teams during the course of the simulation. Similarly, being able to specify what equipment and/or capabilities each agent possesses would be useful.

5 Proposed Future Infrastructure Developments

5.1 Community Development

The Robocup Rescue community is somewhat fragmented at present. Although there are teams from all over the world competing there is little collaborative research or development. A project has been established on sourceforge.net that will hopefully encourage more participation in the development of simulator components, but a spirit of cooperation needs to be fostered within the community. The competitive nature of the simulation competition, while pushing researchers to produce better solutions, has the unfortunate side effect of encouraging teams to be secretive with their ideas and code.

Having a common "bulletin board" for the presentation of questions, ideas and contacts would make it easier for new researchers to become involved in the field and would also contribute to more of a "community feeling". Having a steering committee to guide the direction of development of the simulator and production of useful tools would also help to build the community.

Finally, the establishment of a program track or workshop at the annual Robocup symposium dedicated to Robocup Rescue would make it easier to consolidate the research that is being carried out at diverse institutions around the world.

5.2 Balancing Complexity and Accessibility

Improving the simulation so that it closely approximates reality raises an interesting dilemma: how can we balance the increasing complexity of the system with the need to keep it accessible enough for new researchers to become involved? A beginning team already has difficulties developing the most basic of agent implementations due to the complexity of the communication system and problem domain. The learning curve will only become steeper as the simulator becomes more realistic.

There is, therefore, a strong need to produce supporting code libraries at the same time as new simulator developments appear, as well as continuing the existing practice of asking teams to release their source code after every competition. The more tools that are available for teams, such as standard search algorithms, useful abstractions of the simulated world and communication libraries, the easier it will be for a new team to enter the competition. Of course, documentation will also be required if these tools are to be of any practical use.

5.3 Industry Development

Another area that needs development is the establishment of links with industry and government organisations. Clearly developing a detailed urban disaster simulation will be of little practical use if the real emergency services cannot apply it to their own activities. In addition, without input from the people who manage disaster risk professionally it is unlikely that the Robocup Rescue community will develop a simulator that is realistic. Discussions with the New Zealand Police [8] have already shown one popular misconception: after a disaster such as an earthquake, most people do not panic and attempt to flee the city. Instead, experience with real disasters has shown that survivors generally begin helping with the rescue effort almost immediately [9].

Development of industry links serves two main purposes:

1. Input from industry will ensure that the software developed accurately reflects what goes on in the real world.
2. Developing tools based on the simulator that industry can use will be beneficial to both the Robocup Rescue community and to real emergency services.

The ultimate goal of providing a system that can be used for training, testing plans and provision of real-time command and control support will never be realised unless the end users - the real emergency services - have an input from early on.

5.4 Development of a Roadmap

As we have seen several times in the last few years, disaster can strike almost anywhere and affect large numbers of people. Part of the appeal of the Robocup Rescue project is that it has the potential to help people. The best way to "contribute [to] human social welfare" [1] is for the competitions to help push the research towards something we can offer to emergency services.

The Robocup Rescue project could help in this respect by developing a roadmap of agent-based technology advances that might make contributions to emergency services. The roadmap could indicate milestones that would mark our field's progress towards points of advancement and would suggest how the competition might be expected to evolve. These milestones might be in the form of challenge problems.

The development of this roadmap should be a co-operative effort between researchers and emergency service managers. Ideally the emergency service managers would come from both national and international (e.g., the United Nations) agencies. One suggestion is that the Robocup Rescue Simulation Organising Committee look into the formation of a permanent Steering Committee whose members would be involved in organising the development of the roadmap and in monitoring the evolution of the competition according to that roadmap.

6 Conclusion

The Robocup Rescue Simulation League has come a long way since it was established 5 years ago. The software has been developed from scratch to include simulators for traffic flow, fire spread, building collapse, road blockage and civilian behaviour as well as the simulator kernel that binds these components together. Development is continuing on all aspects of the simulator platform, including new simulators and the development of libraries that will make development and maintenance of the simulator components much easier in future. There is still, however, a large amount of work to be done before the goal of providing a comprehensive urban disaster simulator can be achieved, most notably in increasing both the scale and the level of detail of the simulations. The introduction of the infrastructure competition in 2004 has encouraged development of high-quality simulator components and toolkit implementations.

The Robocup Rescue Simulation community is steadily growing in size. The main challenge at present is to provide a collaborative environment that makes it easier for researchers in different parts of the world to share information, find out what other researchers are doing, and to get help when they need it. The provision of a permanent website with "bulletin boards" and forums would go a long way towards meeting this challenge.

To help move the Robocup Rescue project effectively towards its goal of being able to help society better deal with large-scale disasters, we make the following suggestions:

- The Robocup Rescue Organising Committee set up a Steering Committee.
- The Robocup Rescue Steering Committee be responsible for developing a roadmap for how the competition should evolve in order to realise the Robocup Rescue goals.
- The roadmap should be developed as a joint effort between researchers and emergency service agencies.
- Ideally the emergency service agencies would include both national and international agencies.

- The roadmap milestones might be a series of challenge problems that would represent different points in the evolution of the competition.
- The Steering Committee would be responsible for monitoring the evolution of the competitions with respect to the roadmap.

In another five years, it would be good to report that the Robocup Rescue Simulation platform and the research arising from the competition had led to technology that enabled emergency service agencies to better cope with some classes of large-scale disasters. It would be nice to be able to say that lives had been saved because of our work!

References

1. The Robocup Rescue Technical Committee: Robocup-Rescue Simulator Manual, http://sakura.meijo-u.ac.jp/ttakaHP/kiyosu/robocup/Rescue/manual-English-v0r4/index.html. (2000)
2. ResQ Freiburg: www.informatik.uni-freiburg.de/ rescue/ (2005)
3. CVS: www.cvshome.org (2005)
4. Sourceforge: www.sourceforge.net/projects/roborescue (2005)
5. Bowling, M.: www-2.cs.cmu.edu/ mhb/research/rescue/ (2005)
6. Morimoto, T.: ne.cs.uec.ac.jp/ morimoto/rescue/yabapi/ (2005)
7. Skinner, C., Teutenberg, J.: www.sourceforge.net/projects/rescuecore (2005)
8. New Zealand Police: www.police.govt.nz (2005)
9. Garth Stockley (New Zealand Police Risk Manager): Personal communication (2004)

Self Task Decomposition for Modular Learning System Through Interpretation of Instruction by Coach

Yasutake Takahashi[1,2], Tomoki Nishi[1], and Minoru Asada[1,2]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering
[2] Handai Frontier Research Center,
Osaka University
Yamadagaoka 2-1, Suita, Osaka, 565-0871, Japan
{yasutake, nishi, asada}@er.ams.eng.osaka-u.ac.jp
http://www.er.ams.eng.osaka-u.ac.jp

Abstract. One of the most formidable issues of RL application to real robot tasks is how to find a suitable state space, and this has been much more serious since recent robots tends to have more sensors and the environment including other robots becomes more complicated. In order to cope with the issue, this paper presents a method of self task decomposition for modular learning system based on self-interpretation of instructions given by a coach. The proposed method is applied to a simple soccer situation in the context of RoboCup.

1 Introduction

Reinforcement learning (hereafter, RL) is an attractive method for robot behavior acquisition with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [1, 2]. However, a simple and straightforward application of RL methods to real robot tasks is difficult because of the enormous time for exploration that scales exponentially with the size of the state/action space. The recent robots tend to have many kinds of sensors like normal and omni-vision systems, touch sensors, infrared range sensors, and so on. They can receive a variety of information from these sensors, especially vision sensors. This fact indicates that the difficulty of RL application to real robot tasks becomes more serious.

Fortunately, a long time-scale behavior might often be decomposed into a sequence of simple behaviors in general, and therefore, the search space can be divided into smaller ones. In the existing studies, however, task decomposition and behavior switching procedures are given by the designers (ex. [1, 3, 4]). Others adopt the heuristics or the assumption that are not realistic from the view point of real robot application (ex. [5, 6, 7, 8]).

When we develop a real robot that learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 640–647, 2006.

the robot to accelerate the learning before it starts to learn (ex. [9, 10]). This idea was applied to a monolithic learning module. To cope with more complicated tasks, this idea can be extended to a multi-module learning system. That is, the instruction will help a learner to find useful subtasks.

In this paper, we introduce an idea that the capability of a learning module defines the size of subtasks. We assume that each module can maintain a few number of state variables and this assumption is reasonable for real robot applications. Then, the system decomposes a long-term task into short-term subtasks with self-interpretation of coach instructions so that one learning module with limited computational resources can acquire a purposive behavior for one of these subtasks. We show experimental results with much more sensors such as normal and omni-vision systems and 8 directions infrared range sensors.

2 Basic Idea

There are a learner and a coach in a simple soccer situation (Fig. 1). The coach has *a priori* knowledge of a task to be played by the learner while s/he does not have any idea about the system of the learner. On the other hand, the learner just follows the instructions without any knowledge of the task. After some instructions given by a coach, the learner decomposes the whole task into a sequence of subtasks, acquires a behavior for each subtask, and coordinates these behaviors to accomplish the task by itself . In Fig. 1, the coach instructs an shooting a ball into a yellow goal with obstacle avoidance. Fig. 2 shows an example that the system decomposes this task into three subtasks and assigns them to three modules that maintain state spaces consist of ball variables, opponent and goal ones, and goal ones, respectively.

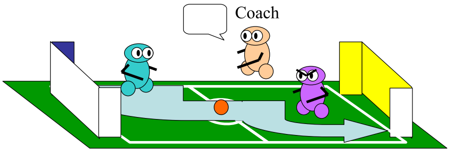


Fig. 1. A coach gives instructions to a learner

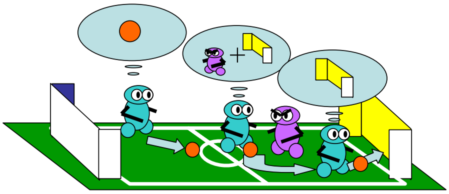


Fig. 2. The learner follows the instructions and finds basic behaviors by itself

Fig. 3 show a rough sketch of the idea of the task decomposition procedure. The top of the Fig. 3 shows a monolithic state space that consists of all state variables $(x_1, x_2, \cdots, x_n)$. The red lines indicate sequences of state value during the given instructions. As we assume beforehand, the system cannot have such a huge state space, then, decomposes the state space into subspaces that consist of a few state variables. The system regards that the ends of the instructions

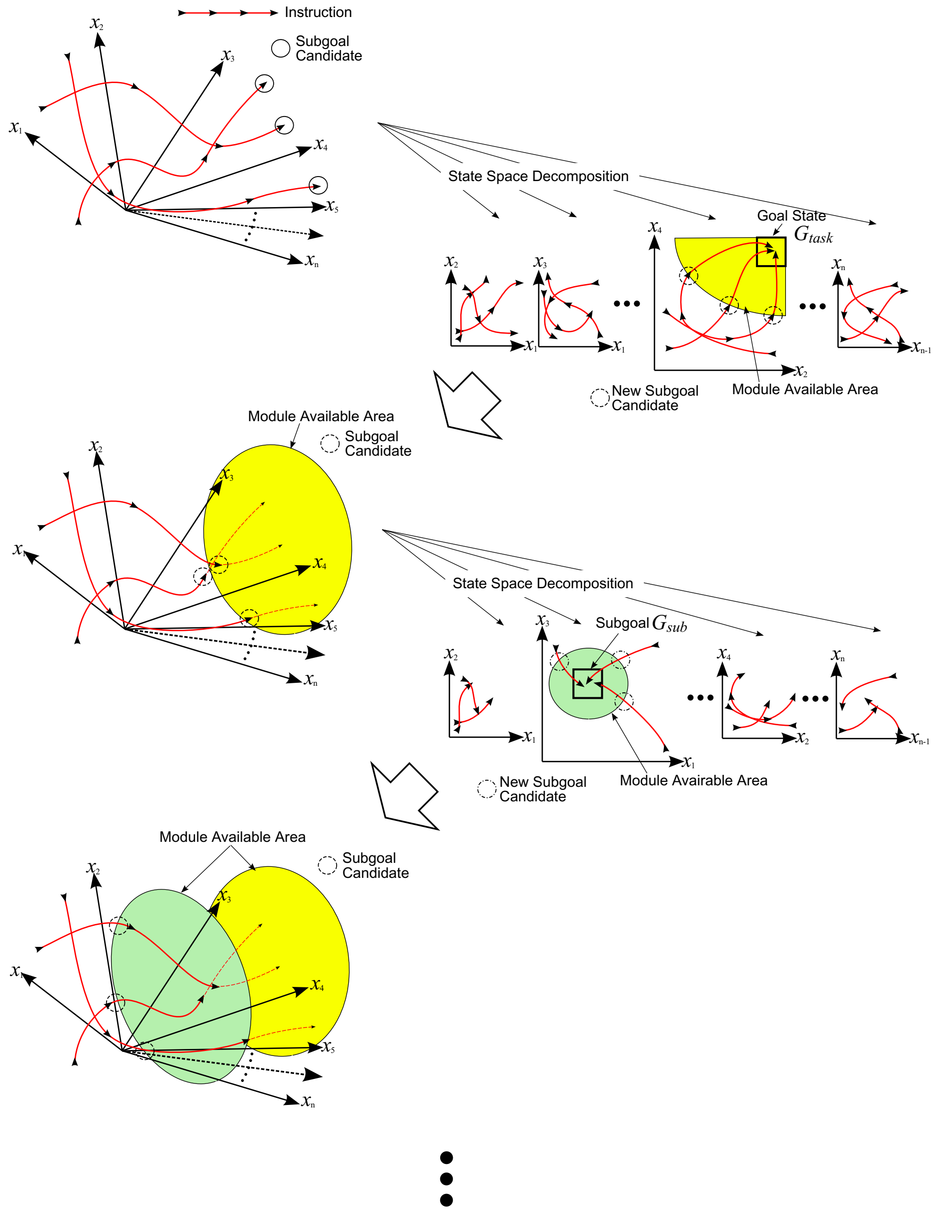


Fig. 3. Rough sketch of the idea of task decomposition procedure

represent goal states of the given task. It checks all subspaces and selects one in which the most ends of the instruction reach a certain area (G_{task} in Fig. 3). The system regards this area as the subgoal state of a subtask which is a part of the given long-term task. The steps of the procedure are as follows:

1. find module unavailable areas in the instructions and regard them as unknown subtask.
2. assign a new learning module.
   (a) list up subgoal candidates for the unknown subtasks on the whole state space.
   (b) decompose the state space into subspaces that consist of a few state variables.
   (c) check all subspaces and select one in which the subgoal candidates reach a certain area best (G_{sub} in Fig. 3).
   (d) generate another learning module with the selected subspace as a state space and the certain area as the goal state.
3. check the areas where the assigned modules are available.
4. exit if the generated modules cover all segments of instructed behaviors. Else goto 1.

3 Robot, Tasks, and Assumption

Fig. 4 shows a mobile robot we have designed and built. The robot has a normal camera in front of body, an omni-directional camera on the top, and infra red distance sensors around the body. Fig. 5 show the images of both cameras. A simple color image processing is applied to detect the ball, the goal, and an opponent in the image in real-time (every 33ms). The robot has also 8 directions infrared range sensors. The robot has totally 39 candidates of state variables. The details of the candidates are eliminated because of space limitations. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane). The tasks for this robot are chasing a ball, navigating on the field, shooting a ball into the goal, and so on. We assume that the given task has some absorbing goals, that is, the tasks are accomplished if the robot reaches to certain areas in state spaces which consist of a few state variables.

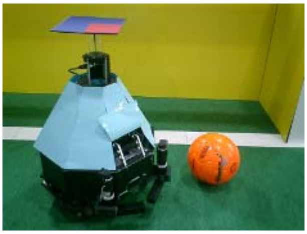


Fig. 4. A real robot

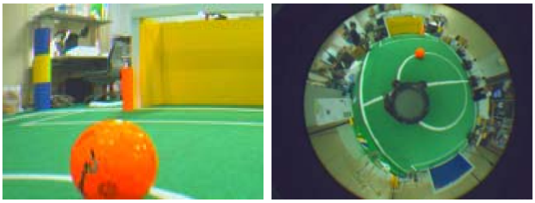


Fig. 5. Captured camera images

4 Availability Evaluation and New Learning Module Assignment

The learner needs to check the availability of learned behaviors that help to accomplish the task by itself because the coach neither knows what kind of behavior the learner has already acquired nor shows perfect example behaviors from the learner's viewpoint. The learner should suppose a module as valid if it accomplishes the subtask even if the greedy policy seems different from the example behavior. Now, we introduce AE in order to evaluate how suitable the module's policy is to the subtask:

$$AE(s, a_e) = \frac{Q(s, a_e) - min_{a'} Q(s, a')}{max_{a'} Q(s, a') - min_{a'} Q(s, a')} , \tag{1}$$

where a_e indicates the action taken in the instructed example behavior. AE becomes larger if a_e leads to the goal state of the module while it becomes smaller if a_e leaves from the goal state (see Fig. 6). Then, we prepare a threshold AE_{th}, and the learner evaluates the module as valid for a period if $AE > AE_{th}$. If there are modules whose AE exceeds the threshold AE_{th}, the learner selects the module which keeps $AE > AE_{th}$ for longest period among the modules (see Fig. 7). In Fig. 3, "Module Available Area" indicates the one in which $AE > AE_{th}$.

If there is no module which has $AE > AE_{th}$ for a period, the learner creates a new module which will be assigned to the subtask (see procedure 2 in 2). To assign a new module to such a subtask, the learner identifies the state space and the goal state. The system follow the two steps to select an appropriate state space and the goal state for the subtask:

- selection of one state variable that specifies the goal state, and
- construction of a state space including the selected state variable.

In order to find one state variable that specifies the goal state best, the system lines up the candidates for a goal region in term of state variables. On the other

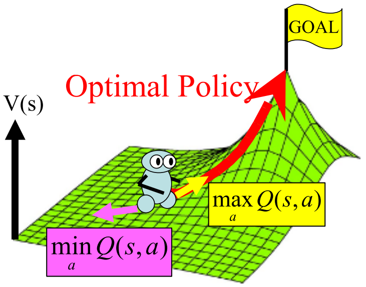


Fig. 6. Sketch of state value function and action value

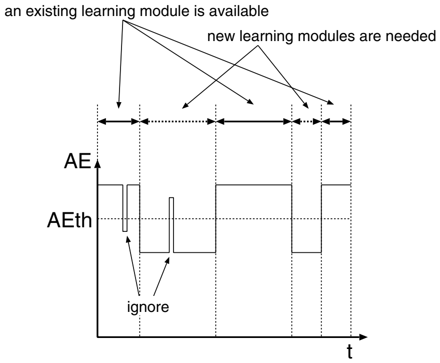

an existing learning module is available
new learning modules are needed
AE
AEth
ignore
t

Fig. 7. Availability identification during the given sample behavior

hand, in order to select another state variable, the system evaluates performance of Q value estimation.

The details of the procedure are eliminated because of space limitations.

5 Experiments

We have instructed the robot from a simple behavior (ball chasing) to a complicated one (shooting a ball with obstacle avoidance) in [11], however, there is a criticism that the step-by-step instruction implies task decomposition to the robot. Therefore we adopt only shooting behavior for the task decomposition and the coordination. Fig. 8 shows four examples of the behaviors instructed by the coach. The total number of instruction is 21 for this experiment.

According to the learning procedure, the system produced four modules for the instructed behaviors. The modules are $LM_1(A_{pb}, X_{pb})$, $LM_2(\theta_{og})$, $LM_3(Y_{ob}, X_{ob})$, and $LM_4(A_{ob}, \theta_{ob})$. For example $LM_1(A_{pb}, X_{pb})$ indicates that the modules

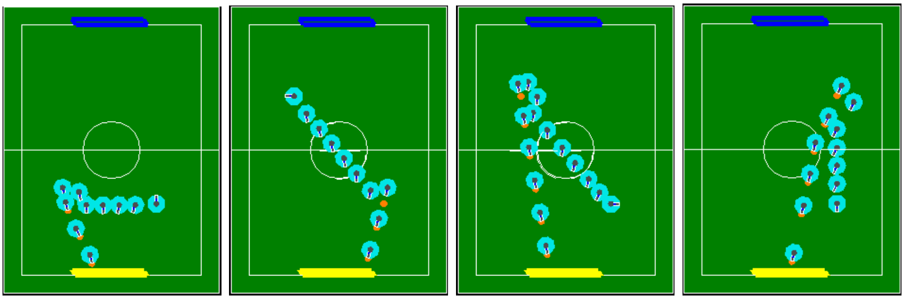


Fig. 8. Examples of Instructed behaviors

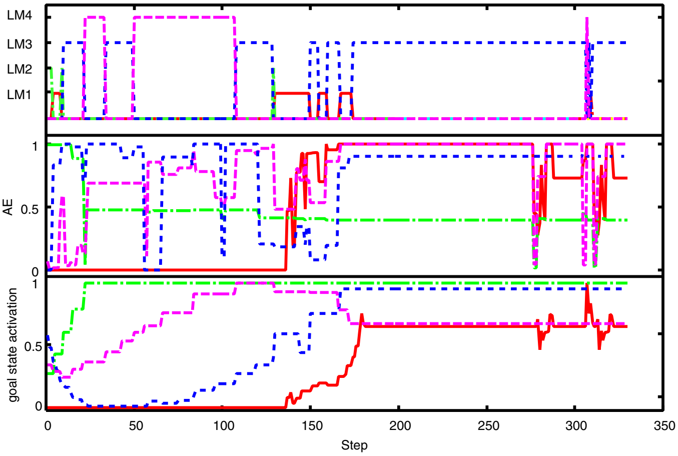


Fig. 9. Sequences of the selected module, availability evaluations and goal state activations of modules through an instruction

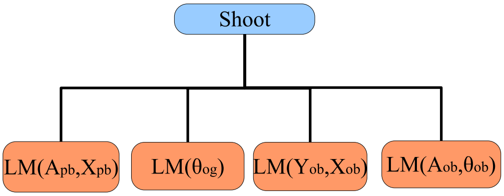


Fig. 10. Acquired hierarchy for the shooting behavior

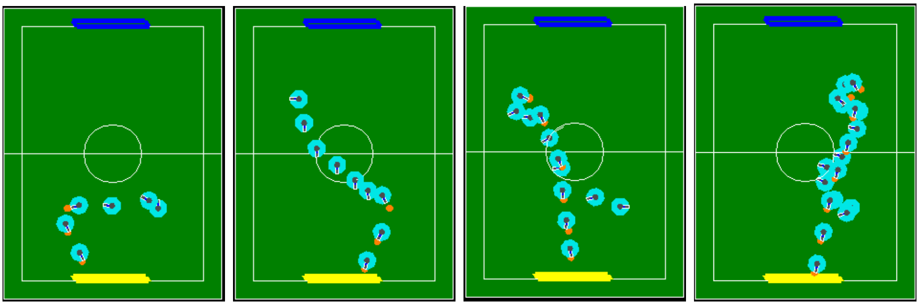


Fig. 11. Acquired behaviors for shooting task

has a state space that consists of the area of ball on the normal camera image (A_ob) and the x position of the ball on the normal camera image (X_{pb}). Fig. 9 shows sequences of the selected module, availability evaluations and goal state activations of modules through an instruction.

Fig. 11 shows the learned behaviors. The start positions of the behaviors are the same ones of the instructions for comparison. The trajectories of learned behaviors are different from the instructed behaviors. This fact indicates that the learner recognizes the subtasks based on its own modules, understands the objectives of the subtasks, and executes appropriate actions for them.

6 Conclusion

We proposed a hierarchical multi-module learning system based on self-interpretation of instructions given by a coach. We applied the proposed method to our robot and showed results for a simple soccer situation in the context of RoboCup.

References

1. J. H. Connell and S. Mahadevan, *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.
2. M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda, “Purposive behavior acquisition for a real robot by vision-based reinforcement learning,” *Machine Learning*, vol. 23, pp. 279–303, 1996.

3. P. Stone and M. Veloso, “Layered approach to learning client behaviors in the robocup soccer server,” *Applied Artificial Intelligence*, vol. 12, no. 2-3, 1998.
4. P. Stone and M. Veloso, “Team-partitioned, opaque-transition reinforcement learning,” in *RoboCup-98: Robo Soccer World Cup II* (M. Asada and H. Kitano, eds.), pp. 261–272, Springer Verlag, Berlin, 1999.
5. B. L. Digney, “Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments,” in *From animals to animats 4: Proceedings of The fourth conference on the Simulation of Adaptive Behavior: SAB 96* (P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, eds.), pp. 363–372, The MIT Press, 1996.
6. B. L. Digney, “Learning hierarchical control structures for multiple tasks and changing environments,” in *From animals to animats 5: Proceedings of The fifth conference on the Simulation of Adaptive Behavior: SAB 98* (R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, eds.), pp. 321–330, The MIT Press, 1998.
7. B. Hengst, “Generating hierarchical structure in reinforcement learning from state variables,” in *6th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)* (R. Mizoguchi and J. K. Slaney, eds.), vol. 1886 of *Lecture Notes in Computer Science*, Springer, 2000.
8. B. Hengst, “Discovering hierarchy in reinforcement learning with HEXQ,” in *Proceedings of the Nineteenth International Conference on Machine Learning (ICML02)*, pp. 243–250, 2002.
9. S. D. Whitehead, “Complexity and cooperation in q-learning,” in *Proceedings Eighth International Workshop on Machine Learning (ML91)*, pp. 363–367, 1991.
10. M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, “Vision-based reinforcement learning for purposive behavior acquisition,” in *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 146–153, 1995.
11. Y. Takahashi, K. Hikita, and M. Asada, “Incremental purposive behavior acquisition based on self-interpretation of instructions by coach,” in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 686–693, Oct 2003.

Specifications and Design of Graphical Interface for Hierarchical Finite State Machines

Vincent Hugel, Guillaume Amouroux, Thomas Costis,
Patrick Bonnin, and Pierre Blazevic

Laboratoire de Mécatronique et de Robotique de Versailles (LMRV),
10/12 avenue de l'Europe,
78140 Vélizy, France
{hugel, thcostis, bonnin, blazevic}@lrv.uvsq.fr
http://www.lrv.uvsq.fr

Abstract. This paper presents the specifications and the design of a simple graphical interface for building hierarchical finite state machines. This kind of tool can prove very useful for quickly designing hierarchical behaviors. It can be used in the frame of RoboCup to develop deterministic complex behaviors without focusing on *C*++ coding because source code can be generated from the interface. It is also possible to use it to generate hierarchical finite state machines for whatever purpose needed. The user can create state diagrams by drawing boxes for states and specifying transitions between states. A state diagram can represent a behavior and be considered as a metastate. Diagrams of metastates are possible to constitute several levels.

1 Introduction

In the field of autonomous robotics it is necessary to design the decision making system that is to be implemented into the robots. When the behavior is deterministic, finite state machines (FSM) are often used. But there is no common behaviour design technique that is employed by a majority of teams that participate in robotics competitions. Since *C* code is generally used to compile binaries it is necessary either to write *C* code, or to generate this code from other software or from some kind of high level pseudocode.

The solution of writing *C* code for FSM was used in the early years of RoboCup. One of the big problem was that it was very difficult to upgrade the code quickly without errors. Developers had to be very good at programming and know how to code FSM properly. And even with good programmers, writing or changing FSM source code is very time-consuming and not error prone. In addition it is often a repetitive and tedious task .

A lot of research about the theory of states and events representation has been conducted so far. FSM can be represented formally by Petri nets for example [5]. There exist some graphic interfaces that can simulate the evolution of the FSM but none of them include all the features needed for implementation such as *C* code generation. In most of the cases the solution is built to solve a particuliar problem.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 648–655, 2006.

Some companies propose software products that include graphical interface to help design FSM [12] [13] [14] but these softwares do not offer all the features related to FSM such as assigning priorities to transitions or inserting instructions in transitions. Also it is not possible to modify any of the features that could be useful for the behavior design.

Consequently each team participating in RoboCup uses its own recipe. There is no off-the-shelf solution that may be used as is for robotics applications. Some teams have decided to develop their own high level language to describe the behavior architecture. For example the german team has developed an extensible specification language XABSL [2] [3] based on XML to describe the behaviours, change them and incorporate them among others. It helps a lot to develop behaviours in parallel. The german team used it with success in the RoboCup competition in the legged league. However the developer must have solid knowledge in computer sience. He must know the concept of pseudo code, learn the XML and XABSL languages. He must also master the whole chain from writing XABSL code until generating the binaries. It is possible to display the charts using XML editors for checking, but the design is made by writing pseudo-code and not graphically. The use of XABSL is a real progress but it would be more helpful if it would be possible to make any behaviour design using a graphical interface.

This paper presents the specifications and the design of a very simple graphical interface for the design of hierarchical finite state machines (HFSM). This tool enables the user to create states, metastates that include state or metastate subdiagrams and transitions between them. The number of state levels is not limited. The validity of the FSM can be checked and C^{++} source code can be generated automatically. The code can then be compiled directly using the cross-compiler. Anybody not familiar with C programming could use it and would not waste time focusing on finite state machine coding. This interface was used to design state and event driven diagrams to represent the robots' behaviours in the RoboCup legged league soccer games and challenges in 2004.

2 Specifications

The idea to use a graphical tool to develop behaviors is not new but was never concretized really. This was due to portability and maintenance concerns, and to the big amount of time it was thought it would be needed to develop such an interface. In addition neither commercial solution nor research beta-test softwares did meet all the requirements.

The experience of the French team in RoboCup conducted its members to really tackle this problem in 2004. Before that date everything related to behavior was coded in C directly, and it was clear that it was not possible to keep on that way any more because of the lot of time lost in coding, testing and correcting. Since the coding of FSM follows some strict rules it is possible to automatize the task. Consequently a set of rules have been defined to the generation of C code. From these rules, specifications have been elaborated that lead to the design of the graphical tool.

The specifications required for the interface are very simple so that any trainee not very familiar with computer science can design behaviours. Specifications are detailed in the following section where the concepts of states, metastates and transitions are given. The functionalities the interface offers are also presented. The next section focuses on the design.

2.1 States

- A state represents a situation where the robot can be, or an action it is being executing, for example *waiting-for-start* or *going-to-the-ball.* It can be a temporary state or a state that lasts a certain amount of time. It helps describe the different behaviors a system can adopt. There is no real time consideration here, in contrast to research study on synchronous charts like in [11]. The HFSM update function is called periodically. The calling time period is adjusted according to the processing power at disposal and the time consumption of other tasks running on the robot.
- A state must be represented by a resizeable box and given a name.
- Actions can take place inside states. The related instructions must be included in a specific function whose name should be codified like *what_to_do_in_state_s1()*, where *s1* is the name of the state. The *Code* button permits to get access to the code of this *action* function and modify it.

2.2 Metastates

- Several levels of states are possible. A metastate represents a higher level of states, and includes a sub-state diagram (see paragraph on *diagram* below). A metastate can be seen as a behaviour of a certain level. Each metastate should incorporate an entry state that plays the role of the starting state. For example, a metastate could be given the name of a behaviour ”role” like attacker1, assistant1, defender1, etc. If possible the generalization of state as metastate should be used so that several levels of metastates are possible.
- A metastate must be represented by a resizeable box and given a name.
- States and metastates can cohabitate in the same metastate.
- There is no action associated with a metastate because it is a high-level behavior in itself. The actions will be determined inside the sub-diagram it represents.
- It should be possible to import subdiagrams into an empty meta-state.

2.3 Transitions

- A transition joins a departing (meta)state to a destination (meta)state. So transitions can exist between metastates. A transition can also join a state to a metastate or vice-versa. This means that transitions can occur between states and metastates in the same state-metastate diagram. It can be seen as an event or a change in the environment conditions that triggers a switch of the current (meta)state to another (meta)state. In the following of the paper, the term *state* will refer to state or metastate indifferently except where explicitly mentioned.

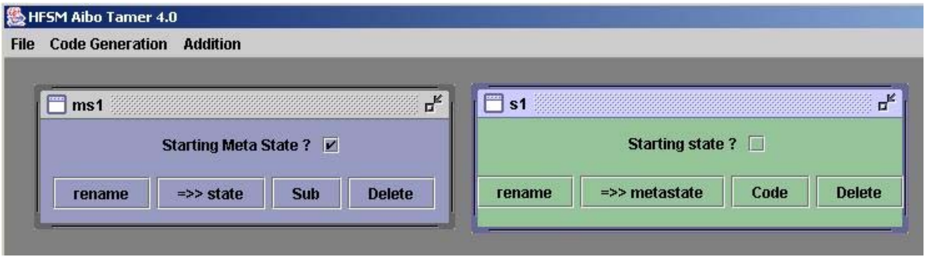


Fig. 1. One metastate called *ms1*, and one state called *s1*

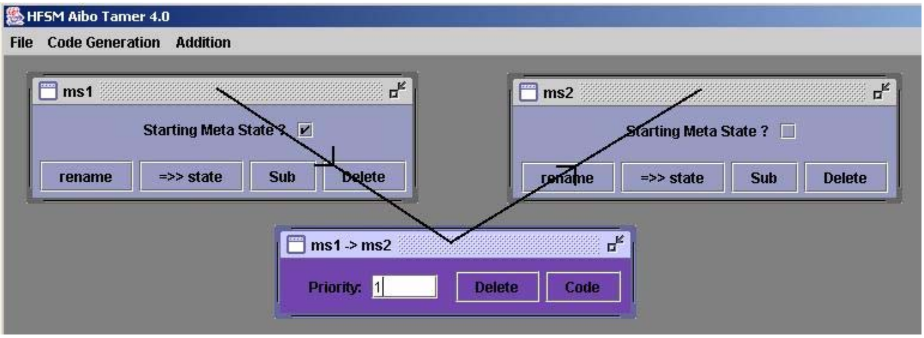


Fig. 2. Transition between 2 metastates

- Transitions between states must be represented graphically, with a transition box and two arrows, one joining this transition box to the arrival state, and the other linking the departure state to this box.
- Transitions must be named according to the departure and arrival states.
- There can be several transitions departing from a state. Therefore these transitions must be arranged by order of priority. This is also a specification that is rarely observed in the available commercial products. As a matter of fact we do not want the transitions from the same state to be strictly exclusive. It may happen that two transitions of the same origin could be validated. In this case the priority must be used to decide whether to activate the one or the other transition. For example, transitions must be checked starting with the one of highest priority. The transition of highest priority that is validated can then be executed.
- Actions can take place inside transitions between states. Strangely, this is a specification that is also rarely found in the available commercial products. But incorporating instructions inside transitions - to be executed before the instructions of the next state, can be very useful to trigger some behaviors. For example from the state *searching-for-ball* to the state *tracking-ball* the transition instructions can trigger a scanning head procedure. The head-position updating function of the *head class* instance that manages all head

joint motions is called regularly in the joint update module specific to the robot's hardware. In the *head scanning* state there is no function call of updating joint. This is a way of simplifying the code implementation.

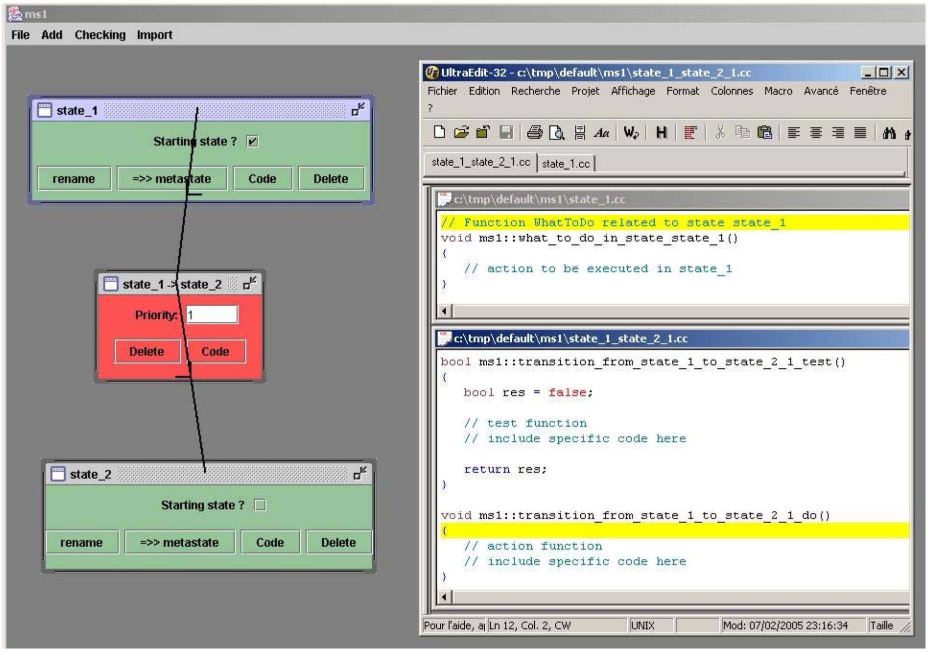
- Transitions must involve two kinds of functions (see fig. 3):
  - one function called *test function* that returns a boolean to indicate whether the transition can be activated or not.
  - one function called *action function* that executes specific actions in case the transition is activated.
- All transitions from a current state must be checked. Instead of investigating the transitions in their order of priority until one returns *true*, it is preferable to run all the checkings because they could be used for debug.
- After all transitions have been tested and there exists at least one that returns a *true* value, only the *action function* related to the transition activated of highest priority must be executed. If there is no transition that can be activated, the *action function* related to the current state is executed. In case of a metastate the actions of the initial state of the sub-diagram are executed.

2.4 State Or Metastate Diagrams

- States, metastates and transitions between them on the same level constitute a diagram. States and metatstates can cohabitate inside the same metastate diagram. A diagram can be composed of one state only. If the state is a simple state, it must be defined as the starting state. If the state is a metastate it must contain a simple starting state at least.

2.5 Functionalities

- **Transformation state-metastate.** It must be possible to transform an existing state into a metastate. This is very useful to decompose a state into a substate diagram when the initial diagram of states becomes too complicated. The reverse operation is also possible.
- **Importation of subdiagram.** Since a metastate can represent a behaviour, the interface should offer the option to import a complete behaviour as a metastate. This metastate must be compatible and can itself be a hierarchical finite state machine.
- **Source code generation.** The option of source code generation must be available. Regarding source code generation, there should be one file for every level of finite state machine definition. This file describes the information about the FSM such as number and list of states, transitions, metastates, connections between states through transitions, etc. There is also one file for every state of the FSM that contains the action function, and one file for every transition that includes the code related to both *test* and *action* functions. This is a requirement to simplify compilation. Source code that is specific to the robot can be included inside *test and action* transition functions, and also inside state *action* functions. Specific code means for



```
// Function WhatToDo related to state state_1
void ms1::what_to_do_in_state_state_1()
{
   // action to be executed in state_1
}
```

```
bool ms1::transition_from_state_1_to_state_2_1_test()
{
   bool res = false;

   // test function
   // include specific code here

   return res;
}

void ms1::transition_from_state_1_to_state_2_1_do()
{
   // action function
   // include specific code here
}
```

Fig. 3. Function prototypes appear in the editor after pushing the *Code* button of a simple state or a transition box. Function code is stored inside C files. For example here the *What_to_do* code for the action function of state 1 is stored inside file called *state_1.cc*. The test and action function source codes of the transition from state_1 to state_2 are stored in the file *state_1_to_state_2_1.cc*. The last number indicates the priority of the transition, which is 1 by default.

example accessing the data resulting from image acquisition and treatment. To include code inside these functions it must be possible to edit the file by clicking somewhere on the state or transition box.

- **Debugging.** A log procedure must be incorporated to help the debug. This means that a log file can be updated at execution if the *log* option was selected for the source code generation. Also some coherence checking must be available to check whether the graphical design of the finite state machine is correct. For instance, there should be no state without transition, all states should be given a name, there should not be several transitions from the same state with the same order of priority, etc.

3 Design and Example of Application

For portability reasons, Java has been used to design the interface. Therefore FSM source code for the robot can be generated using Windows, Linux or other operating systems where the cross-compiler is available. The interface designed looks like a colored state-event diagram, where states appear as rectangular boxes and transitions as arrows associated with boxes.

A typical example where the interface proves very useful is the design of the first level of metastate that deals with the different gamecontroller states when it comes to define game player programs. This first level is composed of 6 metastates: gc_initial, gc_set, gc_ready, gc_playing, gc_penalized, and gc_finished. Each of them contains a sub-diagram that can be accessed graphically by clicking on the *sub* button. In this example if the robot is in the metastate of gc_playing, it means that it is playing, it can have falled down and be standing up, or it can be playing as an attacker or as a goalkeeper. The state of *standing_up_because_of_fall* can be implemented at a higher level here. If the robot falls down the state *standing_up_because_of_fall* is the current state until the movement of recovering is finished. And if the robot detects it has falled down whatever the current state (*players* or *goal*) it enters this state.

Under the level of the *players* metastate there is a metastate called *attacker* that itself contains a sub-diagram of states. But it is possible to imagine other metastates at the same level as the *attacker* metastate.

4 Conclusion and Future Developments

This interface can be used to design multi-layered behaviours based on finite state machines. It is also possible to import some other FSM describing other behaviours. A special option is available for generating AIBO code that can be directly cross-compiled with the other files of the project.

However it is also possible to generate code related to the designed HFSM only. This can be very useful for educational purposes. A first version can be downloaded from http://www.lrv.uvsq.fr/research/legged/software/aibo_tamer.tgz.

In the current version of implementation the HFSM updating function performs only one pass at every time period onboard. This means that the current state can only switch to one of the next states it is connected to. It can be useful to run several passes until reaching a stable state. This is also a way to check that the FSM is correct and that it does not contain any deadlock or infinite loop.

References

1. Hugel et al., RoboCup 2004 French team *Les Trois Mousquetaires* technical report.
2. M. Lötzsch, J. Bach, H.D. Burkhard, M. Jüngel, Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL, 7th Int. Workshop on RoboCup 2003 Padova, Italy.
3. Röfer et al., RoboCup 2004 German Team technical report.
4. A. Arnold, Systèmes de transitions finis et sémantique des processus communicants (in French), Masson Editor, 1992.
5. Haiping Xu, A model-based approach for development of multi-agent software systems, PhD thesis, Chicago University, Illinois, 2003.

6. G. Amouroux, Mise en place de stratégies coopératives entre plusieurs robots autonomes (in French), Master Degree thesis, Versailles University, 2004.
7. Holger Knublauch, Thomas Rose, Tool-supported Process Analysis and Design for the Development of Multi-Agent Systems, Research Institute for Applied Knowledge Processing, Germany. 2002.
8. D. Bonura, Leonardo Mariani et Emanuela Merelli, Designing Modular Agent Systems, Universita degli Studi di Milano Bicocca via Bicocca degli Arcimboldi, Italy, 2003.
9. Pierangelo DellAcqua, Ulf Nilsson et Luis Moniz Pereira, A Logic-Based Asynchronous Multi-Agent Systems, Université de Ferrata, Italie, 2001.
10. S.Fournier, T. Devogele et C. Claramunt, A role-based multi-agent model for concurrent navigation systems - Proceedings of the 6th AGILE Conference on Geographic Information Science, Gould, M. et al. (eds.), presse polythechniques et universitaires romandes, 2003.
11. Charles André,Semantics of SyncCharts, I3S Laboratory, University of Nice-Sophia Antipolis.
12. Matlab toolbox stateflow Coder, http://www.mathworks.com.
13. StateWorks, SW Sotware, http://www.stateworks.com.
14. Visual Case, UML tutorial, http://www.visualcase.com/tutorials
15. XML language extensible markup language,
http://java.sun.com/webservices/docs/1.0/tutorial.
16. Other existing softwares,
http://www.csd.uwo.ca/research/grail/links.html.

Task Allocation for the Police Force Agents in RoboCupRescue Simulation

Mohammad Nejad Sedaghat, Leila Pakravan Nejad,
Sina Iravanian, and Ehsan Rafiee
{mnsedaghat, lpakravan, sina_iravanian, e_rafiee}@yahoo.com

Abstract. In this paper, three mechanisms for task allocation among police force agents in the rescue simulation environment are presented. Three different approaches namely full auction-based, partitioning-based and hybrid approaches are briefly described. The empirical results of using the hybrid approach show a significant improvement in performance over the other two approaches. By using the hybrid mechanism for the police forces, our agents together with other types agents ranked third in the RoboCupRescue 2004 simulation competitions.

1 Introduction

The rescue simulation environment is a multi-agent environment in which fire brigades, ambulance teams and police forces work together to mitigate a simulated disaster [1][2][3]. Extinguishing a spreading fire, saving civilians lives, and clearing blocked roads are the responsibilities of these agents respectively. In this paper we investigate the use of task allocation algorithms in this domain. In a disaster mitigation problem, agents are not aware of the tasks in advance and the priority of tasks change over time. Agents of each type are responsible for their own duties. The problem is then how to assign agents to these tasks dynamically. We have introduced three approaches namely full auction-based mechanism [4][5][6][7], partitioning-based mechanism, and a hybrid mechanism which merges the benefit of the other two approaches. The auction-based mechanism has been used by Caspian 2004 Rescue Simulation team during the German Open 2004 competitions in which we ranked fourth. The hybrid mechanism also was exploited during RoboCupRescue 2004 simulation competition where we won the third place.

In the next section the characteristics of the rescue simulation environment is briefly described. Section three describes the usage of task allocation mechanisms in this domain. In the first part of section three, the fully auction-based mechanism is introduced followed by the partitioning-based approach. After that, in the third part, our hybrid approach is presented. The fourth section is devoted to empirical results of using these approaches, and makes a comparison between them.

2 The Rescue Simulation Environment

The rescue simulation environment is a multi-agent environment in which three types of agents are defined to cooperate with each other with the goal of mitigating a

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 656 – 664, 2006.

simulated disaster. Fire Brigades are responsible for extinguishing a spreading fire. Ambulance Teams are responsible for saving the lives of the civilians who need help and Police Forces are in charge of clearing blocked roads. Fire Station, Ambulance Center, and Police Office help corresponding agents accomplish their tasks. In such an environment the problem of disaster mitigation can be considered as a task allocation problem, where each type of agent is responsible for its own defined duties. The agents are not aware of the tasks in advance. Indeed, as the simulation continues new tasks emerge, and the fact that the priority of tasks change over time, makes the situation even worse. Sometimes there's a need to explore the world for accomplishing a task (e.g. finding civilians who need help), and sometimes the tasks are reported by other agents who faced them (e.g. clearing a road through which another agent needs to pass). The problem is then how to assign agents to these tasks dynamically.

3 Task Allocation

The focus of this research is on the development of a dynamic task-allocation algorithm for the Police Force Agents. Police Forces are the agents that help other agents accomplish their tasks better. These agents will be informed about new tasks either by themselves exploring the world or by other agents who faced the obstacles. To address the problem of dynamic task allocation among police forces we have developed three approaches which will be explained in later sections in more detail.

3.1 Full Auction-Based Mechanism

Auction mechanism is a market-based paradigm which consists of an auctioneer (seller) and potential bidders (buyers) in which items are sold to a buyer who suggested the highest (or lowest) price [6][7]. Once the auctioneer wants to sell an item, he will notify the bidders of the item. The auctioneer receives all the bids and determines the best suggestion, and notifies the bidders about the winner. One of the main advantages of auction-based mechanisms is that they are distributed in the sense that each agent performs a local computation regarding its own bidding strategy.

In our approach, the police office takes on the role of an auctioneer, and the police forces take on the roles of the bidders. The items that are bid for are the tasks or, in this environment, the blocked roads. In an auction algorithm, both single and combinatorial mechanisms can be used. In our approach we used a single method. The following scenario is applied:

1. Fire brigades and ambulances send a request for clearing the obstacles in a blocked road to the fire station or the ambulance center respectively.
2. Fire station and ambulance center collect the received requests and send a request to the police office.
3. The police office notifies police forces about the received requests. The real auction starts here.
4. Police forces receive the requests and make bids on them, and send their bids to the police office.

5. The police office collects all the bids and determines the winner.
6. All the police forces will be notified about the winner by the police office.
7. After accomplishing the task, the police force who won the auction will notify the center that he is free, so that he can participate in coming auctions.

In addition to the above scenario, at the beginning of each cycle the police forces bid for the tasks which are remained unassigned yet. To complete our discussion, more detail about the bidding strategy and winner determination is given below.

- **Bidding Strategy:** Once a free police force is informed about a new task he will make a bid for it. To this end, the agent uses a cost function to calculate the cost of performing that special task. This cost depends on some parameters, like the distance to the destination route, distance of the destination from the refuges, the distance of the destination from fiery buildings, and some other parameters. In an auction-based mechanism an agent can use any kind of bidding strategy for calculating the cost of accomplishing a special task. It does not matter what the strategy exactly is while it is proportional among other agents. For these reasons we do not discuss the bidding strategy in more detail here.

  As another point, we assume that each task is performed by a single agent. Considering that removing a blockade is not too costly, this assumption does not affect the whole problem.
- **Winner Determination:** The police office receives all the bids and determines the least costly proposal as the winner. Since we assume that each free agent makes bids on all tasks, the winner determination procedure is straightforward. The auctioneer assigns the first task to the agent who made the least costly bid for it. Then the agent and the other bids it had made are removed from the list of suggested bids. The auctioneer continues his work by finding the next least costly bid and so on. This means that in this approach the least costly bid is preferred over the others.

Empirical results show that using full auction-based mechanism has the following advantages:

- **Different Bidding Strategies:** Bidders are allowed to implement different bidding strategy, which leads to a more flexible structure. We can assume different responsibilities for police forces, for example consider a police force which is more inclined to help fire brigades or ambulances. Therefore it is possible to assign different responsibilities in this level. However note that the final task-allocation is made by the police office and all the received bids are assumed to be made by identical agents, and the least costly bidder will win the auction.

- **Efficient Task Allocation:** Auction-based task allocation ensures that each request (task) will be assigned to a bidder (police force). This leads to the fact that no tasks remain unassigned.

Besides the mentioned advantages, there are several shortcomings:

- **Large Number of Communicated Messages:** In the auction-based mechanism, it is necessary to transmit a number of messages to set up an auction. In our approach, the first message is an announcement of a new task. The second type of message is the bids that agents send to the police office. The third message is the winner notification. All of the above messages should be transmitted in order to assign one task to an agent. Considering that in a typical disaster situation the number of police forces involved in the environment varies from 10 to 15, and all of the free agents bid on a task, the number of communicated messages is too large. Event if we encapsulate information in one message, the number of communicated messages would be still significant.
- **Delay in Task Execution:** As mentioned above three types of messages should be transmitted in order to assign a task to an agent. It will take at least 3 cycles for a task that is requested to the police office to be assigned to a police force (actually this is because of the limitations in the simulated environment). However, by using a proper algorithm, the police force can remove the obstacle in fewer cycles.

To address the above shortcomings, we have mixed the auction algorithm, with a partitioning-based task allocation mechanism to build a hybrid approach. The basis of the partitioning-based approach is given in the following section, and the hybrid approach is introduced later.

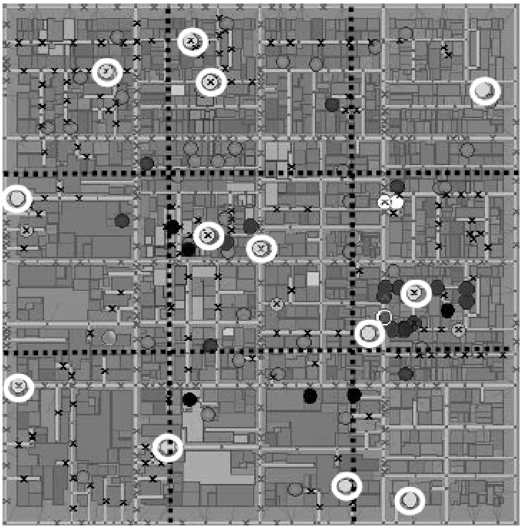


Fig. 1. The partitions made and the assigned police forces to each partition. White circles represent police forces.

3.2 Partitioning-Based Task Allocation

Partitioning-based approach is another way of simply allocating tasks between the police force agents. In this way the map of the city is partitioned into several regions, resembling a grid. Note that the number of partitions is less than the number of police force agents. The police office is responsible for partitioning the disaster space and assigning the police forces to these partitions. Also there might be a need to change the activity region of an agent from one partition to another.

More details about these steps are given below:

- **Partitioning the Disaster Space:** Efficient partitioning the disaster space is one of the open challenges in the rescue simulation domain. In our approach the disaster space is statically partitioned into a number of regions. This was carried out in the beginning of the simulation and did not change till the end. The number of partitions must be less than the number of police force agents. A better solution to this problem is to dynamically partition the disaster space with respect to the density of the blockades in different parts of the city, and the degree to which other types of agents are busy in a special region. However, this is yet an open issue in this approach needing further research.
- **Assigning Police Forces to the Partitions:** Assigning police forces to different partitions is carried out both statically and dynamically. Meaning that in the beginning of the simulation one police force is assigned to each partition. Since the number of partitions is less than the number of police forces, there are always some free police forces available that can be assigned to more busy regions with more requests dynamically.
- **Changing the Activity Region of a Police Force:** The disaster space is a dynamic space, in which no preliminary information about the potential tasks of an agent is available. This makes it impossible to define a planning for clearing the roads in advance. The change in activity area of the fire brigades and the ambulances makes the need for clearing the roads in different parts of the disaster space change over time. It may be necessary to assign more police forces to a special region than the usual number. To this end the police office changes the activity region of one or more police forces according to the rate of requests in different regions. Note that after all the roads of a partition are cleared, the police forces assigned to that region are considered as free agents.

In contrast with the fully auction-based mechanism, the advantages of this approach are:

- **Faster Task Assignment:** In the fully auction-based approach it took 3 simulation steps for the police office to assign a task to a police force. But in this approach each task that arises in a region is immediately added to the agent's list of tasks. In some situations (not all) this leads to a better performance for the police team.
- **No Permanent Need to the Police Office:** One of the fundamental capabilities of a rescue team should be its ability to operate well in absence (failure) of the decision-making centers. In this approach since the disaster space is partitioned into several regions, in the case of police office failure, the police forces will explore the whole area and find and remove the obstacles on their way. Although it is inefficient, the police office failure does not cause the

police forces to cease operating, which was not possible in the full-auction model.

- **Less Communication Messages:** The communication messages in this approach are reduced to the messages through which the police office notifies the police forces about the tasks. This is much less than what was seen in the auction-based approach.

Besides the aforementioned advantages this approach has several disadvantages. The most important shortcoming of this approach is that the agents perform the tasks inefficiently – just contrary to the auction approach. In this approach all the tasks in a partition are considered altogether, and the agent(s) assigned to that region are responsible for carrying them out. If the agents are busy, there's a possibility that a task be carried out too late.

3.3 Hybrid-Method for Task Allocation

In the hybrid approach we have tried to combine the benefits of the two methods, and avoid common shortcomings as far as possible. For this method, two types of requests have been defined; one the urgent requests that must be dealt with as soon as possible, and the other the normal requests, for which there's no emergency in handling them. The urgent requests are made by the other types agents when thy need an obstacle to be removed as soon as possible. Examples are, when the only path to the fire refuge is blocked, or an agent is trapped in a road that is blocked on both heads. In the beginning of the simulation the police office partitions the disaster space and assigns one or more police forces to each partition. The agents operate and react to the normal requests in the same way as the second approach. When there's an urgent request made, the police office holds an auction for it. All the police forces in all parts of the disaster space participate in the auction, and bid for the urgent task. It does not matter which region the task belongs to, any agent belonging to any region may win the auction. After accomplishing the task, the winning agent returns to the partition to which he was originally assigned to.

The advantage of this hybrid approach is that the police forces operate in the whole disaster space, while some of them are dealing with urgent tasks. In this way fire brigades and ambulances can change their activity area with less performance loss.

4 Empirical Results

In order to measure the performance of the three methods discussed above, we conducted 5 sets of experiments. The different setups used to perform the experiments are as follows:

1. Full auction mechanism is used for the police force agents.
2. The partitioning-based method is used for the police force agents.
3. The hybrid method is used for the police force agents.
4. While using the hybrid approach the police office is not running.
5. The blockade simulator is not running.

Note that other types of agents are identical in our experiments. For all the experiments the score of the whole rescue team is used to measure the performance. Although the score is achieved by all types of agents involved in the simulation (not the police forces alone), the difference in the score helps evaluate the different methods used for the police forces. The 5 sets of experiments were performed using 4 different maps, which were used in the RoboCup 2004 final round, namely Kobe, VC, Random Map, and Foligno. Each experiment was performed 10 times, and the average of the scores gained is presented.

In the first set of experiments the full auction mechanism was used. This method was used by the police agents in the Caspian rescue simulation team during German-Open 2004 competitions. The second set shows the scores gained while using the partitioning-based approach. In the third set of experiments the hybrid-method which makes use of the advantages of the other two methods is tested. This method was used by the Caspian rescue simulation team during the RoboCup 2004 competitions. In the set of experiments in which the police office is not running, there's no auction, and no clearing requests will be sent to the police forces either. The police forces discover the tasks by themselves. The result of the experiment in which the blockade simulator is not running shows the overall score of the rescue team in absence of the blockades. This helps to take into consideration the maximum possible score of the rescue team.

Comparing the auction mechanism to the hybrid-method shows that in the beginning of the simulation the police forces using the auction mechanism scored better than those using the hybrid-method. But the situation changes over time, and the hybrid-method ends with a higher score. That is because in the auction- mechanism the

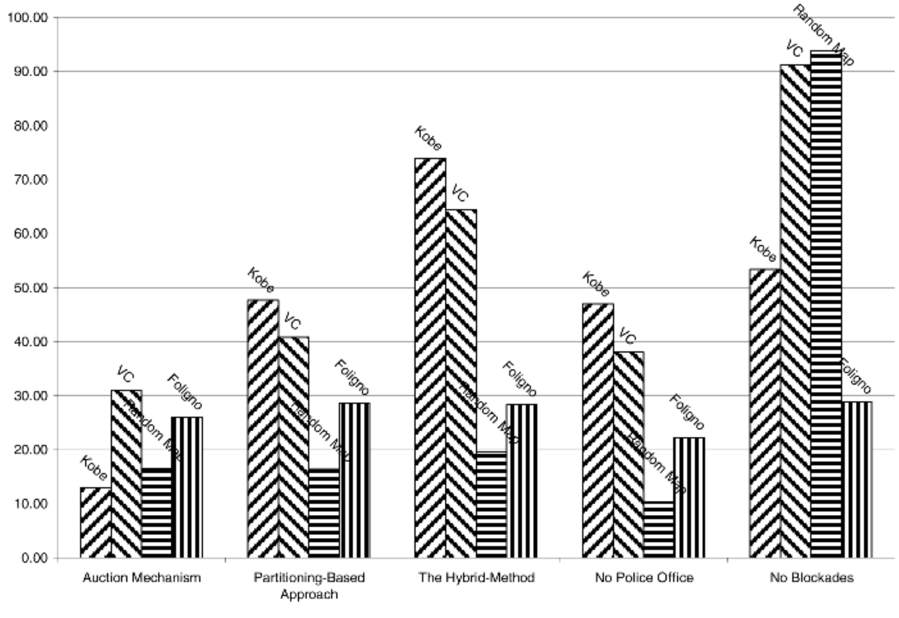


Fig. 2. The average scores of the experiments using the 3 methods in 4 different maps

police forces deal with the blockades in the region of the fire brigades and ambulances focus. As the simulation continues, these agents need to change their activity area, but other parts of the city are remained full of blockades. In the hybrid-method one or more police forces are assigned to each partition, and other police forces deal with the urgent requests. That causes the city be cleared from blockades, while the urgent requests by the fire brigades and ambulances are handled.

The scores gained through the experiments are illustrated in figure 2. In this figure it is shown that the hybrid-method had a higher performance over other methods in all the maps used. Police forces using the partitioning-based mechanism scored higher than those using the auction mechanism. One point to be clarified in fig. 2, is that the team scored less in the Kobe map when no blockades were running, with respect to the situation where the hybrid-method was used. The reason behind such facts is that the score of the team is highly dependant to the map configuration, and special situations happening in each run, and facts such as which fire do the fire brigades choose to extinguish first.

5 Conclusion and Future Work

In this paper, the usage of task allocation mechanisms for police force agents in the disaster mitigation environments has been described. Three different approaches have been presented. The empirical results of using these approaches show that the fully auction-based mechanism looks promising in situations where there is an urgent need to reply. However, in rescue simulation domain where the agents change their working area rapidly, using this approach will lead to a significant performance loss, because other parts of the disaster space were not considered. By using the second method, partitioning-based approach, we have noticed that all parts of the disaster space are cleared simultaneously, but the urgent requests are handled too late. The third method combines the advantages of the other two. In this method the urgent requests are handled as fast as possible, while other agents are distributed in other partitions and serve other parts of the city monotonically.

By using the hybrid method in the Caspian 2004 Rescue simulation team, our agents together with other type agents succeeded to rank third.

In our research work we are going to enhance the performance of the system by adding the ability for the fire fighters and ambulance teams to report the priority of their requests. Partitioning the disaster space dynamically based on the observed priority of each region, is another work to do.

References

1. H. Kitano et. al, RoboCup-Rescue: Search and Rescue for Large Scale Disasters as a Domain for Multi-Agent Research, In Proceeding of IEEE Conference, SMC-99, 1999.
2. The RoboCupRescue Technical Committee, RoboCup-Rescue Simulator Manual version 0 revision 4, 2000
3. M. Takeshi, How to Develop a RoboCupRescue Agent, 2000

4. Nair, R., Ito, T., Tambe, M., Marsella, S., Task Allocation in the RoboCupRescue Simulation Domain: A short note.
5. Nair, R., Ito, T., Tambe, M., Marsella, S., RoboCup Rescue: A Proposal and Preliminary Experiences
6. Martinez, V., Sklar, E., Parsons, S., Exploring Auction Mechanisms for Role Assignment in Teams of Autonomous Robots. In Proceedings of the RoboCup Symposium, 2004.
7. Baron, S., Resource and Task Allocation in Distributed Environments. A Multi-Agent System Approach.

The Advantages of the Signaling Strategy in a Dynamic Environment: Cognitive Modeling Using RoboCup

Sanjay Chandrasekharan[1], Babak Esfandiari[2], and Tarek Hassan[2]

[1] Institute of Cognitive Science
[2] Department of Systems and Computer Engineering,
Carleton University, Ottawa, Canada, K1S 5B6
{schandra, babak, thassan}@sce.carleton.ca

Abstract. We report a cognitive modeling experiment where the RoboCup simulation environment was used to study the advantages provided by signals. We used the passing problem in RoboCup as our test problem and soccer-players' 'yells' of their 'passability' values as the task-specific signals. We found that yells improve pass completion – using yells to decide the best player (to pass the ball) led to a 8-17 percentage points increase in performance compared to a centralized calculation of best pass. However, the passability values themselves did not make a difference, indicating that the advantage of signals come from their different perspective in identifying a pass, the actual content of signals do not matter. We present some problems we faced in using Robocup as a modeling environment, and suggest features that would help promote the use of RoboCup in cognitive modeling.

1 Introduction

Many organisms generate stable structures in the world to reduce cognitive complexity, for themselves, for others, or both. Wood mice distribute small objects, such as leaves or twigs, as points of reference while foraging. They do this even under laboratory conditions, using plastic discs. Such "way-marking" diminish the likelihood of losing interesting locations during foraging [1]. Red foxes use urine to mark food caches they have emptied. This helps them avoid unnecessary search [2]. The male bower bird builds colorful bowers (nest-like structures), which females use to make mating decisions [3]. Such epistemic structures (ES), usually termed signals, are used widely, and form a very important aspect of animal life across biological niches. These structures allow organisms to hive off a part of their cognitive load to the world [5]. How much cognitive advantage do such structures provide in noisy, dynamic and adversarial environments? How robust is this advantage? What are its components? These are the problems addressed in this paper. We used RoboCup to study the cognitive advantages provided by the signaling strategy.

2 Agent Design Taxonomy

The section below develops a framework to understand how signaling (or the ES strategy, where the environment is changed in a way that it contributes task specific

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 665 – 672, 2006.

structures for decision-making), fits in with other agent-environment relationships. We categorize agent-world relations into four strategies, and use the design problem of providing disabled people access to buildings to illustrate these strategies.

Strategy 1: This involves building an all-powerful, James Bond-style vehicle that can function in all environments. It can run, jump, fly, climb spiral stairs, raise itself to high shelves, detect curbs etc. This design does not incorporate detailed environment structure into the vehicle, it is built to overcome the limitations of all environments.

Strategy 2: This involves studying the vehicle's environment carefully and using that information to build the vehicle. For instance, the vehicle will take into account the existence of short curbs, stairs being non-spiral and having rails, level of elevator buttons etc. So it will have the capacity to raise itself to short curbs, climb short flight of straight stairs by making use of the rails etc. Note that the environment is not changed here.

Strategy 3: This involves adding structure to the environment. For instance, building ramps and special doors so that a simple vehicle can have maximum access. This is the most elegant solution, and the most widely used one. Here structure is added to the environment, the world is "doped", so that it contributes to the agent's task.

Strategy 4: This strategy is similar to the first, but here the environment is all-powerful instead of the vehicle. The environment becomes "smart", and the building detects all physically handicapped people, and glides a ramp down to them, or lifts them up etc. This solution is an extreme case of strategy III, we will ignore it in the following analysis.

The first strategy is similar to the centralized AI one, which ignores the structure provided by specific environments. The environment is something to overcome, it is not considered a resource. This strategy tries to load every possible environment on to the agent, as centrally stored representations. The agent then tries to map the encountered world on to this internal template structure. The second strategy is similar to the situated AI model promoted by Rodney Brooks [6]. This strategy recognizes the role of the environment as a resource, and analyses and exploits the detailed structure that exists in the environment to help the agent. Notice the environment remains unchanged, it is considered a given. The third strategy is similar to one aspect of distributed cognition, where task-specific structures are generated in the environment, allowing the agent to hive off part of the computation to the world. Kirsh [7] terms this kind of "using the world to compute" *active redesign*. This principle is at work in the "intelligent use of space" where people organize objects around them in a way that helps them execute their functions [8].

3 Using RoboCup to Study Epistemic Structure

RoboCup provides an interesting dynamic and adversarial environment to study the efficiency of the ES strategy. However there is not much scope to add task-specific structure to the environment. The only structure that can be added is 'yells', or signals from teammates. We chose to use this structure, and studied the passing problem (i.e. how an agent in control of the ball can decide whom to pass the ball) to test the efficiency of this structure. We developed RoboCup teams that used three different approaches to passing. The teams were based on the UvA TriLearn 2002 code [9].

3.1 Team 1: Centralized Passing

This team (A1) uses approach 1 in our agent design taxonomy. A1 does not depend on task-specific information from other agents. In A1, when an agent has possession of the ball (i.e., the ball is within a kickable margin), it calculates the pass suitability (passability) for each teammate, and passes the ball to the teammate with the highest passability. If no teammate has passability above a fixed threshold value, the agent will dribble the ball toward the opponent goal. The goalie in this team is based on the original UvA algorithm, except for one modification: in a goal kick or free kick, the goalie will use A1 to calculate the best receiver for a pass and kick the ball to that teammate.

3.2 Team 2: Passing with Yells

This team (A2) is an implementation of the Active Design approach. Here every agent calculates its own passability. This calculation is done for every cycle a teammate has control of the ball. The fastest player in a set who can reach the ball is determined to have control of the ball. Once the passability value is calculated, each player uses the 'say' command to communicate this number to teammates. When updating the world model, every agent uses incoming aural messages from teammates to track the best passability at a given time. If a message arrives announcing a higher passability, then the sender of the message becomes the new best pass receiver. Every five cycles, the best passability is reset to the minimum threshold, to ensure that old information is not used to make the passing decision. As in centralized passing, the goalie uses A1 to calculate the pass receiver, but unlike its teammates, the goalie uses the centralized approach with no input from teammates. This ensures that the goalie always passes to someone.

3.3 Team 3: Passing with Filtered Yells

This team (A3) is also an implementation of Active Design, but it has some properties of the Brooksian approach, in the sense that it takes into consideration the limitation of the communication channel, which is a significant property of the environment. In A3, instead of agents saying their passability every cycle, agents listen to others' yells and compare their passability with the ones they hear. They announce their passability only if it is better. This lowers the load on the communication channel, by allowing only the best messages through. Once again, the goalie uses the centralized approach to passing.

4 Experiments

Each modified UvA team was pitted against the original UvA team to test the passing algorithms. Each team played 10 games. Logs of individual agents' decision-making were collected and analyzed to extract the successful and unsuccessful passes, and the passability values[1]. Note that even though A1 uses centralized decision-making to pass, the other agents in A1 calculate their own passabilities and store these values. In effect, all agents in all the three conditions calculate their passabilities when a teammate has the ball. In A2 and A3, this information was ‘yelled’, and the passing agent’s decision to pass was based entirely on this information. In A1, there was no yelling by individual agents, they just stored their passability values, calculated in a centralized manner.

5 Results

5.1 Pass Completion

We analyzed the log files of games played by the three algorithms, and checked who next kicked or caught the ball after a player made a pass. If it was the intended recipient, the pass was completed, otherwise it failed. Table 1 shows the results of running the three modified teams against the original UvA team, and testing over ten games for each team.

Table 1. Number of passes completed

Team	Total Passes	Passes Completed	Percentage
A1	2091	789	37.7%
A2	1534	401	28%
A3	3426	960	26.1%

The number of passes are lower in the case of A2 than A1 because agents in A2 wait to hear yells, and if they don’t hear a yell, they will dribble, instead of passing. The number of passes in A3 is higher in the case of A3 than A1 because A3 hears more yells. The above values show the performance of the three algorithms, given the narrow width of the communication channel (default, 1 message per cycle). Since the communication channel could not be broadened easily, we used two techniques to filter out the bottleneck effect of the communication channel, and capture the performance of the ES strategy better.

Technique 1: This involved potential receivers of the ball calculating their passabilities and logging them, even as the agent in control of the ball was calculating passabilities using the centralized algorithm. This means when an agent (say X) had control over the ball, all agents who could see X (say agents C, F, G, H) calculated their passabilities and stored their values. X calculated its passability in a centralized fashion and logged that value. Our first filtering technique involved using these stored values to filter out only those passes where the agent in control of the ball (say X) decided to pass to the agent with the highest passability (say F) among the possible receivers, according to the estimates of the receivers. For example, let’s say X had the ball and Agents C, F, G and H could see X. In the first team (centralized algorithm, A1), X calculated the passabilities of agents C,F,G and H in a centralized manner. At

[1] We thank Neal Arthorne for implementing this algorithm and a log analysis tool.

the same time, agents C,F,G and H calculated their own passability values with regard to X. In A2 and A3 (the ES algorithms), X waited to hear the messages announcing passability from C,F,G and H.

Consider the first case (algorithm A1). Let's say according to X's centralized calculation, C was the best pass. But according to the calculations of C,F,G and H, agent G was the best pass. Here the passing agent and potential receivers disagree. But in some other instances, both the centralized calculation (passing agent) and the calculation by potential receivers agree (say they both calculate C as the best pass). Considering only passes of this latter kind is gives the same result as finding out situations where all the messages get through. The agreed situations pick out the instances where the passing agent decides to pass to the agent considered best by potential receivers. Note that this is true for all the three algorithms (A1, A2 and A3). This leads to a subset of the total passes being considered for analysis. These idealized passes (termed Agreed Passes) present the situation where the teammates' decision was communicated to the passing agent, and these passes incorporate their different perspective from the centralized agent. The following table presents the results from this analysis.

Table 2. Number of Completed Passes among Agreed Passes

Team	Total Passes	Passes Completed	Percentage
A1	803	369	45.9%
A2	566	210	37.1%
A3	1536	518	33.7%

These results show that the performance is significantly higher for agreed passes. This means receiving information from teammates (incorporating their perspective) leads to an increase in performance. There is an anomaly, however. The performance of A2 and A3 are still much below that of A1, with A2 barely matching A1's performance from the first analysis, and A3 performing 4 notches below that. Since agreement essentially takes away the limitations of communication, and considers only the scenarios where the signal is available to all the three algorithms, A2 and A3 should perform at least at the same level as A1, because they are all now using the same information (the potential receivers' assessment), and the same base level skills. Why is their performance lower?

One reason for the lower performance of A2 and A3 could be that the agents in control of the ball in A2 and A3 receive messages from potential receivers they can't see, like agents behind them, or at an angle to them. Passing to these invisible agents would be difficult, and the probability of such passes being completed is quite low. On the other hand, since A1 calculates only passabilities for agents it can see, agreed passes in A1 automatically leaves out agents it cannot see. This raises the power of its kicks and lowers randomness in the direction of the ball once it is kicked, raising the probability of completing the passes. This interaction between perspective and performance presents a trade-off in using the ES strategy in dynamic environments. On the one hand, the ES strategy can provide information from another perspective, and this is information an agent cannot get by using the centralized strategy. But on the other hand, given the physical limitations imposed by their perspectives, agents receiving this information may not be able to use it always. This means the ES strategy would be most effective in situations where the physical limitations of the agent are used to filter the information provided by the ES. To weed out the perspective-performance interaction, we analysed the data again from another angle.

Technique 2: This technique tries to filter out the perspective problem involved in a pass and focus entirely on the quality of information, i.e. the correlation between completion of passes and agreement. This is done by examining only the set of completed passes, and seeing how many of them were agreed passes. The chart below captures this.

Table 3. Number of agreed passes within completed passes

Team	Total Passes	Passes Completed	Percentage
A1	789	369	46.7%
A2	401	210	52.3%
A3	960	518	53.9%

The analysis shows that agreement predicts completion almost 48% of the time for A1, similar to the last analysis. This is expected, because A1 is not limited by perspective constraints, it calculates only those passes it can execute. On the other hand, the performance for A2 and A3 increases to around 52 and 54%. If the passes can be executed well, the information provided by the ES strategy predicts completion better than the information provided by the centralized strategy (around 17 percentage points increase).

5.2 Robustness Experiments

To examine the robustness of the ES strategy, we manipulated two variables -- noise and time taken to calculate the pass. These variables approximated variations in the environment and the processing capabilities of organisms. The noise was varied by changing the *player_rand* parameter of the soccer server from 1 to 10. The ES strategy outperformed the centralized strategy at all noise levels. The time taken to calculate the pass was varied by adding a sleep parameter to the passability function, and then varying the amount of sleep. The ES strategy once again performed better than the centralized decision-making strategy. These results show that the ES strategy is quite robust.

5.3 Analysis of Passability Values

From the previous analysis, we know that the passability calculation done by the potential receivers provides a better predictor of completion. But this analysis only compares the agent identified by the yelling agents and the passing agents (best passability), and not the *passability values* generated by the passing agent and the potential receivers. Is there a minimum value below which completion rates are low, and above which they are high? To understand this, we looked at the best passability values calculated by the yellers and the centralized agents, and then broke them down into 12 categories (10-20, 20-30 etc.), and looked at the total number of passes, completed passes and agreed&completed passes. For a clearer picture on this, we looked at the total number of passes in each band of passability values (10-20, 20-30 etc.), and then looked at the average completion for each band. There was a slight increase in the completed passes as passability values increased, but the difference was not significant between the yells and centralized calculations. The similar completion rates (for yells and centralized) seem to indicate that the passability value does not differentiate between the two approaches (centralized, yells). One possible reason for this could be that the passability calculation is wrong, and does not make any difference at all. This can be ruled out, because agreement makes a significant difference in pass completion. A more plausible reason could be limitations imposed by the physical states of the agents (like stamina, view etc), which influences the strength of the kick, direction of the kick etc. The physical constraints set an upper limit to the completion of passes. The similar pattern of completion for different values, compared to the better completion rate seen in the earlier analysis for agreed passes, taken together indicate that having a higher passability value does not lead to better passes, but identifying the best agent makes a difference. That is, if both the yells and the centralized calculation identify agent C as the best pass, that improves the chance of the pass being completed. But yells or the centralized calculation deriving a 70-80 passability value for Agent C does not improve the chance of completion. This means *the passability value is useful only for its relative perspective information, helping determine who is better among possible receivers. It is not a good indicator for pass completion.* The ES strategy is better because of its perspective, and not because of its accuracy in calculating the passability value.

6 Limitations and Future Work

One of the major limitations of the study is the indirect way of assessing the effectiveness of the ES strategy. This is a direct result of the narrow communication channel. If the server parameters had allowed us to manipulate the hear_max value beyond 2 messages per cycle, we would've been able to judge the effectiveness of the strategy better. This would've also provided a way to better understand the relationship between channel-width and signal effectiveness in a dynamic environment. The freedom to change parameters, and a more user-friendly way of doing this, could lead to the RoboCup environment being used more widely by disciplines like cognitive science.

In this study, the opponent team was the same one for all the games. Even though this could be considered as providing a standardization for the results reported here, it is desirable to test a cognitive strategy in a variety of situations. Similarly, tests need to be done to determine the optimal number of waiting cycles used by a player in A2 and A3 before deciding on whom to pass. A further limitation is that the opponent team was not designed to intercept the passability messages, or to manipulate them. So the adversarial nature of the environment was limited to pass interception. In future work, we plan to use different teams against our teams. We also plan to investigate how unreliable messages affect the ES strategy. This is the equivalent of mimicry in biological systems.

References

1. Stopka, P. and Macdonald, D.W. Way-marking Behaviour: An Aid to Spatial Navigation in the Wood Mouse (Apodemus Sylvaticus). BMC Ecology,(2003)
2. Henry, J.D. The Use of Urine Marking in the Scavenging Behaviour of the Red Fox (Vulpes Vulpes). Behaviour, 62:82-105 (1977)
3. Zahavi, A. and Zahavi, A. The Handicap Principle: A Missing Piece of Darwin's puzzle. Oxford University Press, Oxford (1997)
4. Bradbury, J.W. & Vehrencamp, S.L. Principles of Animal Communication. Sunderland, Mass: Sinauer Associates. (1998)
5. Chandrasekharan, S. & Stewart, T.: Reactive agents learn to add epistemic structures to the world. In K. D. Forbus, D. Gentner & T. Regier (Eds.), proceedings of the 26th Annual Meeting of the Cognitive Science Society, CogSci2004, Chicago. Hillsdale, NJ: Lawrence Erlbaum, (2004)
6. Brooks, R. A. Intelligence without Representation, Artificial Intelligence, 47:139-160. (1991)
7. Kirsh, D. Adapting the Environment Instead of Oneself. Adaptive Behavior, Vol 4, No. 3/4, 415-452. (1996)
8. Kirsh, D. The Intelligent Use of Space. Artificial Intelligence. 73: 31-68 (1995)
9. Kok, J.R. UvA TriLearn 2002 Source Code, University of Amsterdam (UvA), Faculty of Science Intelligent Autonomous Systems Group, (2002)

Towards Eliminating Manual Color Calibration at RoboCup

Mohan Sridharan[1] and Peter Stone[2]

[1] Electrical and Computer Engineering, The University of Texas at Austin
smohan@ece.utexas.edu
[2] Department of Computer Sciences, The University of Texas at Austin
pstone@cs.utexas.edu
http://www.cs.utexas.edu/~pstone

Abstract. Color calibration is a time-consuming, and therefore costly requirement for most robot teams at RoboCup. This paper presents an approach for autonomous color learning on-board a mobile robot with limited computational and memory resources. It works without any labeled training data and trains autonomously from a color-coded map of its environment. The process is fully implemented, completely autonomous, and provides high degree of segmentation accuracy. Most importantly, it dramatically reduces the time needed to train a color map in a new environment.

1 Introduction

Upon arrival at RoboCup competitions, one of the first steps for most teams in any of the real robot leagues is *color calibration*: the process of mapping raw camera pixel values to color labels, such as green or orange. Due to differences in lighting conditions and object colors between the teams' labs and the competition venue, pre-trained vision modules are unlikely to work "out of the box."

The time required for color calibration contributes in large part to the need for multiple days of setup time before each competition, a costly proposition both from the perspective of reserving the venue and from the perspective of individual travel expenses. In addition, both soccer-playing and rescue robots must eventually be able to operate in natural, changing lighting conditions. Rescue robots in particular must be operational as soon as possible after arriving at a disaster site.

Though events, to date, have all been held under constant, bright lighting conditions, it takes an hour or more to train the robot to recognize the desired colors in its environment. One way to dramatically reduce this time is to enable the robot to autonomously learn the desired colors from the environment using the inherent structure. Doing so may also enable them cope more easily with changing lighting conditions.

In the abstract, automatic color segmentation can be characterized by the following set of inputs, outputs and constraints:

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 673–681, 2006.

1. *Inputs:*
   - ∗ A color-coded map of the robot's *world*. This contains a representation of the size, shape, position, and colors of all objects of interest.
   - ∗ A stream of limited-field-of-view images that present a view of the *structured* world with many useful objects, and many unpredictable elements.
   - ∗ The initial position of the robot and its joint angles over time, particularly those specifying the camera motion.
2. *Output:*
   - ∗ A *Color Map* that assigns a *color label* to each point in the color space.
3. *Constraints:*
   - ∗ Limited computational and memory resources with all processing being performed on-board the robot.
   - ∗ Rapid motion of the limited-field-of-view camera with the associated noise and image distortions.

This paper presents an approach for autonomous color learning on-board a mobile robot with limited computational and memory resources. It works without any labeled training data and trains autonomously from a color-coded map of its environment. The process is fully implemented, completely autonomous, and provides high degree of segmentation accuracy. Most importantly, it dramatically reduces the time needed to train a color map in a new environment.

2 Background Information

The SONY Aibo, *ERS-7*, is a four legged robot whose primary sensor is a CMOS camera with a field-of-view of 56.9^o (hor) and 45.2^o (ver), providing the robot with a limited view of its environment. The images are captured in the *YCbCr* format at $30Hz$ and an image resolution of 208×160 pixels. The robot has 20 degrees-of-freedom (dof). It also has noisy touch sensors, IR sensors, and a wireless LAN card for inter-robot communication. The camera jerks around a lot due to the legged locomotion modality, and images possess common defects such as noise and distortion. Figure 1 shows a picture of the robot and the $4.4m \times 2.9m$ playing field.

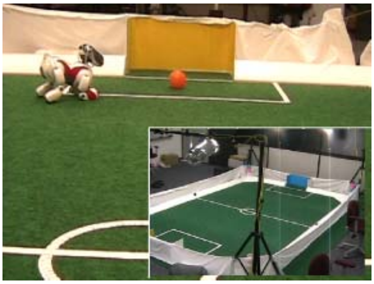


Fig. 1. An Image of the Aibo and the field

On the robot, visual processing typically occurs in two stages: color segmentation and object recognition ([3] presents our implementation). Color segmentation is a well-researched field in computer vision with several good algorithms, for example [2, 12]. But these involve computation that is infeasible to perform on autonomous robots given the computational and memory constraints. In the RoboCup domain too, several methods have been applied, from the baseline approach of creating mappings from the YCbCr values to the color labels [4], to the use of decision trees [13] and axis-parallel rectangles in the color space [14]. All of them involve an elaborate training process wherein the color map is generated by hand-labeling several ($\approx 20 - 30$) images over a period of at least an hour.

The color map is used to segment the image pixels to one of the desired colors and construct connected constant-colored blobs. The blobs are used to detect useful objects (e.g. markers and the ball). The robot uses the markers to localize itself on the field and coordinates with its team members to score goals on the opponent. All processing, for vision, localization, locomotion, and action-selection, is performed on board the robots, using a 576MHz processor.

Though games are currently played under constant and reasonably uniform lighting conditions, a change in illumination over several days forces teams to re-calibrate the vision system. Also, the overall goal of eventually playing against humans in natural lighting puts added emphasis on the ability to learn the color map in a very short period of time. Attempts to automatically learn the color map have rarely been successful. One such instance is [7]), wherein the author presents a method to learn the color map using three layers of color maps with increasing precision levels. But the generated map is reported to be not as accurate as the hand-labeled one and other domain specific constraints are introduced to disambiguate between object colors, during the object recognition phase. In [9], colors are estimated using a hierarchical Bayesian model with *Gaussian* priors and a joint posterior on robot position and environmental illumination.

This paper presents a novel approach that enables the robot to autonomously learn the entire color map, using the inherent structure of the environment and about seven images, in less than five minutes. It involves very little storage and the resultant segmentation accuracy is comparable to that obtained by the color map generated by the hand-labeling process.

3 Problem Specification

Here, we formally describe the problem of generating a color map for the robot.

To be able to recognize objects and operate in a color-coded world, a robot generally needs to recognize a certain discrete number (N) of colors ($\omega \in [0, N-1]$). A complete mapping identifies a color label for each possible point in the color space [5] under consideration:

$$\forall p, q, r \in [0, 255] \qquad \{C_{1,p}, C_{2,q}, C_{3,r}\} \mapsto \omega|_{\omega \in [0, N-1]} \tag{1}$$

where C_1, C_2, C_3 are the three color channels (e.g. RGB or YCbCr), with the corresponding values ranging from $0 - 255$.

We represent colors using a Three-Dimensional (3D) Gaussian model (reasonably approximates actual distributions) with the assumption of mutually independent color channels. In practice, the independence assumption, which implies a lack of correlation among the three color channel values for any given color, does not hold for all colors. Nonetheless, it closely approximates reality and greatly simplifies the calculations — computationally expensive operations such as inverting a covariance matrix need not be performed.

Each color $\omega \in [0, N-1]$ can then represented by the density distribution:

$$p(\omega|c_1, c_2, c_3) = \frac{1}{\sqrt{2\pi}\prod_{i=1}^{3}\sigma_{C_i}} \cdot \exp -\frac{1}{2}\sum_{i=1}^{3}\left(\frac{c_i - \mu_{C_i}}{\sigma_{C_i}}\right)^2 \tag{2}$$

where, $c_i \in [C_{i_{min}}, C_{i_{max}}]$ represents the value at a pixel along a color channel C_i while μ_{C_i} and σ_{C_i} represent the corresponding means and variances.

Under this model, the means and variances of the distributions are the only statistics that need to be collected and stored for each color that is to be learnt, making the learning process fast and feasible to execute on the robot. Next, we describe the learning setup and the actual process that the robot goes through to learn the color map.

4 Learning Setup

In this section we describe the algorithm (summarized in Algorithm 1) that the robot executes to autonomously learn the color distributions.

Algorithm 1. General Color Learning

```
Require: Starting Pose Known, Map of the robot's world.
Require: Empty color map.
Require: Array of poses for learning colors, $Pose[]$.
Require: Array of objects, described as shapes, from which the colors need to be
    learnt, $Objects[]$.
Require: Ability to move to a target pose.
$i = 0, N = MaxColors$
$Time_{st} = CurrTime$
while $i < N$ and $CurrTime - Time_{st} \leq Time_{max}$ do
  $Motion$ = RequiredMotion( $Pose[i]$ )
  Perform $Motion$ {Monitored using visual input}
  if LearnGaussParams( $Colors[i]$ ) then
    Learn Mean and Variance of color from candidate image pixels
    UpdateColorMap()
    if !Valid( $Colors[i]$ ) then
      RemoveFromMap( $Colors[i]$ )
    end if
  end if
  $i = i + 1$
end while
Write out the color statistics and the color map.
```

The algorithm can be described as follows: The robot starts off at a known position in its map of its world. It has no initial color information, i.e. the means and variances of the colors to be learnt are initialized to zero. It also has three lists: the list of colors to be learnt (*Colors*), a list of corresponding positions that are appropriate to learn those colors (*Pose*), and a list of corresponding objects, defined as shapes, that can be used to learn the colors. Using a navigation function (RequiredMotion()), the robot determines the motion required, if any, to place it in a position corresponding to the first entry in *Pose*, and executes the motion command. The object shape definition – the corresponding entry in the *Objects* array – leads to a set of constraints (heuristic *candidacy tests*) that are used to select the candidate blob. The robot stops when either a suitable blob is found or it thinks it has reached its target position. Further details of the *candidacy tests* can be found in a technical report [3].

Once in position, the robot executes the function *LearnGaussParams()* to learn the color. If a suitable candidate blob of *unknown* color (*black* in our case) exists, each pixel of the blob is examined. If the pixel value is sufficiently distant from the *means* of the other known color distributions, it is considered to be a member of the color class under consideration. When the entire blob has been analyzed, these pixels are used to arrive at a *mean* and a *variance* that then represent the *3D Gaussian density function* of the color being learnt.

The function *UpdateColorMap()* takes all the learned Gaussians as input and generates the complete mapping from pixel values to the color labels. This process of assigning color labels to each cell in the 128 × 128 × 128 cube is the most intensive part of the learning process. Hence, it is performed only once very five seconds or so. Each cell is assigned a color label corresponding to the color whose density function (Equation 2) has the largest *probability* value. The updated color map is used to segment all subsequent images.

The segmented images are used for detecting objects, which are in turn used to validate the colors learnt (*Valid()*). The entire learning procedure is repeated until all desired colors are learnt and/or the predecided learning time ($Time_{max}$) has elapsed. A more detailed description can be found in [11].

5 Experimental Setup

A line drawing of the legged league field, with its color coded goals and markers, is shown in Figure 2. We present the results when the robot always starts off in *Position-1* and moves through a deterministic sequence of positions (the elements of the array *Pose[]*).

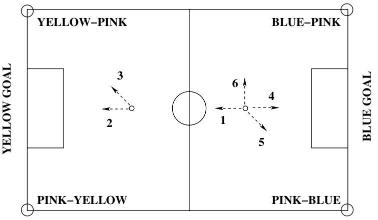


Fig. 2. The Learning positions

The steps involved in the algorithm can be presented as an ordered list of positions, colors (to be learnt) and objects:

1. Step-1: Position1 with head tilted down, *white* and *green*, Field line and center circle.
2. Step-2: Position-2, *yellow*, Yellow goal.
3. Step-3: Position-3, *pink*, Yellow-pink marker.
4. Step-4: Position-4, *blue*, Blue goal.

5. Step-5: Position-5, *blue* (Disambiguate *green* and *blue*), Pink-blue marker.
6. Step-6: Position-6 with head tilted down, ball color (*orange*), Ball.
7. Step-7: Position-6 with head horizontal, opponent's uniform color, Opponent.

The robot then writes out the color map and the statistics to the memory stick. A few important points are to be noted with regard to the learning process. In *Position-1*, learning is performed based on the fact that a large portion of the image (in that position) consists of *green*. The algorithm is entirely dependent on inherent structure of the environment and not on the particular color that is being learnt. The positions for learning the ball and opponent colors are set so as to minimize the movement. Currently we only learn *red* for the opponent uniform color, though the process could be used to learn *darkblue* too. The video of the learning mechanism, as seen from the robot's camera, can be viewed online [1].

6 Experimental Results

We tested the accuracy of the color maps that were learned autonomously on the robots by comparing their segmentation accuracy with a color map generated by the prevalent approach of hand-segmenting a set of ≈ 25 images. We refer to this color map as the *Hand Labeled* (*HLabel*) color map. This map corresponds to a fixed illumination condition. Here, an intermediate map (IM) of the same size as the overall color map is maintained for each color. Each cell of an IM stores a count of the number of times an image pixel that maps into that cell was labeled as the corresponding color. Each cell in the final color map is assigned the label corresponding to the color whose IM has the largest count in that cell.

Based on results [6, 8, 10] that the *LAB* color space could be reasonably robust to illumination variations, we trained a color map each in *LAB* and *YCbCr*. Since the colors of the ball and the opponent overlap with the marker colors, we performed the analysis in stages: first with just the fixed marker colors and then with all the colors included.

On a set of sample images of the markers (15) captured using the robot's camera, we first compared the performance of the three color maps with the color labeling provided interactively by a human observer, the *Ground Truth* (*GTruth*). We are interested only in the colors of the markers and other objects on the field and/or below the horizon. Also, the *correct* classification result is unknown (even with *HLabel*) for several background pixels in the image. Therefore, the observer only labels pixels suitable for analysis and these labels are compared with the classification provided by the three color maps. On average, $\approx 20\%$ of the pixels in the image get labeled by the observer. The average classification accuracies are 87.8 ± 3.18, 97.9 ± 0.76, and 98.8 ± 0.44 for the *YCbCr*, *LAB* and *HLabel* color maps respectively, as compared to *GTruth*.

The color labeling obtained by using the *HLabel* color map or the map generated in the *LAB* color space is almost perfect in comparison to the human color labeling. There is not much difference in the qualitative performance between these and the *YCbCr* map. Sample image results are available in [11, 1] – the robot is able to learn a reasonable color map in both color spaces when only the marker colors are considered.

Next, we let the robot learn the ball color (*orange*) in addition to the marker colors. The average classification accuracies are 74.8 ± 9.2%, 94 ± 5.6% and 98.5 ± 0.8% for the *YCbCr*, *LAB* and *HLabel* color maps respectively, as compared to *GTruth*. Figure 3 show the segmentation results over a set of images.

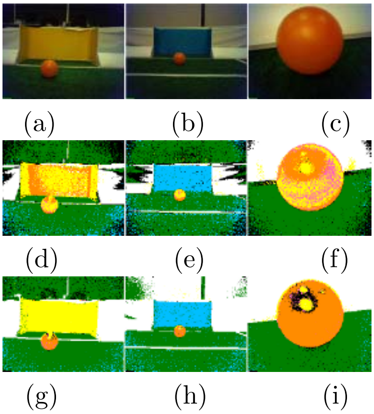


Fig. 3. Sample Images with Ball. (a)-(c) Original, (d)-(f) YCbCr, (g)-(i) LAB

In the *YCbCr* color space, the inclusion of *orange* causes the segmentation to degrade even over the colors (*pink* and *yellow*) that it could classify well before. This is not the case in *LAB* – the object recognition procedure is able to find the ball without any additional constraints (the ball is rarely found in *YCbCr*). Therefore the color of the opponent's uniform (*red*) is learnt only in the *LAB* color space. Images illustrating this can be seen at [11, 1].

While operating in the *LAB* color space, we still do not want to transform each pixel in the test image from *YCbCr* to *LAB* due to computational constraints. So, during the color map update, we assign the color label to each discrete cell in the *YCbCr* color map by determining the label assigned to the corresponding pixel values in *LAB*. The pixel-level transformation increases in the training time. The learning process takes $\approx 2.5 minutes$ in *YCbCr* while it takes $\approx 4.5 minutes$ in *LAB*, still much smaller than the time taken to generate *HLabel*, an hour or more.

When the illumination changes within a range of illuminations, the original color map does not perform well. But the robot is able to learn a new color map in a few minutes.

Finally, we tested the hypothesis that the algorithm is robust to color *re-mapping*. We changed the field setting by moving a goal to a carpet that has a non-uniform blue design and we placed a small piece of white paper on it instead of the field lines. The robot still learnt the carpet color as *green* and proceeded to learn other colors. Next, we started the learning process with the robot in *Position-2*, facing the blue goal (Figure 2). The robot ended up learning the color *blue* as *yellow* and vice versa. This confirms our hypothesis that the process is dependent only on shape and size and not on the particular color that is being learnt.

Sample image results for all experiments can be seen in [11] or on the team web-site [1].

7 Discussion and Conclusion

We have presented an approach to automating the color learning and segmentation process on-board a legged robot with limited computational and storage resources. In spite of the relatively low-resolution images with inherent noise and distortion, the algorithm enables the robot to autonomously generate its color map in a very short period of time. The corresponding segmentation accuracy is comparable to the that obtained by hand-labeling several images over a period of an hour or more. This could result in a substantial reduction in the setup time before the games can begin at RoboCup competitions.

Though we have tested our approach only in the legged league environment, it applies to the other leagues where the vision is done on-board a mobile robot in a known, color-coded environment. Color-calibration in the small-size league is currently more straightforward because vision is often done with a stationary overhead camera. However, as teams move towards on-board vision, they will face the same constraints as the other robot soccer leagues. To apply this method in the rescue league requires the generation of a test-environment with objects of relevant colors in known locations. One could imagine quickly collecting relevant training objects and placing them in fixed locations that can be communicated to the robot for training purposes. However it remains to be shown that doing so generalizes to the larger environment, and if so, that it enables a reduction in manual effort and training time. This is an important area for future research.

The algorithm depends only on the structure inherent in the environment and a *re-mapping* of the colors does not prevent the robot from learning them. Further, the color map can be learnt in several fixed illumination conditions between a minimum and maximum on the field. The learning can be easily repeated if a substantial variation in illumination is noticed.

Currently, the color map is learnt from a known fixed starting position without any prior knowledge of colors. An extension that we are currently working on is to learn from any given starting position on the field.

Acknowledgments

We would like to thank the members of the UT Austin Villa team for their efforts in developing the soccer-playing software mentioned in this paper. This work was supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

1. The Utaustinvilla research website, 2004. `http://www.cs.utexas.edu/users/AustinVilla/?p=research/auto_vis`.
2. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

3. Peter Stone et al. UT Austin Villa 2004: Coming of Age, AI Technical Report 04-313. Technical report, Department of Computer Sciences, University of Texas at Austin, October 2004.
4. William Uther et al. Cm-pack'01: Fast legged robot walking, robust localization, and team behaviors. In *The Fifth International RoboCup Symposium*, Seattle, USA, 2001.
5. R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Prentice Hall, 2002.
6. J. Hyams, M. W. Powell, and R. R. Murphy. Cooperative navigation of micro-rovers using color segmentation. *In Journal of Autonomous Robots*, 9(1):7–16, 2000.
7. M. Jungel. Using layered color precision for a self-calibrating vision system. In *The Eighth International RoboCup Symposium*, Lisbon, Portugal, 2004.
8. B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire. Low-order-complexity vision-based docking. *IEEE Transactions on Robotics and Automation*, 17(6):922–930, 2001.
9. D. Schulz and D. Fox. Bayesian color estimation for adaptive vision-based robot localization. In *The IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.
10. M. Sridharan and P. Stone. Towards illumination invariance in the legged league. In *The Eighth International RoboCup Symposium*, Lisbon, Portugal, 2004.
11. M. Sridharan and P. Stone. Autonomous color learning on a mobile robot. In *The Twentieth National Conference on Artificial Intelligence (AAAI)*, 2005.
12. B. Sumengen, B. S. Manjunath, and C. Kenney. Image segmentation using multi-region stability and edge strength. In *The IEEE International Conference on Image Processing (ICIP)*, September 2003.
13. Sony Legged League Team UNSW. *RoboCup-2001: The Fifth RoboCup Competitions and Conferences.* Springer Verlag, Berlin, 2002.
14. Sony Legged League Team UPennalizers. *RoboCup-2003: The Seventh RoboCup Competitions and Conferences.* Springer Verlag, Berlin, 2004.

Traction Control for a Rocker-Bogie Robot with Wheel-Ground Contact Angle Estimation

Mongkol Thianwiboon and Viboon Sangveraphunsiri

Robotics and Automation Laboratory, Department of Mechanical Engineering
Faculty of Engineering, Chulalongkorn University,
Phayathai Rd. Prathumwan, Bangkok 10330, Thailand
kieng@eng.chula.ac.th,
viboon.s@eng.chula.ac.th
http://161.200.80.142/mech

Abstract. A method for kinematics modeling of a six-wheel Rocker-Bogie mobile robot is described in detail. The forward kinematics is derived by using wheel Jacobian matrices in conjunction with wheel-ground contact angle estimation. The inverse kinematics is to obtain the wheel velocities and steering angles from the desired forward velocity and turning rate of the robot. Traction Control is also developed to improve traction by comparing information from onboard sensors and wheel velocities to minimize wheel slip. Finally, a simulation of a small robot using rocker-bogie suspension has been performed and simulate in two conditions of surfaces including climbing slope and travel over a ditch.

1 Introduction

In rough terrain, it is critical for mobile robots to maintain maximum traction. Wheel slip could cause the robot to lose control and trapped. Traction control for low-speed mobile robots on flat terrain has been studied by D.B.Reister, M.A.Unseren [2] using pseudo velocity to synchronize the motion of the wheels during rotation about a point. Sreenivasan and Wilcox [3] have considered the effects of terrain on traction control by assume knowledge of terrain geometry, soil characteristics and real-time measurements of wheel-ground contact forces. However, this information is usually unknown or difficult to obtain in practice. Quasi-static force analysis and fuzzy logic algorithm have been proposed for a rocker-bogie robot [4].

Knowledge of terrain geometry is critical to the traction control. A method for estimating wheel-ground contact angles using only simple on-board sensors has been proposed [5]. A model of load-traction factor and slip-based traction model has been developed [6]. The traveling velocity of the robot is estimated by measure the PWM duty ratio driving the wheels. Angular velocities of the wheels are also measured then compare with estimated traveling velocity to estimate the slip and perform traction control loop.

In this research, the method to estimate the wheel-ground contact angle and kinematics modeling of a six-wheel Rocker-Bogie robot are described. A traction control is proposed and integrated with the model then examined by simulation.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 682 – 690, 2006.

2 Wheel-Ground Contact Angle Estimation

Consider the left bogie on uneven terrain, the bogie pitch, μ_1, is defined with respect to the horizon. The wheel center velocities v_1 and v_2 parallel to the wheel-ground tangent plane. The distance between the wheel centers is L_B.

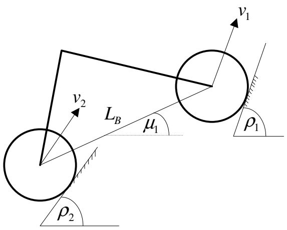


Fig. 1. The left bogie on uneven terrain

The kinematics equations can be written as following

$$v_1 \cos(\rho_1 - \mu_1) = v_2 \cos(\rho_2 - \mu_1) \tag{1}$$

$$v_1 \sin(\rho_1 - \mu_1) - v_2 \sin(\rho_2 - \mu_1) = L_B \dot{\mu}_1 \tag{2}$$

Define $a_1 = L_B \dot{\mu}_1 / v_1$, $b_1 = v_2 / v_1$, $\delta_1 = \rho_1 - \mu_1$ and $\varepsilon_1 = \mu_1 - \rho_2$ then
The contact angles of the wheel 1 and 2 are given by

$$\rho_1 = \mu_1 + \arcsin\left[(a_1^2 - b_1^2)/2a_1\right] \tag{3}$$

$$\rho_2 = \mu_1 + \arcsin\left[(1 + a_1^2 - b_1^2)/2a_1\right] \tag{4}$$

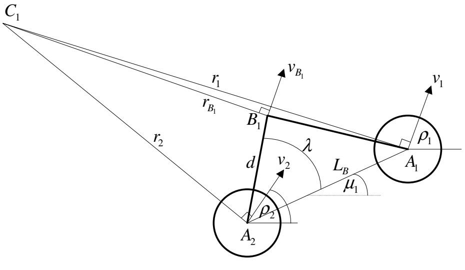


Fig. 2. Instantaneous center of rotation of the left bogie

Velocity of the bogie joint can be written as:

$$v_{B_1} = r_{B_1} \dot{\mu}_1 \tag{5}$$

Consider Left Rocker, the rocker pitch, τ_1, is defined with respect to the horizon direction. The distance between rear wheel center and bogie joint is L_R.

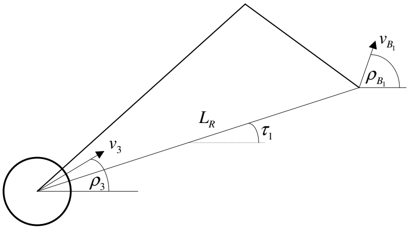


Fig. 3. Left Rocker on uneven terrain

$$\rho_3 = \arccos[(v_{B_1} / v_3)\cos(\rho_{B_1} - \tau_1)] \tag{6}$$

For the right side, the contact angles can be estimated in the same way.

3 Forward Kinematics

We define coordinate frames as in Fig. 4. The subscripts for the coordinate frames are as follows: O: robot frame, D: Differential joint, R_i: Left and Right Rocker ($i = 1,2$), B_i: Left and Right Bogie ($i = 1,2$), S_i: Steering of left front, left back, right front and right back wheels ($i = 1,3,4,6$) and A_i: Axle of all wheels ($i = 1-6$).Other quantities shown are steering angles ψ_i ($i = 1,3,4,6$), rocker angle β, left and right bogie angle γ_1 and γ_2. By using the Denavit-Hartenburg parameters [7], the transformation matrix for coordinate i to j can be written as follows:

$$\mathbf{T}_{j,i} = \begin{bmatrix} C(\eta_j) & -S(\eta_j)C(\alpha_j) & S(\eta_j)S(\alpha_j) & a_jC(\eta_j) \\ S(\eta_j) & C(\eta_j)C(\alpha_j) & -C(\eta_j)S(\alpha_j) & a_jS(\eta_j) \\ 0 & S(\alpha_j) & C(\alpha_j) & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

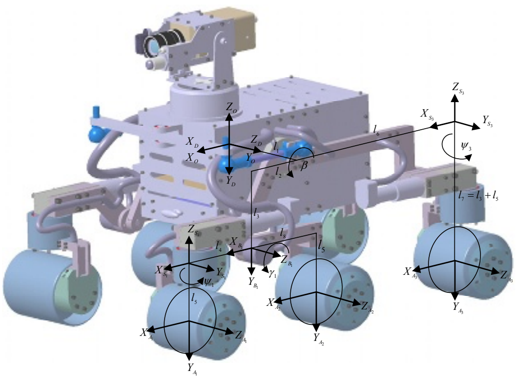


Fig. 4. Robot left coordinate frames

The transformations from the robot reference frame (O) to the wheel axle frames (A_i) are obtained by cascading the individual transformations.

For example, the transformations for wheel 1 are

$$\mathbf{T}_{O,A_1} = \mathbf{T}_{O,D}\mathbf{T}_{D,S_1}\mathbf{T}_{S_1,A_1} \tag{8}$$

To capture the wheel motion, we derive two additional coordinate, contact frame and motion frame. Contact frame is obtained by rotating the wheel axle frame (A_i) about the z-axis followed by a 90 degree rotation about the x-axis. The z-axis of the contact frame (C_i) points away from the contact point as shown in Fig. 5.

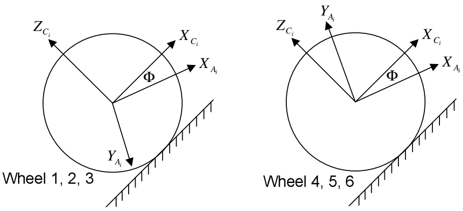


Fig. 5. Contact Coordinate Frame

The transformations for contact frame are derived using Z-X-Y Euler angle

$$\mathbf{T}_{A_i,C_i} = \begin{bmatrix} Cp_iCr_i - Sp_iSq_iSr_i & Cr_iSp_i + Cp_iSq_iSr_i & -Cq_iSr_i & 0 \\ -Cq_iSp_i & Cp_iCq_i & Sq_i & 0 \\ Cr_iSp_iSq_i + Cp_iCr_i & -Cp_iCr_iSq_i + Sp_iSr_i & Cq_iCr_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

The wheel motion frame is obtained by translating along the negative z-axis by wheel radius (R_w) and translating along the x-axis for wheel roll ($R_w\theta_i$).

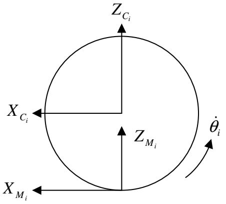


Fig. 6. Wheel Motion Frame

The transformation matrices for the front left wheel can be written as (10) and the transformation for other wheels can be written in the same way.

$$\mathbf{T}_{O,M_1} = \mathbf{T}_{O,D}\mathbf{T}_{D,B_1}\mathbf{T}_{B_1,S_1}\mathbf{T}_{S_1,A_1}\mathbf{T}_{A_1,C_1}\mathbf{T}_{C_1,M_1} \tag{10}$$

To obtain the Jacobian matrices, the robot motion is express in the wheel motion frame, by applying the instantaneous transformation $\dot{\mathbf{T}}_{\hat{O},O} = \mathbf{T}_{\hat{O},\hat{M}_i}\dot{\mathbf{T}}_{M_i,O}$

$$\dot{\mathbf{T}}_{\hat{O},O} = \begin{bmatrix} 0 & -\dot{\phi} & \dot{p} & \dot{x} \\ \dot{\phi} & 0 & -\dot{r} & \dot{y} \\ -\dot{p} & \dot{r} & 0 & \dot{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

where ϕ, p, r = yaw, pitch, row angle of the robot respectively.

Once the instantaneous transformations are obtained, we can extract a set of equations relating the robot's motion in vector form $[\dot{x} \quad \dot{y} \quad \dot{z} \quad \dot{\phi} \quad \dot{p} \quad \dot{r}]^T$ to the joint angular rates. The results of the left and right front wheel are found to be

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} A_i & 0 & B_i & C_i \\ D_i & 0 & E_i & F_i \\ G_i & 0 & H_i & I_i \\ 0 & 0 & 0 & J_i \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & K_i \end{bmatrix} \begin{bmatrix} \dot{\theta}_i \\ \dot{\beta} \\ \dot{\gamma}_i \\ \dot{\psi}_i \end{bmatrix} \qquad i = 1,4 \tag{12}$$

The results of wheel 2 and 5 (the left and right middle wheel) are found to be

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} A_i & 0 & B_i \\ C_i & 0 & 0 \\ D_i & 0 & E_i \\ 0 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_i \\ \dot{\beta} \\ \dot{\gamma}_i \end{bmatrix} \qquad i = 2,5 \tag{13}$$

The results of wheel 3 and 6 (the left and right back wheel) are found to be

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{p} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} A_i & 0 & B_i \\ C_i & 0 & D_i \\ E_i & 0 & F_i \\ 0 & 0 & G_i \\ 0 & -1 & 0 \\ 0 & 0 & H_i \end{bmatrix} \begin{bmatrix} \dot{\theta}_i \\ \dot{\beta} \\ \dot{\psi}_i \end{bmatrix} \qquad i = 3,6 \tag{14}$$

The parameters A_i to K_i in the matrices above can be easily derived in terms of wheel-ground contact angle $(\rho_1,..,\rho_6)$ and joint angle $(\beta, \gamma, \text{and } \psi)$.

4 Wheel Rolling Velocities

Consider forward kinematics of the front wheel (12), define $\dot{x}_d$ as the desired forward velocity and $\dot{\phi}_d$ as desired heading angular rate. The 1st and 4th equation give

$$\begin{aligned} \dot{x}_d &= A_i\dot{\theta}_i + B_i\dot{\gamma}_i + C_i\dot{\psi}_i \\ \dot{\phi}_d &= J_i\dot{\psi}_i \end{aligned} \qquad i = 1,3 \tag{15}$$

The rolling velocities of the front wheels can be written as

$$\dot{\theta}_i = (\dot{x}_d - B_i\dot{\gamma}_i - \frac{C_i}{J_i}\dot{\phi}_d)/A_i \qquad i = 1,3 \tag{16}$$

Similarly, the rolling velocities of the middle and rear wheels can be written as

$$\dot{\theta}_i = (\dot{x}_d - B_i\dot{\gamma}_i)/A_i \qquad i = 2,5 \tag{17}$$

$$\dot{\theta}_i = (\dot{x}_d - \frac{B_i}{G_i}\dot{\phi}_d)/A_i \qquad i = 3,6\ . \tag{18}$$

5 Slip Ratio

In section 3 and 4, we assume that there is no side slip and rolling slip between wheel and ground. Then slip must be minimizing to guarantee accuracy of the kinematics model. The slip ratio S , of each wheel is defined as follows:

$$S = \begin{cases} (r\dot{\theta}_w - v_w)/r\dot{\theta}_w & (r\dot{\theta}_w > v_w : accelerating) \\ (r\dot{\theta}_w - v_w)/v_w & (r\dot{\theta}_w < v_w : decelerating) \end{cases} \tag{19}$$

where r = radius of the wheel
θ_w = rotating angle of the wheel
$r\dot{\theta}_w$ = wheel circumference velocity
v_w = traveling velocity of the wheel

S is positive when the robot is accelerating and negative when decelerating. The robot can travel stably when the slip ratio is around 0 and will be stuck when the ratio is around 1. By measuring of the wheel angles with information from the accelerometer, we can minimize slip so the traction of the robot is improved.

In the traction control loop, a desired slip ratio S_d is given as an input command. The feedback value $\hat{S}$ is computed from a slip estimator. To complete the estimation of the slip, we need the rolling velocity and the traveling velocity of the wheels, ω and v_w. Rolling velocity of the wheels is easily obtained from encoders which installed in all wheels. Traveling velocity of the wheel can be computed from robot velocity by using data from onboard accelerometer.

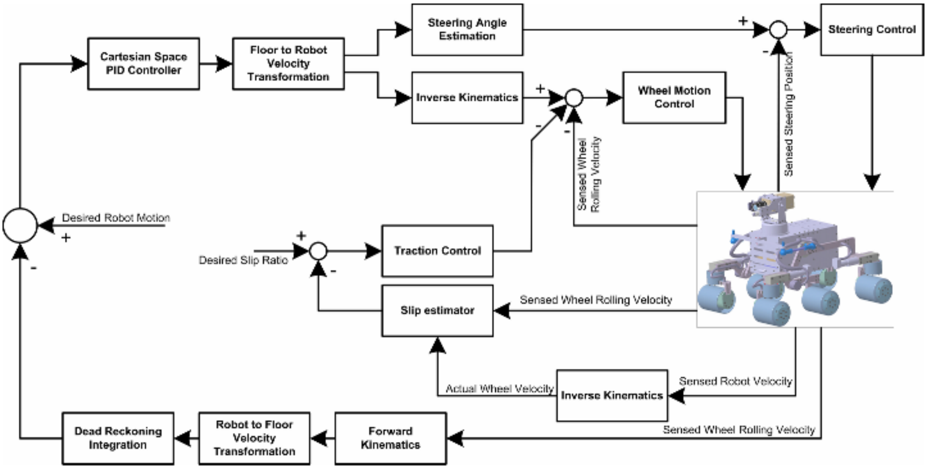

Cartesian Space PID Controller
Floor to Robot Velocity Transformation
Steering Angle Estimation
Inverse Kinematics
Wheel Motion Control
Steering Control
Sensed Steering Position
Sensed Wheel Rolling Velocity
Desired Robot Motion
Desired Slip Ratio
Traction Control
Slip estimator
Sensed Wheel Rolling Velocity
Actual Wheel Velocity
Inverse Kinematics
Sensed Robot Velocity
Dead Reckoning Integration
Robot to Floor Velocity Transformation
Forward Kinematics
Sensed Wheel Rolling Velocity
+
-

Fig. 7. Robot Control Schematic

6 Experiment

The system was verified in Visual Nastran 4D. In Fig. 8, the robot climbs up a 30-degree slope, with coefficient of friction 0.5. Without control, the robot move at 55 mm/s, then the front wheels touched the slope at $t = 0.5$ sec. and begin to climb up. Robot velocity reduced to 25 mm/s. But the robot continues to climb until the middle wheels touch the slope at $t = 9$ sec. The velocity reduced to nearly zero. With control, the sequence was almost the same until $t = 0.5$ sec. Then the velocity reduced to 35 mm/s when the front wheels touched the slope. The middle wheels touched the slope at $t = 6$ sec. and velocity reduced to 28 mm/s. Both back wheels begin to climb up the slope at $t = 15$ sec. with velocity approximately 20mm/s.

In Fig. 9, the robot traversed over a 32mm depth and 73mm width ditch with coefficient of friction about 0.5. The robot move at 55 mm/s, then the front wheels went down the ditch at $t = 0.5$ sec. and begin to climb up when front wheels touch the

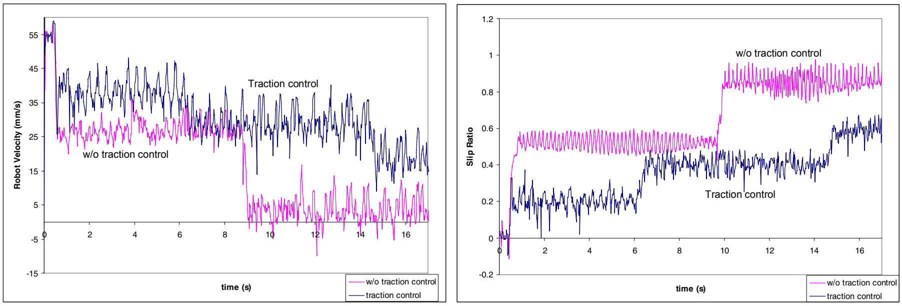


Fig. 8. Velocity and Slip ratio when climbed up 30 degrees slope

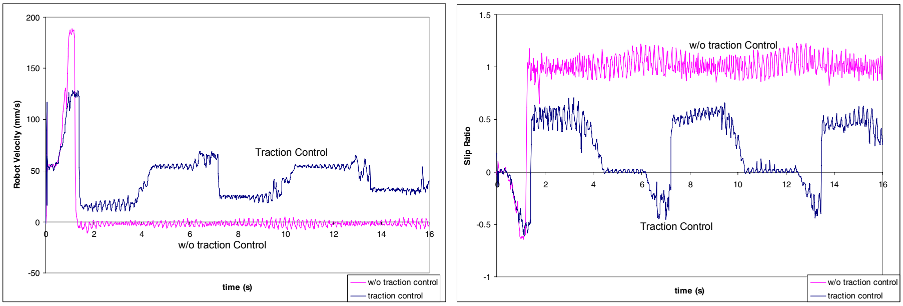


Fig. 9. Velocity and Slip ratio when traversed over a ditch

up-edge of the ditch. But the wheels slipped with the ground and failed to climb up. Then the slip ratio went up to 1 ($S = 1$), the robot has stuck at $t = 1.5$ sec.

With traction control, after the front wheels went down the ditch, the slip ratio was increased. Then the controller tried to decelerate to decrease the slip ratio. When the slip ratio was around 0.5, the robot continued to climb up. Until $t = 4.5$ sec., both of the front wheels went up the ditch completely and the robot velocity increased to the 55 mm/s as commanded. At $t = 6$ sec., the middle wheels went down the ditch. The robot velocity also increased temporary and back to 55 mm/s again when the middle wheels went up completely. The last two wheels went down the ditch at $t = 13$ sec. and the sequence was repeated in the same way as front and middle wheels.

8 Conclusion

In this research, the wheel-ground contact angle estimation has been presented and integrated into a kinematics modeling. Unlike the available methods that applicable to the robots operating on flat and smooth terrain, the proposed method uses the Denavit-Hartenburg notation like a serial link robot, due to the rocker-bogie suspension characteristics. A traction control is proposed based on the slip ratio. The slip ratio is estimated from wheel rolling velocities and the robot velocity. The traction control strategy is to minimize this slip ratio. So the robot can traverse over obstacle without being stuck. The traction control is verified in the simulation with two conditions. Climbing up the slope and moving over a ditch with coefficient of friction 0.5. The robot velocity and slip ratio are compared between using traction control and without using traction control system.

References

1. JPL Mars Pathfinder. February 2003. Available from: http://mars.jpl.nasa.gov/MPF
2. D.B.Reister, M.A.Unseren: Position and Constraint force Control of a Vehicle with Two or More Steerable Drive Wheels, IEEE Transaction on Robotics and Automation, page 723-731, Volume 9 (1993)

3. S.Sreenivasan, B.Wilcox: Stability and Traction control of an Actively Actuated Micro-Rover, Journal of Robotic Systems (1994)
4. H.Hacot: Analysis and Traction Control of a Rocker-Bogie Planetary Rover, M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA (1998)
5. K.Iagnemma, S.Dubowsky: Mobile Robot Rough-Terrain Control (RTC) for Planetary Exploration, Proceedings of the 26th ASME Biennial Mechanisms and Robotics Conference, Baltimore, Maryland (2000)
6. K.Yoshida, H.Hamano: Motion Dynamics of a Rover with Slip-Based Traction Model, Proceeding of 2002 IEEE International Conference on Robotics and Automation (2002)
7. John J. Craig: Introduction to Robotics Mechanics and Control, Second Edition, Addison-Wesley Publishing Company (1989)
8. M.Thianwiboon, V.Sangveraphunsiri, R.Chancharoen: Rocker-Bogie Suspension Performance, Proceeding of the 11th International Pacific Conference on Automotive Engineering (2001)

Velocity Control of an Omnidirectional RoboCup Player with Recurrent Neural Networks*

Mohamed Oubbati, Michael Schanz, Thorsten Buchheim, and Paul Levi

Institute of Parallel and Distributed Systems, University of Stuttgart
Universitaetsstrasse 38, D-70569 Stuttgart, Germany
{Mohamed.Oubbati, Michael.Schanz, Thorsten.Buchheim,
Paul.Levi}@informatik.uni-stuttgart.de
http://www.ipvs.uni-stuttgart.de

Abstract. In this paper, a recurrent neural network is used to develop a dynamic controller for mobile robots. The advantage of the control approach is that no knowledge about the robot model is required. This property is very useful in practical situations, where the exact knowledge about the robot parameters is almost unattainable. The proposed approach has been experimentally tested on an Omnidirectional RoboCup Player available at the Robotics Lab of the University of Stuttgart.

1 Introduction

Compared with the nonholonomic mobile robots, omnidirectional mobile robots provide superior manoeuvring capability. The ability to move simultaneously and independently in translation and rotation makes them widely studied in dynamic environmental applications. The annual RoboCup competition is an example where omnidirectional robots are widely used. However, quite few research studies on this type of robots have been reported. Most of them have been focused on the mechanical design and on the kinematic level control, assuming that there is *"perfect"* velocity tracking. However, as it is well known, the control at the kinematic level may be instable if there is errors control at the dynamic level. Therefore, the velocity control is at least as important as the position control. Recently, dynamic modelling and some analysis for omnidirectional robots have been addressed in [7, 1, 3]. In contrast with these theoretical developments, only few experimental works have been presented. In practical situations, exact knowledge about the dynamic model parameters values is almost unattainable. To deal with this weakness, there are possible methods, which can be used, even when the knowledge about the dynamic model is not complete, like robust adaptive control [4]. Another possible approach is to consider the robot as a "Black box", in order to avoid the estimation of its real parameters. Recurrent Neural Networks (RNNs) have a great potential for "black box" modelling, and they can give complementary/new solutions for system identification and control. Recently in [5], we developed a robot velocity control strategy based on

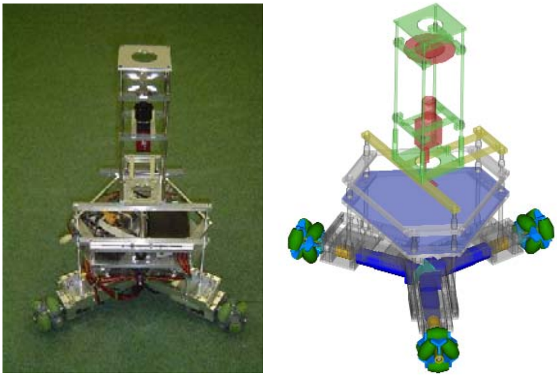
* This work was partially supported by the German Academic Exchange Service (DAAD).

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 691–701, 2006.

a novel RNN called Echo State Network (ESN). More recently in [6], and using the same strategy we demonstrated the ability of a single fixed-weight ESN to act as a dynamic controller for several distinct wheeled mobile robots. The trained controller showed high performances to balance between the variety of the desired velocity and the variety of the robots.

In this paper, we propose the velocity control based ESN as a new dynamic-control strategy for our robots. The controller is designed only by learning I/O data collected from the robot, without knowledge about its dynamic model.

The rest of this paper is organized as follows. Section 2 presents the omnidirectional robot and the problem to solve. A presentation of Echo state network is outlined in section 3, including its principles and training algorithm. The control approach is described in Section 4. In section 5, experimental results are presented. Finally, discussion and conclusion are drawn in section section 6.



(a) (b)

Fig. 1. Omnidirectional robot. a)hardware photo. b) CAD model.

2 Omnidirectional Robot

2.1 Hardware

Experimentations are performed on one omnidirectional robot (Fig. 1) of Soccer-robots team available at the robotics Lab of University of Stuttgart. The robot is equipped with 3 omni-wheels equally spaced at 120 degrees from one to another, and driven by three 90W DC motors. A personal computer on board is used to manage different sensors and tasks. For environment sensing, the robot is equipped by an omnidirectional vision system, based on a hyperbolic mirror and a standard IEEE1394 (FireWire) camera (Fig. 2). Omnidirectional vision provides our robot a very large field of view, which has some useful properties. For instance, it can facilitate the tracking of robots, the ball, and a set of environmental features used for self-localization. More hardware specifications of the robot are listed in Table 1.

Table 1. Robot Specifications

Max Acceleration	$3.5m/s^2$
Max. Speed	$2.5m/s$
Wheels	Omni-wheels (Swedish)
Host Computer	2,6GHz , RAM 600MB,60 GB HD
Power supply	Two 13 V batteries in series
Steering control	Interfaces with 3 DC motors
Dimensions	48,5cm wide, 80cm height
Weight	15 kg

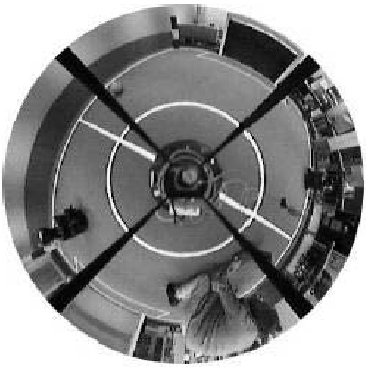


Fig. 2. An image from the omnidirectional camera

2.2 Kinematic Model

The geometry of the omnidirectional robot and its coordinate definitions are shown in Fig. 3. This model is used to transform linear and angular velocities to wheels speeds. It is chosen as in [3]:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} \sin(\theta) & \cos(\theta) & L \\ -\sin(\frac{\pi}{3}-\theta) & -\cos(\frac{\pi}{3}-\theta) & L \\ \sin(\frac{\pi}{3}+\theta) & -\cos(\frac{\pi}{3}+\theta) & L \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} \tag{1}$$

where w_i is the angular speed of wheel i, and x ,y and θ are the pose of the centre of mass (CM) of the robot. L is the distance of the wheels from the CM, and R is the radius of each Wheel. In our case: $R = 0,04$ and $L = 0,2$.

2.3 Control System

The control system is decomposed in two stages. An inner loop, depending on the robot dynamics and used to control linear and angular velocities, and an outer loop, which guarantees that the robot follows the desired trajectory. Both controllers are based on PID control strategy (Fig. 4). In practical situations, the

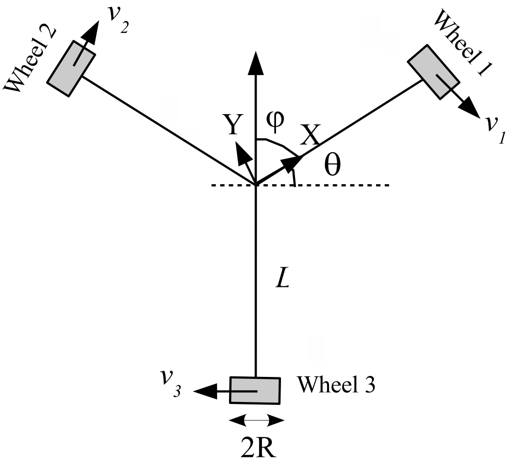


Fig. 3. Kinematic geometry

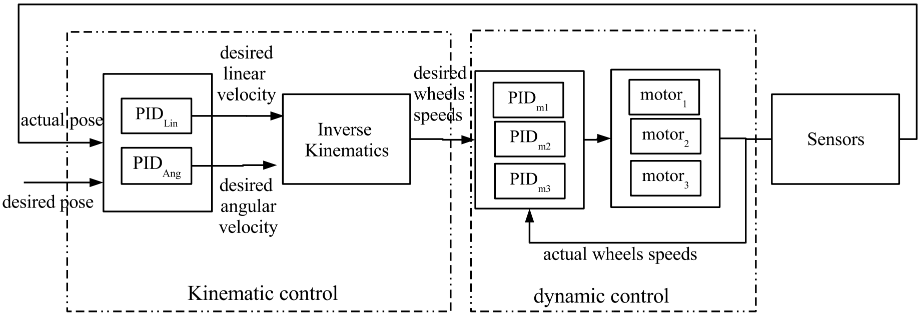


Fig. 4. Kinematic and dynamic control loops

assumption *"perfect"* velocity tracking at the dynamic-level is almost unattainable by the PID controllers implemented on board. The ability of PID controllers to cope with some complex properties of the robot such as non-linearities, friction, and time-varying parameters are known to be very poor. To improve the velocity control, we propose to use a novel RNN called Echo state network(ESN) to approximately model the whole dynamics of the robot as a "black box". Upon completion of the training procedure, we expect that the ESN controller will be capable to minimize reasonably errors between the desired and the actual robots velocities, without knowledge about the dynamic-model parameters.

3 Echo State Network

Echo state network is a RNN formed by a "Dynamic Reservoir"(DR), which contains a large number of sparsely interconnected neurons with non-trainable

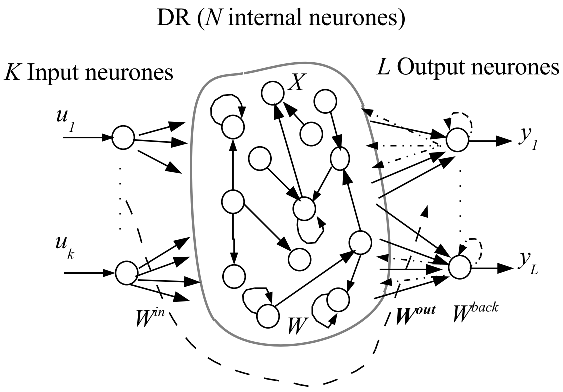


Fig. 5. Basic architecture of ESN. Dotted arrows indicate connections that are possible but not required.

weights. As presented in (Fig. 5), we consider that the network has K inputs, N internal neurones and L output neurones. Activations of input neurons at time step n are $U(n) = (u_1(n), u_2(n), \ldots, u_k(n))$, of internal units are $X(n) = (x_1(n), \ldots, x_N(n))$, and of output neurons are $Y(n) = (y_1(n), \ldots, y_L(n))$. Weights for the input connection in a $(N\text{x}K)$ matrix are $W^{in} = (w_{ij}^{in})$, for the internal connection in a $(N\text{x}N)$ matrix are $W = (w_{ij})$, and for the connection to the output neurons in an L x $(K + N + L)$ matrix are $W^{out} = (w_{ij}^{out})$, and in a $(N\text{x}L)$ matrix $W^{back} = (w_{ij}^{back})$ for the connection from the output to the internal units.

The activation of internal and output units is updated according to:

$$X(n+1) = f(W^{in}U(n+1) + WX(n) + W^{back}Y(n+1)) \tag{2}$$

where $f = (f_1, \ldots, f_N)$ are the internal neurons output sigmoid functions. The outputs are computed according to:

$$Y(n+1) = f^{out}(W^{out}(U(n+1), X(n+1), Y(n))) \tag{3}$$

where $f^{out} = (f_1^{out}, \ldots, f_L^{out})$ are the output neurons output sigmoid functions. The term $(U(n+1), X(n+1), Y(n))$ is the concatenation of the input, internal, and previous output activation vectors. The idea of this network is that only the weights connections from the internal neurons to the output (W^{out}) are to be adjusted.

Here we present briefly an off-line algorithm for the learning procedure:

1. Given I/O training sequence $(U(n), D(n))$
2. Generate randomly the matrices (W^{in}, W, W^{back}), scaling the weight matrix W such that its maximum eingenvalue $|\lambda_{max}| \leq 1$.
3. Drive the network using the training I/O training data, by computing

$$X(n+1) = f(W^{in}U(n+1) + WX(n) + W^{back}D(n)) \tag{4}$$

4. Collect at each time the state $X(n)$ as a new row into a state collecting matrix M, and collect similarly at each time the sigmoid-inverted teacher output $tanh^{-1}D(n)$ into a teacher collection matrix T.
5. Compute the pseudoinverse of M and put

$$W^{out} = (M^{-1}T)^t \tag{5}$$

t: indicates transpose operation.

The ESN is now trained. For exploitation, the network can be driven by new input sequences and using the equations (2) and (3). For more details, a complete tutorial on ESNs can be found in [2].

4 Control Approach

4.1 Controller Training

Training data (PWMs/wheels speeds) are collected by moving the robot "arbitrarily" with different velocities in different directions. 1000 sequences were collected from the robot and stored in a file. To train the ESN as a velocity controller, we used the bloc diagram depicted in Fig. 6. Using its training algorithm, the ESN learned the teacher signals(PWMs), which bring each wheel from an actual speed at time (n) to a future speed at time $(n + 1)$.

The ESN architecture was chosen as follows. 6 inputs (actual and delayed wheels speeds), 13 internal neurons and 3 outputs (PWMs duty ratio for each motor). No back-connection from the output to the DR, and no connections from the input directly to the output. The input and the internal synaptic connections weights were randomly initialized from a uniform distribution over $[-1, +1]$. The internal weight matrix W has a sparse connectivity of 20% and scaled such that its maximum eingenvalue $|\lambda_{max}| \approx 0.3$.

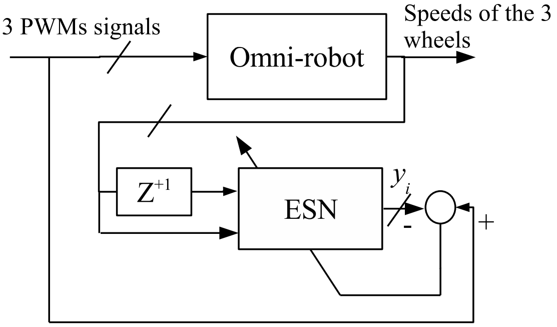


Fig. 6. Training of ESN as a dynamic controller for mobile robots

4.2 Control Procedure

After training procedure has been completed, the network was implemented in the control system on the host computer on-board (Fig. 7). Desired speeds of the wheels are first computed from the desired linear and angular velocities (v_d, w_d), using equation (1). Using the actual (measured) and desired wheel-speeds, the ESN send the appropriate PWMs duty via a serial (RS232) connection to an electronic interface, which produces the correspondent amplified PWMs voltage to the three motors.

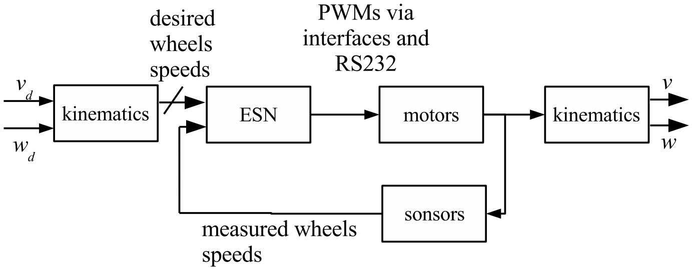


Fig. 7. Structure of the control system

5 Results

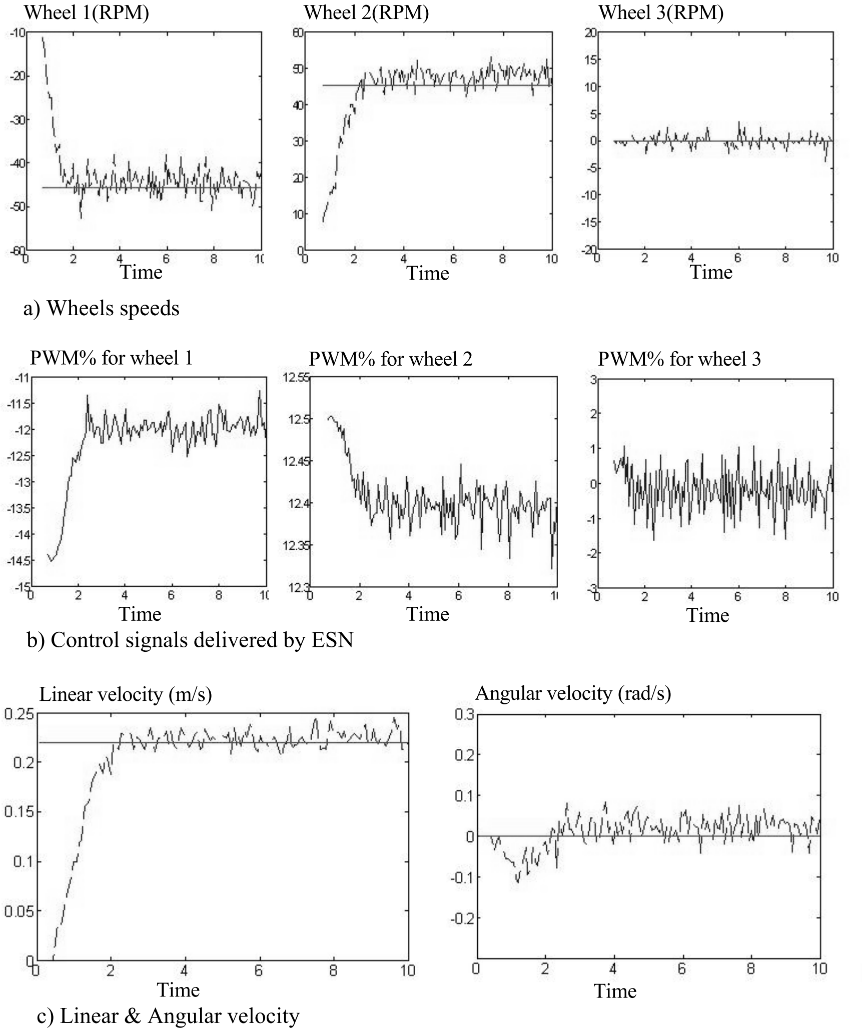
During experiments, we had to solve many practical problems. The first problem was training data. It is technically not possible to use random inputs(PWMs) to collect training data. The only realistic possibility available was to move the robot with different smooth "low" velocities (max $0.5m/s$), in order to avoid slippage of the wheels, and to keep the robot on the field. It is clear that using this method, training data will be not rich enough to give complete information about the robot dynamics. Another problem is the nature of the robot. Omnidirectional characteristics are obtained only by using omni-wheels (Swedish wheels in our case). However, it is known that these wheels are very sensitive to the road (carpet) condition and their performances are limited, compared with the conventional wheels. This limitation produces errors in speeds measurements. Other errors are also produced by the sensors, since they deal only with integers. Another problem is the network architecture. During preparation of the network, it was not easy to find its optimum parameters. Using a "relatively" large dimension (more than 30 internal neurons) the network lost stability at many times and exhibited sometimes high-frequency oscillations on smooth accelerations. This is due perhaps of the high degree of freedom of the closed loop Robot-ESN, and due to the poor training data. With small dimension (say 5-8 internal neurons), we minimize these oscillations, but at many times the network could not react quickly to the velocity variations, and some errors on the steady state control are obtained. With 13 internal neurons the ESN showed relatively an acceptable behavior.

After training, several experimental tests are performed; two of them are reported here. They cover some typical movements of a RoboCup player during a soccer game.

5.1 Experiment 1

In this experiment, the objective is to track the constant reference velocity:

$$\begin{cases} v_d(t) = 0.22\ m/s \\ w_d(t) = 0\ rad/s \quad 0s \leq t \leq 10s \\ \varphi = \frac{\pi}{3} \end{cases} \tag{6}$$



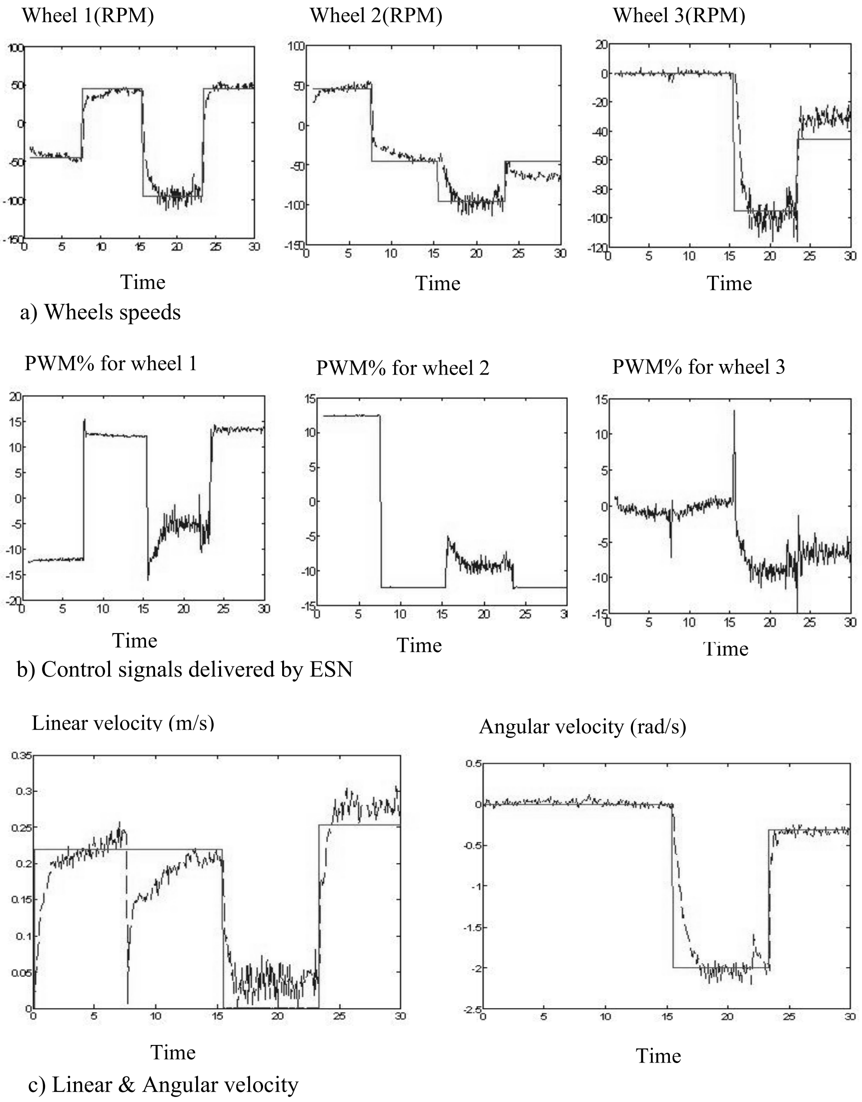
a) Wheels speeds

b) Control signals delivered by ESN

c) Linear & Angular velocity

Fig. 8. Results of Experiment 1. Desired velocity(solid) and actual robot velocity (dashed).

In this experiment, the robot should move straight a head(no angular velocity). Therefore, wheel 3 should be blocked. During this experiment, the ESN controller could bring the robot to the desired velocity, even the presence of high friction between the carpet and wheel 3, which explains the high frequency in the control signal delivered by the ESN for this wheel(Fig. 8).



a) Wheels speeds

b) Control signals delivered by ESN

c) Linear & Angular velocity

Fig. 9. Results of Experiment 2. Desired velocity(solid) and actual robot velocity (dashed).

5.2 Experiment 2

In this experiment, the objective is to track the time varying reference velocity:

$$\begin{cases} v_d(t) = 0.22\ m/s \\ w_d(t) = 0\ rad/s \quad 0s \leq t \leq 7s \\ \varphi = \frac{\pi}{3} \end{cases} \tag{7}$$

$$\begin{cases} v_d(t) = 0.22\ m/s \\ w_d(t) = 0\ rad/s \quad 7s \leq t \leq 15s \\ \varphi = \frac{4\pi}{3} \end{cases} \tag{8}$$

$$\begin{cases} v_d(t) = 0\ m/s \\ w_d(t) = -2\ rad/s\ 15s \leq t \leq 23s \\ \varphi = 0 \end{cases} \tag{9}$$

$$\begin{cases} v_d(t) = 0.25\ m/s \\ w_d(t) = -3\ rad/s\ 23s \leq t \leq 30s \\ \varphi = \frac{\pi}{3} \end{cases} \tag{10}$$

As shown in Fig. 9, during $[0s, 7s]$, the robot behaved like in experiment 1. In $[8s, 15s]$, the same desired linear velocity is given to the controller, but in other direction (according to the angle φ in Fig. 3. At time $8s$ the controller stopped the robot, and tried to reach again the same translation velocity in the new direction ($\varphi = \frac{4\pi}{3}$). During $[15s, 23s]$, the robot had to be turned around it self, since the desired translation velocity is zero, and the desired rotation velocity is-2 rad/s. In this situation, the wheels are more sensitive to the carpet, and exhibit higher frequency around the reference. In the last time interval $[23s, 30s]$, the ESN had to control the robot in a curve. During training, the ESN did not learn this situation. Despite the lack of information, the ESN could successfully control the robot independently in translation and rotation.

6 Conclusion

This paper has presented a velocity controller for an omnidirectional robot available at the robotics Lab of university of Stuttgart. The control approach is based on a novel RNN called "echo state network", which is trained in a way that only the output connections weights will be adjusted. The ESN control approach requires no prior information about the dynamics of the robot. This property makes it very useful in practical situation, where the exact knowledge about the mobile robot parameters is almost unattainable. Despite the "poor" quality of training data, and the performance limitation of the omni-wheels, the ESN controller could achieve acceptable control results. However, we are aware of a certain degree of arbitrariness in our choice of the controller network parameter and architecture. Therefore, substantial investigation on ESN architecture and more experiments on much larger data sets are still needed to improve the results achieved to date.

References

1. J.H. Chung, B. Yi, W.K. Kim, and H. Lee. ”the dynamic modeling and analysis for an omnidirectional mobile robot with three caster wheels”. In *Proc. IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, September 2003. IEEE.
2. H. Jaeger. ”tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach”. Technical Report 159, AIS Fraunhofer, St. Augustin, Germany, 2002.
3. Tamás Kalmár-Nagy, Raffaello D'Andrea, and Pritam Ganguly. ”near-optimal dynamic trajectory generation and control of an omnidirectional vehicle”. *Robotics and Autonomous Systems*, 46:47–64, 2004.
4. Min-Soeng Kim, Jin-Ho Shin, Sun-Gi Hong, and Ju-Jang Lee. ”designing a robust adaptive dynamic controller for nonholonomic mobile robots under modeling uncertainty and disturbances”. *Mechatronics*, 13:507–519, 2003.
5. M. Oubbati, P. Levi, and M. Schanz. ”recurrent neural network for wheeled mobile robot control”. *WSEAS Transaction on Systems*, 3:2460–2467, August 2004.
6. M. Oubbati, P. Levi, and M. Schanz. ”a fixed-weight rnn dynamic controller for multiple mobile robots”. In *24th IASTED International Conference on modelling, identification, and control*, pages 277–282, February 2005.
7. R. Williams, B. Carter, P. Gallina, and G. Rosati. Dynamic model with slip for wheeled omnidirectional robots. *IEEE transactions on Robotics and Automation*, 18(3), June 2002.

Very High Speed, Close Field, Object Positioning Using Tri-linear CCDs

David Jahshan

Department of Electrical and Electronic Engineering,
The University Of Melbourne
Victoria, 3010 Australia
dej@ee.mu.oz.au

Abstract. To be able to effectively intercept and control a soccer ball travelling at high speed, it is useful to be able to accurately track the position of the ball as it approaches the robot. In this paper we present a method that can calculate the position in two dimensions at thousands of frames per second using a pair of inexpensive tri-linear CCDs. Each CCD gathers RGB information, which is then colour segmented. This data is then fused to calculate the location of the object in 2D. Further, the amount of processing required to detect these objects is low, and can be accomplished using inexpensive electronic components.

1 Introduction

The ability to accurately track the position of an approaching soccer ball is very useful when attempting to intercept and control that ball. Using regular vision systems running at 30FPS it is possible to track a ball travelling at 10m/s every 1/3 of a metre. When attempting to intercept a ball with a diameter of 22 centimetres, that is a worst case error of one and a half ball length. For most ball control devices a few centimetres can be the difference between a ball being captured or awkwardly reflecting off the capture mechanism. This error can be reduced by predicting the projection of the ball from the previous frames.

However should a shot be taken close to the goal, the defending robot might get one or two frames before the ball reaches the back of the net. One solution is to use cameras with higher frame rates. It is possible to purchase cameras with 1000FPS or more. A ball travelling at 10m/s can be pictured every 1cm with such a camera. The cost of purchasing high speed cameras range in the thousands of dollars, and designing electronics to process the images in real time is challenging, and expensive.

Area CCDs have a two dimensional array of pixels. There are two ways to configure a colour CCD, one uses a single CCD with a mosaic of RGB filters placed over the pixels or the more expensive option of using a camera with three individual CCDs with a colour filter for the whole CCD. The mosaic filter on a standard CCD involves the collection of two rows of pixels before the RGB values can start to be extracted.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 702–707, 2006.

Unlike an area CCD a tri-linear CCD has three one dimensional strings of pixels. Each one of these strings of pixels has a colour filter. The three colour pixels are read out in parallel, making the RGB value available immediately.

Tri-linear CCDs are used in a variety of products such as colour scanners. In a scanner the CCD captures the page it is scanning one line at a time, and it is then moved to capture the next line. High resolution linear CCDs are also used in aerial [6] [10] and underwater surveys [4].

There are many linear CCDs on the market, with resolutions ranging from a few hundred pixels to tens of thousands of pixels. An example of a low cost tri-linear CCD is the Sony ILX558K [9]. It has 5340x3 pixels, data is transferred at a rate of 10Mhz and costs less than $US50. This gives this linear CCD a frame rate just over 1800FPS. At that rate, a ball travelling at 10m/s can be imaged every 5.5mm. Also, at a pixel rate of 10Mhz the image can be easily processed using inexpensive electronics. Faster linear CCDs are available such as the UPD3729 [7] by NEC which has 5000x3 pixels at a pixel rate of 30Mhz giving a frame rate of 6000FPS.

2 System Overview

The system consists of a pair of linear CCDs fitted with a cylindrical lens with high numeric aperture so that each CCD can cover a large area. Linear CCDs are capable of seeing up to a few metres away dependent on lighting and optics. The effective area of visualisation is the area of overlap between the two sensors.

Three voltages are generated per clock cycle from the CCD. These voltages are then fed into an application specific Analog to Digital Converter (ADC) that generates a 24 to 48 bit RGB value every clock cycle, (dependent on the ADC used). A number of these bits can be masked to reduce processing requirements.

The data is then processed by any of the many colour segmentation techniques available [5]. Because the data stream is linear, many of these algorithms are greatly simplified. Noise pixels however cause a problem. Many algorithms use pixels on the surrounding lines to cancel out noise. However since there are no surrounding lines in the same frame, these techniques are not applicable. Ignoring pixels that are out of character is the easiest method for eliminating noise. However more sophisticated methods using previous frame data could also be used.

Two streams of colour segmented data can then be merged to find the object. Object pairs of specific colours can be paired and triangulated.

3 System Constraints

The system is only capable of detecting objects on a fixed plane. There are also issues with having multiple objects of the same colour in the area of visualisation. This is less of a problem than when trying to do 3D object detection using two area CCDs because the area being visualised is small.

In Robocup, where the object of interest is a soccer ball, of which only one exists and is usually on the ground, this technique is very effective.

To increase the visualisation area an array of these sensors can be deployed to cover the area that is required.

4 Experimental Results

To confirm the feasibility of this type of visualisation technique, an Acer Prisa 620s flat bed scanner was used. The first experiment involved finding out the detection distance of this particular CCD. Figure 1 shows a 60mm x 70mm x 30mm orange box scanned from 2.2 metres away against a green background. The initial output from the scanner returned a black page, but once the contrast, brightness and colour balance was increased it was possible to make out the object.

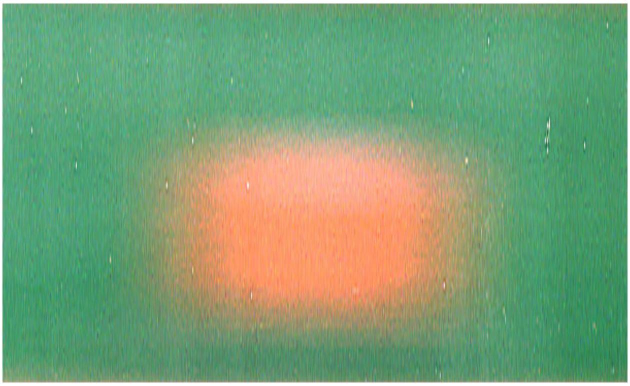


Fig. 1. Orange box on green background 2.2m away

In the second experiment a clear plastic box was used as an observation area. The box was placed in front of half the linear CCD. A mirror at 45^o to the observation area was used to reflect the light onto the other half of the linear CCD as seen in figure 2. Half the linear CCD observed the x axis whilst the other half observed the y axis of the observation area. A white marble with a diameter of 16mm was used as a target object, and was flicked around using a ruler. Black paper was used on the opposite sides of the clear observation area to mask off the background. The scan was initiated and the results appeared on the screen as a graph of position versus time.

Figure 3 shows data gathered over 2 seconds. The ball was flicked in the y direction at a slight angle. The ball hit the wall closest to the linear CCD and bounced back, indicated by the first inflection of the y axis. The ball continued at almost the same angle on the x axis. The ball then hit the wall to the right of the CCD. By this point most of the energy had been absorbed by the walls, and the ball started to roll to the centre of the plastic box.

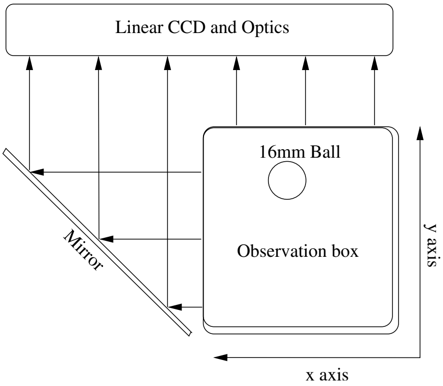


Fig. 2. Experimental setup

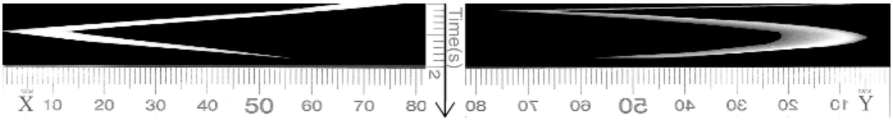


Fig. 3. Result of 2d imaging over 2 seconds

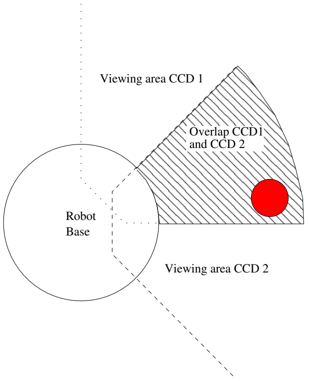


Fig. 4. Proposed setup of two of eight linear CCDs on the base of MU-Penguins robot

5 Implementation

In the system planned for MU Penguins mid sized robot team, eight tri-linear CCDs with a viewing angle of 90^o will be placed around the circular base of the robot. At 5340 pixels covering 90^o, the resolution of an object 2 metres away would be 0.6mm. Each tri-linear CCD will overlap 45^o of the previous CCD.

The signals from the eight CCDs are processed by the ADC and then fed into an FPGA [5] [3] [8] [1] [2]. The FPGA uses a lookup table to do colour segmentation. Object start points and end points are noted, and the object

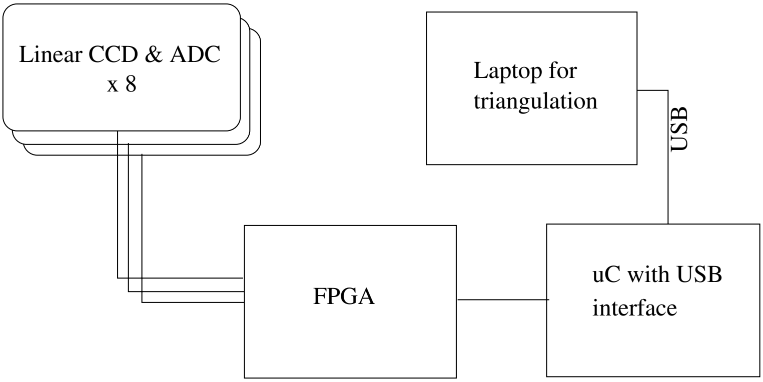


Fig. 5. Proposed processing of data from linear CCDs

table of each CCD is transferred to the robots laptop using USB. The laptop is responsible for carrying out the triangulation calculations.

6 Conclusion

The system proposed above is a low cost effective way of scanning for objects appearing on a fixed plane at many thousands of frames per second. Using two linear CCDs that are positioned such that there is overlap between them, it is possible to triangulate the position of objects. Further, the amount of processing required to detect these objects is low, and can be accomplished using inexpensive electronic components.

References

1. D.Demigny, L.Kessal, R.Bourgiba, and N Boudouani. How to use high speed recongifgurable fpga for real time image processing. *IEEE*, 2000.
2. Johan Andren-Dinerf et al. Chipvision a vision system for robots based on reconfigurable hardware. In *Proceedings of Robocup 2003 World Cup VII*, 2003.
3. K.Benkrid, D.Crookes, J.Smith, and A.Benkrid. High level programming for real time fpga based video processing. *IEEE*, 2000.
4. K.Moore, J.Jaffe, and B.Benjamin. Development of a new underwater bathymetric laser imaging system. *Journal of Atmospheric and Oceanic Technology*, 17(8), 2000.
5. L.Lucchese and S.K.Mitra. Color image segmentation: A state-of-the-art survey. In *Proc. of the Indian National Science Academy*, 2001.
6. M.Nakagawa, R.Shibasaki, and Y.Kagawa. Fusing stereo linear ccd images and laser range data for building 3d urban model. In *Geospatial Theory, Processing and Applications, ISPRS, Vol. XXXIV, part 4*, 2002.
7. NEC. upd3729 5000 pixels x 3 color ccd linear image sensor datasheet, 2001.
8. P.M.Curry, F.Morgan, and L.Kilmartin. Xilink fpga implementation of an image classifier for object detection applications. *IEEE*, 2001.

9. Sony. Ilx558k 5340-pixel x 3-line ccd linear colour sensor datasheet.
10. Y.Kagawa. Automatic acquisition of 3d city data with air-borne tls and laser scanner. In *Graduate School of Frontier Sciences, Institute of Environmental Studies, University of Tokyo, Japan*, 2001.

Visual Based Localization for a Legged Robot[⋆]

Francisco Martín, Vicente Matellán, Jose María Cañas, and Pablo Barrera

Robotic Labs (GSyC), ESCET, Universidad Rey Juan Carlos,
C/ Tulipán s/n CP. 28933 Móstoles (Madrid), Spain
{fmartin, vmo, jmplaza, barrera}@gsyc.escet.urjc.es

Abstract. This paper presents a visual based localization mechanism for a legged robot in indoor office environments. Our proposal is a probabilistic approach which uses partially observable Markov decision processes. We use a precompiled topological map where natural landmarks like doors or ceiling lights are recognized by the robot using its on-board camera. Experiments have been conducted using the AIBO Sony robotic dog showing that it is able to deal with noisy sensors like vision and to approximate world models representing indoor office environments. The major contributions of this work is the use of an *active vision* as the main input and localization in *not-engineered environments*.

1 Introduction

One of the basic tools for mobile robot operations is the localization capability [6]. It can be defined as the ability of a robot to determine its position in a map using its own sensors. Many works [13] have been developed to estimate the robot location as robot behavior may depend on its position inside the world.

Most of these algorithms, i.e. [11],have been designed for robots equipped with wheels, where locally accurate odometric information can be achieved and 360° sensory information is available. These requirement makes these methods unusable in legged robots. The solution we present is intended to solve the problem for a legged robot were odometric information is not reliable, even locally, and 360° sensory information is not available. This typoe of information from sonar or laser sensors is easier to process than camera images, available in our robot. The camera swinging in legged robots don't let a continuous image processing, making unusable the majority of the wheeled robot techniques for navigation, for instance.

In the literature, some works face this problem using vision as main sensor [3],[2] and [14], but most of them make their experiments in reduced and engineered environment (that is, placing ad-hoc landmarks or active beacons), mainly in the Robocup four legged league. In contrast, our work has been tested in a large office environment using natural pre-existing landmarks (doors, ceiling lights, etc). In addition, most of the approaches using vision as the main sensor for localization are *passive*, i.e. neither the sensor position, nor orientation are

[⋆] This work has been supported by grant DPI2004-07993-C03-01 of Spanish Government.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 708–715, 2006.

controlled. Our approach is *active*, as long as it commands the sensor orientation to get the information we need from the environment.

Another key difference with previous works is that our approximation is topological and most of the previously works use a metric approach. The office environment has been divided into states, where the set of nodes is built depending on the observations that can be obtained in each place of this environment (flat corridors and foyers). Others works using topological localization are, for instance [10], but again, these approaches are used in wheeled robots, where the information needed for applying these techniques are not available.

Our work is based on Partially Observable Markov Decision Processes [3]. We calculate a probability density (belief) over the entire states space (nodes of the topological map). These technique is also used in many other works as [1], [4] and [9], but using ultrasonic or infrared sensors to determine the obstacles around the robot, and using available odometry information. In [7], a markovian vision-based method is used, but the information extracted from the images are histograms and scale-invariant (SIFT) features [8] that are calculated from a wide set of image obtained in each map location. Our approach do not need previously taken images, only a 2D map where the landmarks needed for our model are displayed.

There are other approaches which use sampling algorithms for localization. In [12] *Monte Carlo* approximation is used, but in very controlled scenarios, and in [5] it has been reported that this technique is not effective in noisy environments as ours. Once again, these works have been tested in reduced and engineered environments (RoboCup mainly).

Summarizing, the major contributions of this paper are three:

1. The development of a probabilistic localization technique for legged robots. It has been tested in large environments. Other works previously cited using same robots are mainly devoted to small engineered spaces (mainly the Robocup playground).
2. The development of a topological framework for navigation in not-engineered indoor environments. Majority of works on legged robots localization has focused on metric localization in engineered environments.
3. The use of the robot on-board camera as the main sensor for existing landmarks (doors, ceiling lights, etc.) detection in natural indoor scenarios, and the control of the position of the camera.

Our work has been developed in an AIBO ERS7 robot. This robot is a completely autonomous robot which incorporates an embedded MIPS processor running at 576MHz, and 64MB of main memory. It gets information from the environment through a 350K-pixel color camera and 2 infrared sensors. AIBO locomotion main characteristic is its dog aspect with four legs.

The remainder of this paper is organized as follows: in section 2 we make a brief review of the Markov Localization technique used. In section 3 we describe our model and its components in detail. In section 4 the experiments and results are shown. Finally, we expose some conclusions and envisioned improvements in section 5.

2 Markovian Localization Framework

Localization based on indirect information provided by the robot sensors (sonar, laser, etc.) has been successfully integrated in the probabilistic framework and has exhibited good results [13]. In particular, sampling methods that speed up the estimation[3] are currently the most popular ones [11].

In our work, we have used a partially observable Markov decision processes (POMDP) where a probability distribution Bel, over all the possible locations $S = \{s_1, s_2, ...\}$ is maintained. $Bel_t(S = s)$ represents the belief of being in state s at time t. Depending on the knowledge about the initial position of the robot $Bel_0(S)$, the initial state will be uniformly distributed, if the position is not known. If the position is known, the distribution wil be centered in the initial state.

The belief Bel actualization is divided in two atomic steps:

Movement step. Robot actions are modelled by the probability $p(s'|s, a)$ (action model). This is the probability of reaching state s' if an action a is executed at state s. To obtain the a priori belief for the whole set of states $Bel_t(S')$ bayesian updating is assumed. When an action is executed we apply:

$$Bel_t(s') = \sum_{s \in S} p(s'|s, a) \cdot Bel_{t-1}(s), \forall s' \in S \tag{1}$$

Observation step. To calculate the corrected belief $Bel_t(S)$ we take $p(o|s)$ (sensor model) as the probability of getting the observation o being in the state s and we operate, as it is described in [9]. When a new independent observation is obtained the belief is updated using (2).

$$Bel_t(s) = p(\mathbf{o}|s) \cdot Bel_t(s'), \forall s, s' \in S. \tag{2}$$

3 Our Model

Summarizing, our localization method needs three components to be described, and these will be defined in more detail in this section:

1. The map and how it is translated to a set of states.
2. A set of actions that the robot can perform and their probabilistic action model related to states $p(s'|s, a)$
3. A set of observations the the robot perceives from the environment, and its probabilistic model related to states $p(o|s)$.

3.1 The State Space

We name possible locations of the robot as "states". These states are defined over an indoor office environment (see Fig. 1) made up by corridors (represented in

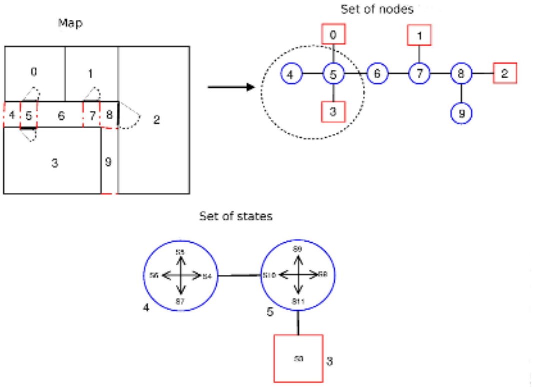


Fig. 1. The set of states is built from the map and its node decomposition

the set of nodes as circles) and rooms (represented as boxes). Nodes are defined as places with similar sensory characteristics.

Once the set of nodes has been defined, each node has been divided in four different states, representing the same robot position with four orientations: north, east, south, and west.

3.2 Action Model

The action primitives we have implemented in this work are: to turn 90° on the left $a_{\{T_L\}}$, to turn 90° on the right $a_{\{T_R\}}$and go forward $a_{\{F\}}$ until the next state with the same orientation is reached.

When the robot executes an action primitive, i.e. when the robot moves, it updates the belief as it is shown in (3). The action model defines $p(s'|s,a)$ as the probability of to reach state s', starting at state s and executing the action a:

$$p(s'|s,a), \forall s \in S, \forall a \in A = \{a_{\{F\}}, a_{\{T_L\}}, a_{\{T_R\}}\} \tag{3}$$

This probability $p(s'|s,a)$ will represent our action model and it is calculated *a priori*, depending on the possible action the robot can perform in that state space.

3.3 Sensor Model

Our sensor model take three types of sensations from the image taken by the robot's camera:

Depth. The main target for this observation is measure how far the robot is from the wall when it is orientated to the end of the corridor. For this purpose we detect the number of ceiling lights that the robot perceive. If the number of ceiling lights is high, the robot is far from the end. If this measure is low, the robot is near to the end. In Fig. 3.3 we can see the original image and the image with the ceiling lights and doors detected.

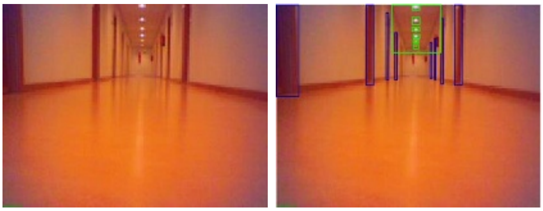


Fig. 2. Image information extraction results. Detecting 6 ceiling lights and 8 doors.

Doors. Using a color filter, the robot is able to count the number of doors it can observe ahead.

Near landmark. This observation give us information about which landmarks are around the robot. We define landmarks as the doors or walls that are situated in the right, left and front side of robot.

Once the data is collected, we apply the equation (2) to correct the belief.

$$Bel_{subsequent}(s) = p(\mathbf{o}|s) \cdot Bel_{previous}(s), \forall s \in S. \tag{4}$$

4 Experiments

We have made several experiments in a corridor of our office environment in a normal daily work. In Fig. 3.3, we can see the corridor that we have used for the experiment and how we have divided it into nodes. Afterwards we have divided the set of nodes into states. This office environment is very symmetric and that is why this scenario entrails much more difficulty for the localization system.

For the experimental results, we have used the error function shown in equation (5), where $state_{high}$ denotes the state with the greatest belief and $state_{current}$ is the robot actual position. The *distance* is measured as the number of steps needed to reach one state from another.

$$\begin{aligned} error = ||prob(state_{high} - prob(state_{current}))) \cdot \\ \cdot distance(state_{high}, state_{current})|| \end{aligned} \tag{5}$$

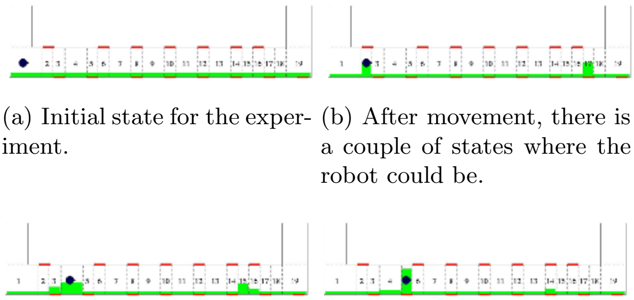
4.1 Ability to Recover of an Action Error

In the first experiment we want to verify if the system is robust enough to cope with action errors. The system must be able to detect when the movement was wrong using its sensors, and recover from this situation.

For this purpose, we have situated the robot in state 15 and we ordered it to go forward along the corridor. The robot knows its initial position, in other words, the probability distribution is concentrated in the state 15.

$$Bel_0(s_{15}) = 1$$

$$Bel_0(s_i) = 0, \forall s_i \in S, s_i \neq s_{15}$$



(a) Initial state for the experiment.

(b) After movement, there is a couple of states where the robot could be.

(c) In the next step, the robot skips the node 3.

(d) In this step the simetry is broken.

Fig. 3. Experiment done in a corridor. The amount of green in each state represent the belief in it.

We commanded the robot to perform the actions secuentially. Some of these actions did not execute correctly. Although in the *movement step* the belief is changed in a wrong way, in the *observation step* the belief is corrected due to the information obtained from the environment in all the experiments.

4.2 Localization Speed

In this experiment the robot does not know its starting position, so the first time the location probability distribution is uniform.

$$Bel_0(s_i) = \frac{1}{|S|}, \forall s_i \in S \tag{6}$$

This experiment was realized with a lot of sensor noise because there were a lot of people walking along the corridor. Despite this difficulty, the robot is able to be localized with a small error in a few movements and can recovery to sensor error quickly, as we see in Fig. 3(a)-3(d).

In Fig. 3(a) the robot starts at node one and the distribution (painted in green) is uniform along all the nodes. For this explanation we will talk about *node* instead of *states*, which is actually what we use in our model, to simplify this explanation. So, a node will be padded in green depending on the belief of the state situated in this node, orientated on the right. When the robot moves forward it reach to the node 2 (Fig. 3(b)) and it takes data from its sensors. With this data the model evolves and the probability is concentrated in state 2 and 17, because these two states have almost the same observation properties. The robot goes forward, but an error occurs and the robot reaches node 4, instead of node 3. This anomaly is observed in the model and it is corrected in the *observation phase*, as we see in Fig. 3(c). In the last movement the robot reaches the node 5 and then the simetry is broken, concentrating the probability in the node 5, as we see in Fig. 3(d).

5 Conclusions

In this article we have presented the preliminary results for the localization of legged AIBO robots in not-engineered environments, using the vision as an active input sensor. We have shown that the robot is able to localize in real time itself even in environments with noise produced by the human activity in a real office. It deal with uncertainly in its action and uses perceived natural landmarks of the environment as the main sensor input.

The data obtained from sensors, mainly the camera, is discriminant enough and let a fast convergence from an initial unknown state, where the belief over the set of states has been distributed uniformly. Also we have shown that the robot can overcome action failures while localizing, and it recovers from them in a efficient way.

The set of observations we have chosen have been descriptive enough to be efficient in the localization process. We think that the way we determine the number of doors and ceiling lights has been the key for the success of the localization system.

We believe that probabilistic navigation techniques hold great promise for getting legged robots reliable enough to operate in real office environments.

References

[1] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

[2] Stefan Enderle, Marcus Ritter, Dieter Fox, Stefan Sablatnög, Gerhard Kraetzschmar, and Günther Palm. Soccer robot localization sporadic visual features. In *IAS-00*, 2000.

[3] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic enviroments. *Journal of Artificial Intelligence Reseaarch*, 1999.

[4] Franck Gechter, Thomas Vincent, and François Charpillet. Robot localization by stochastic vision based device. In *Cybernetics and Informatics (SCI)*, 2001.

[5] Pablo Guerrero and Javier Ruiz del Solar. Auto-localización de un robot móvil aibo mediante el método de monte carlo. Technical report, Instituto de Ingenieros de Chile, 2003.

[6] J.Borenstein, B.Everett, and L.Feng. *Navigating mobile robots: Systems and techniques.* Ltd. Wesley, MA, 1996.

[7] Jana Kosecká and Fayin li. Vision based topological markov localization. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, Barcelona (Spain), April 2004.

[8] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[9] María E. López, Luis Miguel Bergasa, and M.S.Escudero. Visually augmented pomdp for indoor robot navigation. In *Applied Informatics*, pages 183–187, 2003.

[10] Dandapani Radhakrishnan and Illah Nourbakhsh. Topological localization by training a vision-based transition detector. *IROS*, 1999.

[11] Reid Simmons and Sven Koening. Probabilistic navigation in partially observable environments. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, Montreal (Canada), July 1995.

[12] Mohan Sridharan, Gregory Kuhlmann, and Peter Stone. Practical vision-based monte carlo localization on a legged robot. In *IEEE International Conference on Robotics and Automation*, April 2005.

[13] S.Thrun. Robotic mapping: A survey. *Technical Report CMU-CS-02-111*, 2002.

[14] Manuela Veloso, Elly Winner, Scott Lenser, James Bruce, and Tucker Balch. Vision-servoed localization and behavior-based planning for an autonomous quadruped legged robot. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, pages 387–394, Breckenridge, CO, April 2000.

VolksBot - A Flexible Component-Based Mobile Robot System

Thomas Wisspeintner, Walter Nowak, and Ansgar Bredenfeld

Fraunhofer Institute for Autonomous Intelligent Systems (AIS),
D-53754 Sankt Augustin, Germany
{thomas.wisspeintner, walter.nowak, ansgar.bredenfeld}@ais.fraunhofer.de
http://www.ais.fraunhofer.de

Abstract. In this paper we present a component-based framework for rapid prototyping of mobile robots for research, education and application. The VolksBot construction kit addresses the rising demand for reusability in software, electronic hardware and mechanics by offering open and clearly defined interfaces as well as standardized components in all three fields. We show the versatility of the concept by applying it to different domains, particularly RoboCup Middle Size League as well as the Rescue scenario.

1 Introduction

Participating in RoboCup Middle Size League since 1998, we constructed six generations of different mobile robot platforms. Like many other teams we faced several problems related to specialized system development, time-consuming maintenance and high fluctuation of people combined with loss of knowledge. With this experience in mind, we started the VolksBot project [1] in September 2002. The goal of the project is to create a scalable, cost-effective and robust robot construction kit for advanced research experiments, education as well as for effective prototyping of applications.

Prominent examples of already existing robot construction kits used in education are Lego Mindstorms [2], Fischertechnik Mobile Robots [3], Tetrixx[5] or the Cube System[4]. Although aspects of modularity are addressed by these systems, they are limited in complexity since the building blocks are simple and focus on miniaturization. On the other side, several robot platforms of higher complexity are usually specialized for a certain field of application like indoor or outdoor [6][7][8] or do not follow a construction kit approach. An interesting exception is an approach presented in the MoRob project with a focus on educational robotics [9]. In this paper, we present the VolksBot concept by showing its usage in the RoboCup domain.

2 The VolksBot Concept

VolksBot is a flexible and modular mobile robot construction kit. The component-based approach offers a plug-in architecture with open interfaces in mechanics,

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 716–723, 2006.

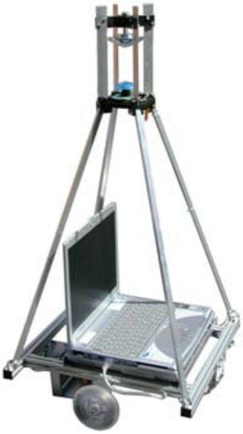


Fig. 1. Example of an indoor VolksBot platform

electronic hardware and software. Quick integration of own modules combined with reuse of existing ones foster application-specific but effective system development. An example of a VolksBot indoor platform is depicted in Figure 1.

2.1 Mechanical Design

To obtain a high grade of flexibility we set up some design criteria for the robot's body construction. Among these are:

- Usage of standardized parts
- Light-weight but rigid construction
- Easy modification
- Quick access and easy exchange of components

Regarding these criteria, we decided to use aluminum machine construction extrusions (x-profile) and proper connectors to build up the robots main frame. Size and shape of the body can be adjusted individually to the needs by simple mechanical processing. All sides of the x-profiles can be used to connect to additional elements. This enables all hardware components to be connected to the main frame. Therefore only geometrical dependencies between the component and the main frame occur, not between the components themselves. Figure 2 shows, how different components, like a battery, a motor-controller and the holonomic drive can be attached to the main frame. The mounting positions of the components are variable.

2.2 Hardware Components

A set of basic hardware components consisting of processing unit, actuators and sensors is described here. A standard notebook or an embedded PC is used as the robot's central control unit.

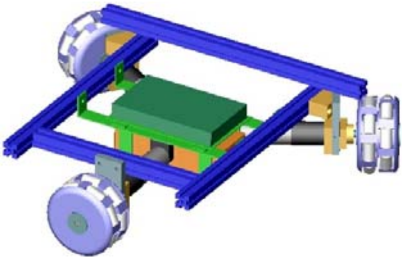


Fig. 2. CAD model of a VolksBot main frame with attached components including holonomic drive

The motor-controller TMC200 is connected via serial interface to the control PC. The controller offers odometric data analysis, thermal motor protection, battery voltage monitoring, PID-control of velocity and current for three DC-motors up to 200W power. The drive units consist of a DC-motor, a scalable planetary gear and a shaft encoder being attached to the supported wheel shaft via a damped coupling. The entire unit is encapsulated in an aluminum block which can be attached at different positions on the main frame.

The catadioptric vision system AISVision includes an IEEE1394 CCD camera and a hyperbolic mirror as shown in Figure 1. Before construction, the system was designed entirely in simulation using ray-tracing software. In an iterative process, all relevant geometry parameters of the system were optimized for the use on a RoboCup Middle Size field. These include height of the mirror with respect to the camera, height of the entire vision-system above the ground, diameter of the mirror, focal distance of the camera and especially the two parameters a and b of the mirrors hyperbolic surface equation

$$\frac{z^2}{a} - \frac{r^2}{b} = 1 \tag{1}$$

with r being the radius and z the dimension along the optical axis. The criteria for this optimization were full visibility of all landmarks from any position in the field, including goals and corner-posts, and a good visibility of the close region. The rendered and the real camera image are depicted in Fig. 3. The optimization can be repeated for any other scenario with the described method.

2.3 Software Framework

Also in software a clear framework concept with well-defined components is being used. The aim is to provide easy access with a low entry level via an intuitive programming interface, similar to systems like Lego Mindstorms and Fischertechnik. At the same time it should be possible to use native computer languages like C++ to implement also sophisticated algorithms. Further demands are:

- Direct and stable hardware access
- Library of existing functionality

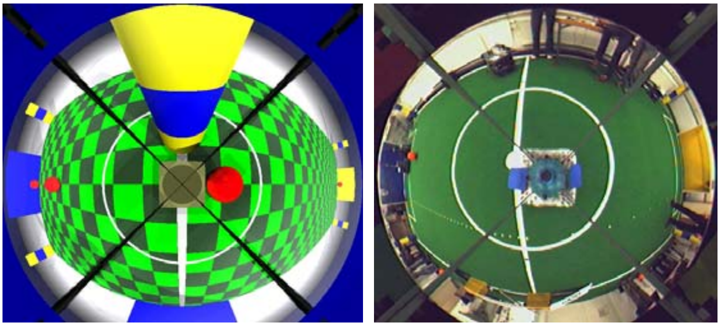


Fig. 3. Comparison between simulated (left) and real (right) camera image

- Easy integration of own algorithms
- High performance and real-time capability
- Structured architecture
- Custom GUI building

ICONNECT [10], a professional software framework by Micro-Epsilon specialized on signal processing and industrial system automation fulfills these demands well. Other similar software packages such as Simulink [11] or Labview [12] are viable, but fail in some aspect like real-time capability or hardware independency. The underlying principle of ICONNECT are signal graphs built up with interconnected modules. A simple example of a signal graph used for robot control is depicted in Figure 4. These graphs provide a well-structured representation for system control and help to bridge the gap between easy access and high scalability by offering different levels of abstraction, ranging from parametrization and visual composition of signal graphs to coding of user-specific modules. The execution of such graphs is handled by ICONNECTs scheduler which provides real-time capability under Microsoft Windows. Different graphs can be executed without recompilation, supporting an iterative development process and rapid prototyping. In Figure 5 the ICONNECT programming environment is depicted, including an example of an easy to build graphical user interface.

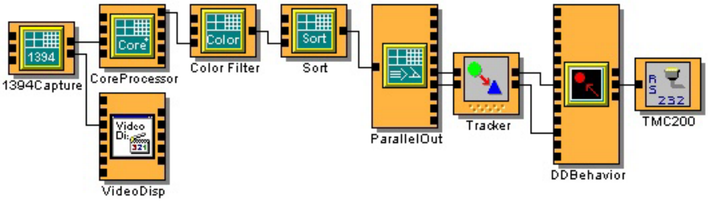


Fig. 4. A simple ICONNECT signal graph for robot control including camera interface, image visualization, image processing, robot behavior and motor control

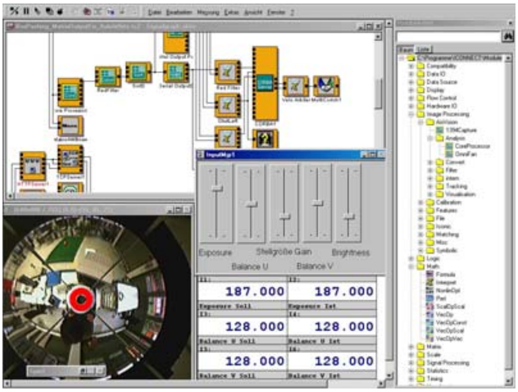


Fig. 5. ICONNECT GUI with signal graph (top), visualization of sensor data (left),a user cockpit (center) and the module library (right)

Several ways to implement new functionality exist in ICONNECT. Besides visual composition of modules and building hierarchies by macro modules, completely new modules can be written in several programming languages ranging from integrated script code over Visual Basic and Perl up to native C++. Such a module is an encapsulated software component, having a well-defined interface specifying the type of in- and outgoing data. Technically it is implemented as a dynamic library which is loaded and instantiated at runtime, independently from other modules. In order to emphasize the notion of an independent, encapsulated and ready-to-use component, each module is associated with a help page, example graph and an individual set of parameters. ICONNECT offers pre-defined modules in a module library, covering areas like signal processing, image processing or hardware IO. We extended the ICONNECT module library by providing robot-specific software modules. In such a way integrated functionality, even when coming from very different backgrounds and sources, can work together in one system in a compatible manner. Thus the software framework acts as a means for standardization.

We already implemented several modules in the domain of robotics, reaching from simulation based on the ODE engine [13] and Matlab over image processing with OpenCV [14], interfaces to CAN-bus and IEEE1394 to RoboCup related behavior. With this method, we constantly add reusable functionality to the existing module-pool.

3 Volksbot in RoboCup Middle Size League

In the beginning of 2004 an international student-team (AIS/BIT) using VolksBot was built up under guidance from AIS Fraunhofer to participate in RoboCup Middle Size League. The modular concept of VolksBot supported this international collaboration and is elaborated in the following.

The main demands on Middle Size League robots are quite different from other scenarios, requiring higher dynamics, superior motion control and real time color vision. To meet the demands set by this special scenario, the team had to introduce only a few additional plug-in components to the existing system. Some of them, like AISVision or our motor-controller TMC200 have even been used by other Middle Size teams for their own robots. Since the goal keeper has a different role in the game than the field players, it motivates a special design with larger main frame and different orientation of the kicking device. Reusing already existing hardware components, this variant could be build up within very short time.

All of these modifications in hardware required only minor software changes due to the component-based structure of ICONNECT. As each hardware component is directly related to one module, only the module itself had to be changed, without affecting the entire system. This also holds true for the behavior architecture itself, where we focused on the use of Dual Dynamics[15], an architecture based on dynamical systems. The DD-Designer [16] tool was extended to directly generate ICONNECT modules, which made the behavior become an easily interchangeable component.

An important aspect of the development process is simulation. A module incorporating physical simulation of robots based on the ODE engine was developed. It has the same interfaces as the hardware, so the development of behaviors can be done without any special treatment, just by replacing the simulator with the corresponding hardware access modules in the graph. This modular concept makes it much easier for new students to get an overview of the system and to become productive, as the number of dependencies between modules is minimized and made explicitly by the module interfaces. This eases also ongoing cooperations with other research groups and universities. After specifying few module interfaces, the spatially distributed groups could work on different modules with minimal integration issues. Similar results regarding successful teamwork were also achieved in RoboCup Real Rescue described in the next section.

4 RoboCup Real Rescue and Outdoor

In 2004, the idea had risen to extend the VolksBot concept, which until then had been purely used for indoor robots, to fit to the needs of rough terrain including the Real Rescue scenario and outdoor. As a consequence new demands have to be set on the system including high payload, mobility, ground clearance and rigidity. A Universal Drive Unit was developed enabling us to build new variants of VolksBot for rough terrain, the new VolksBot RT.

Within only 3 weeks, including development of this unit, manufacturing of parts and final assembly, a six-wheeled version of the VolksBot as depicted in Figure 6 was ready to use. This is due to the fact that we mainly reused existing VolksBot components combined with available standard parts. Only four different parts had to be machined to build up the Universal Drive Unit. Equipped with two 150W DC-motors the robot is able to climb a slope of 43 degrees and

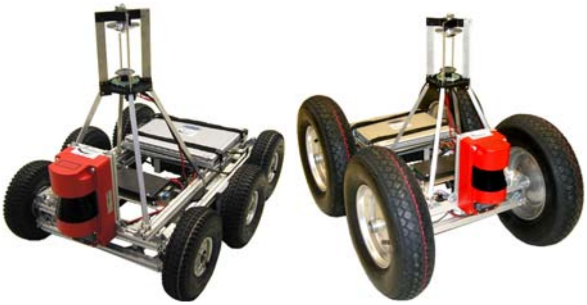


Fig. 6. 6-wheeled and 4-wheeled version of VolksBot RT

has a maximum speed of 1.3m/s. As the motor gears can be exchanged as easily as for the indoor version, the maximum speed can be adjusted according to the demands. With little effort different variants in size and wheel configuration of the VolksBot RT can be built. This 6-wheeled version of VolksBot RT was used as base platform at the RoboCup Rescue Workshop 2004 in Rome. There, within 15 hours of lab-activities, two groups of three and six persons - with no prior experience of the system - worked together to build a functional rescue robot with autonomous behavior which has been demonstrated at the end of the workshop. The task of one group was to build up the entire control system on the robot including signal processing of laser-scanner data, image-processing, compression and WLAN transmission of the AISVision image stream, interfaces for tele-operation, autonomous behavior and motor control.

An obstacle-avoidance method was modified to achieve the desired behaviors like general obstacle avoidance, "left- and right-wall following" or "centering between the aisle". The task of the other group was to build an interface for the operator including visualization of the robots state, camera image and laser-scanner radar. Further on it was required to set the robots state e.g. from manual to autonomous and build an interface to joystick and throttle for proper tele-operation.

The two groups worked together well, first defining the interfaces then testing the results iteratively. In summary the VolksBot concept of rapid system development resulted in a running system within very short time.

5 Conclusion and Future Work

In this paper we have presented our approach to foster re-usability and systematic construction of hardware and software components for mobile robotic systems. Since we concentrate on the concept and not on a single platform, we offer much freedom in the actual design of the robot. This was effectively demonstrated in RoboCup where the VolksBot concept was put into practice in Middle Size League as well as in Real Rescue. It supports research by re-use and exchange of existing components and therefore gives more time and opportunities to concentrate on innovative development. The already existing VolksBot module-pool will be further extended and shared with the growing community. Further on didactic material is being collected, so lecturers can exchange their course material with others to minimize their preparation time. We will continue to build up new variants based on this concept and widen the range of applications, with a special focus on outdoor and navigation.

References

1. AIS Fraunhofer: Volksbot. http://www.volksbot.de
2. Mikhak, B., Berg, R., Martin, F., Resnick, M., Silverman, B.: To Mindstorms and Beyond: Evolution of a Construction Kit for Magical Machines. Robots for Kids: Exploring New Technologies for Learning Experiences 00 (2000)
3. Fischertechnik: http://www.fischertechnik.de
4. Birk, A.: Fast Robot Prototyping with the CubeSystem. Proc. ICRA 04 (2004)
5. Enderle, S., Sablatnog, S., Simon, S., Kraetzschmar, G.: Tetrixx - A Robot Development Kit. Proc. First International Workshop on Edutainment Robots 00 (2000)
6. Evolution Robotics ER1: http://www.evolution.com/er1
7. ActiveMedia: Pioneer. http://www.activrobots.com/ROBOTS/p2dx.html
8. K-Team: Koala robot. http://www.k-team.com/robots/koala/index.html
9. Gerecke, U., Hohmann, P., Wagner, B.: Concepts and Components for Robots in Higher Education. Proc. WAC 04 (2004)
10. Mandl, R., Sick, B.: Messen, Steuern, Regeln mit ICONNECT. (2003), ISBN: 3528058129
11. The MathWorks, Inc.: Simulink Users Guide. (2004)
12. Kalman, CJ.: LabVIEW: a software system for data acquisition, data analysis, and instrument control. J Clin Monit 95 (1995) 51–58
13. Smith, R.: Open Dynamics Engine User Guide. (2005)
14. OpenCV: http://sourceforge.net/projects/opencvlibrary
15. Jaeger, H., Christaller T.: Dual dynamics: Designing behavior systems for autonomous robots. Artificial Life and Robotics (1998) 108–112
16. Bredenfeld, A., Indiveri, G.: Robot Behavior Engineering using DD-Designer. Proc. ICRA 01 (2001)

Author Index