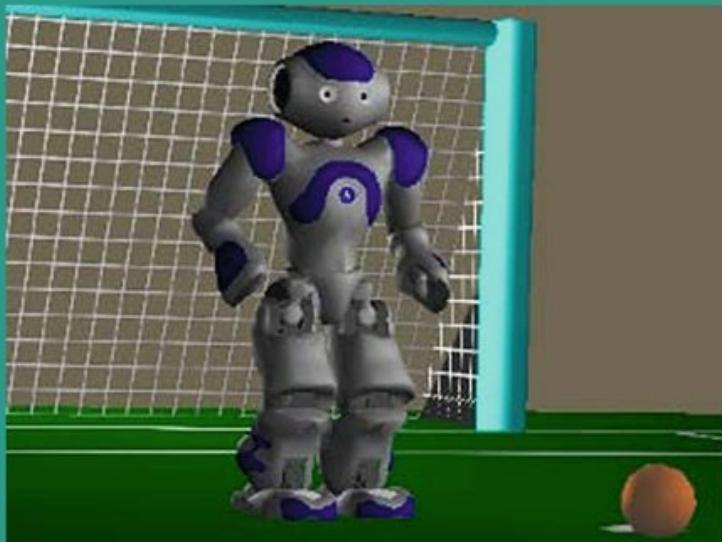


Javier Ruiz-del-Solar
Eric Chown
Paul G. Plöger (Eds.)

LNAI 6556

RoboCup 2010: Robot Soccer World Cup XIV



 Springer



Lecture Notes in Artificial Intelligence 6556
Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Javier Ruiz-del-Solar Eric Chown
Paul G. Plöger (Eds.)

RoboCup 2010: Robot Soccer World Cup XIV

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada

Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Javier Ruiz-del-Solar

Universidad de Chile, Department of Electrical Engineering

and Advanced Mining Technology Center

Av. Tupper 2007, Santiago de Chile, Chile

E-mail: jruizd@ing.uchile.cl

Eric Chown

Bowdoin College, Department of Computer Science

8650 College Station, Brunswick, ME 04011, USA

E-mail: echown@bowdoin.edu

Paul G. Plöger

Hochschule Bonn-Rhein-Sieg, Fachbereich Informatik

Grantham-Allee 20, 53757 Sankt Augustin, Germany

E-mail: paul.ploeger@h-brs.de

ISSN 0302-9743

e-ISSN 1611-3349

ISBN 978-3-642-20216-2

e-ISBN 978-3-642-20217-9

DOI 10.1007/978-3-642-20217-9

Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011923803

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

LNCS Sublibrary: SL 7 – Artificial Intelligence

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

On June 25, 2010 Singapore Polytechnic in the city-state of Singapore played host to the 14th annual RoboCup International Symposium. This year the symposium took place the day after the RoboCup competition was concluded. Participating leagues in the RoboCup include Simulation, Soccer, Rescure, @home and Junior. This year the symposium was organized in two scientific tracks that were complemented by an additional industrial track and a special track focusing on robotics and education.

Through the 14 years a number of research topics have been stable focal points of scientific research while new areas have evolved only more recently. We used these ideas to structure the scientific tracks into four sessions entitled “Simulation and Rescue Robots,” “Robot Perception and Localization,” “Robot Motion and Humanoid Robots” and “Human Robot Interaction and Semantic Scene Analysis.”

The symposium began with a keynote address given by Eric Grimson, who heads the Computer Vision Group of MIT’s Computer Science and Artificial Intelligence Laboratory and is also Chair of the Computer Science Department at MIT. Dr. Grimson’s address was entitled “Computer Vision for Surgery and Disease Analysis” and focused on his groundbreaking work in using computer vision to guide and aid surgery. Our second keynote, entitled “The Three Laws of Robotics and the Coexistence of Human and Robot in Harmony,” was presented by Chong Tow Chong, the founding Provost of the Singapore University of Technology and Design.

This year the symposium had 78 submissions in a double-blind review process, out of which 20 were accepted as full papers, i.e., a 26% acceptance rate, with another 16 accepted as short papers for an overall acceptance rate of 46%. Thanks to the support of the Program Committee we had at least three reviewers per paper, with the symposium Co-chairs making the final decisions in case of conflicting evaluations. The authors Andreas Seekircher, Thomas Röfer and Tim Laue received the best paper award for their contribution on “Entropy-Based Active Vision for a Humanoid Soccer Robot.”

We want to thank the RoboCup Organizing Committee for making the local arrangements, which worked out effectively and smoothly. In particular we want to thank Lau Lee Yee, the Local Chair for the Symposium, and Zhou Changjiu the General Chair for RoboCup 2010. Further, we would like to thank all of the Program Committee members for their hard work and the truly

marvelous job of refereeing they did. Last but not least we want to thank all of the authors for their contributions. The next RoboCup competition will run from July 5 to 11 in Istanbul with the conjoint symposium taking place on July 11.

October 2010

Javier Ruiz-del-Solar
Eric Chown
Paul G. Plöger

Organization

Symposium Chairs

Javier Ruiz-del-Solar

Universidad de Chile, Chile

Eric Chown

Bowdoin College, USA

Paul G. Plöger

Bonn-Rhine-Sieg University of Applied Science,
Germany

Program Committee

C. Acosta

H. Kenn

M. Restelli

H.L. Akin

T. Kimura

A.F. Ribeiro

L. Almeida

A. Kleiner

C. Ribeiro

F. Amigoni

G. Kraetzschmar

R. Rojas

J. Anderson

M. Lagoudakis

J. Ruiz del Solar

J. Baltes

S. Levy

P. Rybski

S. Behnke

P. Lima

V. Santos

A. Birk

G. Lopes

S. Schiffer

A. Bonarini

A. Matsumoto

A. Shahri

A. Bredenfeld

M. Matteucci

S. Shiry

R. Brena

M. Mayer

D.G. Sorrenti

H. Burkhard

E. Menegatti

M. Sridharan

V. Caglioti

T. Mericli

P. Stone

S. Carpin

C. Mericli

J. Strom

R. Cassinis

Z. Mingguo

T. Takahashi

W. Chen

D. Monekosso

Y. Takahashi

E. Chown

Y. Nagasaka

A. Tawfik

P. Costa

T. Nakashima

T. van der Zant

M.B. Dias

D. Nardi

S. Velastin

A. Eguchi

T. Naruse

U. Visser

A. Farinelli

E. Nazemi

O. von Stryk

E. Frontoni

I. Noda

A. Weitzenfeld

A. Ghaderi

T. Nomura

A. Williams

G. Gini

O. Obst

M.A. Williams

G. Grisetti

T. Ohashi

F. Wotawa

T. Hermans

E. Pagello

G. Xie

A. Hofmann

P. Plöger

C. Zhou

G. Indiveri

D. Polani

S. Zickler

L. Iocchi

M. Quinlan

V.A. Ziparo

D. Jahshan

L.P. Reis

M. Jamzad

T. Röfer

Table of Contents

Full Papers

Entropy-Based Active Vision for a Humanoid Soccer Robot	1
<i>Andreas Seekircher, Tim Laue, and Thomas Röfer</i>	
A Realistic Simulation Tool for Testing Face Recognition Systems under Real-World Conditions	13
<i>Mauricio Correa, Javier Ruiz-del-Solar, S. Isao Parra-Tsunekawa, and Rodrigo Verschae</i>	
Thermal Face Recognition Using Local Interest Points and Descriptors for HRI Applications	25
<i>Gabriel Hermosilla, Patricio Loncomilla, and Javier Ruiz-del-Solar</i>	
On Progress in RoboCup: The Simulation League Showcase	36
<i>Thomas Gabel and Martin Riedmiller</i>	
Odometry Correction for Humanoid Robots Using Optical Sensors	48
<i>Stefan Czarnetzki, Maximilian Hegele, and Sören Kerner</i>	
SSL-Humanoid: RoboCup Soccer Using Humanoid Robots under the Global Vision	60
<i>Tadashi Naruse, Yasuhiro Masutani, Noriaki Mitsunaga, Yasunori Nagasaka, Takashi Fujii, Masato Watanabe, Yukiko Nakagawa, and Osamu Naito</i>	
Localization with Non-unique Landmark Observations	72
<i>N. Ergin Özkucur and H. Levent Akin</i>	
MR-Simulator: A Simulator for the Mixed Reality Competition of RoboCup	82
<i>Marco A.C. Simões, Josemar Rodrigues de Souza, Fagner de Assis Moura Pimentel, and Diego Frias</i>	
Learning Footstep Prediction from Motion Capture	97
<i>Andreas Schmitz, Marcell Missura, and Sven Behnke</i>	
Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots	109
<i>Judith Müller, Tim Laue, and Thomas Röfer</i>	
Towards Semantic Scene Analysis with Time-of-Flight Cameras	121
<i>Dirk Holz, Ruwen Schnabel, David Droseschel, Jörg Stückler, and Sven Behnke</i>	

Providing Ground-Truth Data for the Nao Robot Platform	133
<i>Tim Niemüller, Alexander Ferrein, Gerhard Eckel, David Pirro, Patrick Pobregar, Tobias Kellner, Christof Rath, and Gerald Steinbauer</i>	
Optimizing Particle Filter Parameters for Self-localization	145
<i>Armin Burchardt, Tim Läue, and Thomas Röfer</i>	
Improving People Awareness of Service Robots by Semantic Scene Knowledge	157
<i>Jörg Stückler and Sven Behnke</i>	
An Evaluation of Open Source SURF Implementations	169
<i>David Gossow, Peter Decker, and Dietrich Paulus</i>	
A Semantic World Model for Urban Search and Rescue Based on Heterogeneous Sensors	180
<i>Johannes Meyer, Paul Schnitzspan, Stefan Kohlbrecher, Karen Petersen, Mykhaylo Andriluka, Oliver Schwahn, Uwe Klingauf, Stefan Roth, Bernt Schiele, and Oskar von Stryk</i>	
Improving Biped Walk Stability Using Real-Time Corrective Human Feedback	194
<i>Çetin Mericli and Manuela Veloso</i>	
A Review of Shape Memory Alloy Actuators in Robotics	206
<i>Mohammad Mahdi Kheirikhah, Samaneh Rabiee, and Mohammad Ehsan Edalat</i>	
Biologically Inspired Mobile Robot Control Robust to Hardware Failures and Sensor Noise	218
<i>Fabio DallaLibera, Shuhei Ikemoto, Takashi Minato, Hiroshi Ishiguro, Emanuele Menegatti, and Enrico Pagello</i>	
TOPLEAGUE & BUNDESLIGA MANAGER New Generation Online Soccer Games	230
<i>Ubbo Visser</i>	
Human vs. Robotic Soccer: How Far Are They? A Statistical Comparison	242
<i>Pedro Abreu, Israel Costa, Daniel Castelão, Luís Paulo Reis, and Júlio Garganta</i>	
Learning Powerful Kicks on the Aibo ERS-7: The Quest for a Striker ...	254
<i>Matthew Hausknecht and Peter Stone</i>	
Real-Time Active Vision by Entropy Minimization Applied to Localization	266
<i>Stefan Czarnetzki, Sören Kerner, and Michael Kruse</i>	

Multi-agent Behavior Composition through Adaptable Software Architectures and Tangible Interfaces	278
<i>Gabriele Randelli, Luca Marchetti, Francesco Antonio Marino, and Luca Iocchi</i>	
A Novel Real-Time Local Visual Feature for Omnidirectional Vision Based on FAST and LBP	291
<i>Huimin Lu, Hui Zhang, and Zhiqiang Zheng</i>	
Mixed 2D/3D Perception for Autonomous Robots in Unstructured Environments	303
<i>Johannes Pellenz, Frank Neuhaus, Denis Dillenberger, David Gossow, and Dietrich Paulus</i>	
Hierarchical Multi-robot Coordination	314
<i>Viktor Seib, David Gossow, Sebastian Vetter, and Dietrich Paulus</i>	
Biped Walking Using Coronal and Sagittal Movements Based on Truncated Fourier Series	324
<i>Nima Shafii, Luís Paulo Reis, and Nuno Lau</i>	
Realistic Simulation of Laser Range Finder Behavior in a Smoky Environment	336
<i>Okke Formsma, Nick Dijkshoorn, Sander van Noort, and Arnoud Visser</i>	
Cooperative Localization Based on Visually Shared Objects	350
<i>Pedro U. Lima, Pedro Santos, Ricardo Oliveira, Aamir Ahmad, and João Santos</i>	
Parameter Optimization of a Signal-Based Omni-Directional Biped Locomotion Using Evolutionary Strategies	362
<i>Barış Gökçe and H. Levent Akin</i>	
Designing Effective Humanoid Soccer Goalies	374
<i>Marcell Missura, Tobias Wilken, and Sven Behnke</i>	
A Supporter Behavior for Soccer Playing Humanoid Robots	386
<i>Karen Petersen, Georg Stoll, and Oskar von Stryk</i>	
Utilizing the Structure of Field Lines for Efficient Soccer Robot Localization	397
<i>Hannes Schulz, Weichao Liu, Jörg Stückler, and Sven Behnke</i>	
Robot Detection with a Cascade of Boosted Classifiers Based on Haar-Like Features	409
<i>F. Serhan Daniş, Tekin Meriçli, Çetin Meriçli, and H. Levent Akin</i>	
LearnPNP: A Tool for Learning Agent Behaviors	418
<i>Matteo Leonetti and Luca Iocchi</i>	
Author Index	431

Entropy-Based Active Vision for a Humanoid Soccer Robot

Andreas Seekircher¹, Tim Laue², and Thomas Röfer²

¹ University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146 USA
aseek@mail.cs.miami.edu

² Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de

Abstract. In this paper, we show how the estimation of a robot’s world model can be improved by actively sensing the environment through considering the current world state estimate through minimizing the entropy of an underlying particle distribution. Being originally computationally expensive, this approach is optimized to become executable in real-time on a robot with limited resources. We demonstrate the approach on a humanoid robot, performing self-localization and ball tracking on a RoboCup soccer field.

1 Introduction

Many robot localizations use passive vision systems. Path planning and navigation tasks are often based on a single given robot pose. While acting in an environment to achieve a goal, the measurements used for localization are made independently from the current belief in the state estimation. However, there are several approaches showing that an active vision can be used to reduce uncertainties in localizations. An active vision system controls the robot’s actuators or sets sensor parameters to receive measurements that provide as much information as possible. Active localization is often divided into active navigation and active sensing. In active navigation even the target position of the robot is chosen to minimize uncertainties. In this paper an active sensing system is described that controls the head of a humanoid robot and thereby optimizes the state estimation. The focus of the work presented here is on computational efficiency. i. e. to spend only a few milliseconds of the computation time on an embedded system to implement the active sensing, leaving enough computational resources for the other modules running on the robot that, in our case, realize playing soccer.

This paper is organized as follows: The humanoid robot platform and its environment are shortly described in Sect. 2. The active vision approach is presented in Sect. 3, enhancements regarding its computational efficiency are presented in Sect. 4. Section 5 presents the experiments that have been carried out together with the results achieved.

1.1 Related Work

There are many approaches using mutual information. Fox, Burgard, and Thrun realize active Markov localization by an entropy-minimizing action selection [1,3]. Denzler and Brown use a similar method for choosing camera parameters (pan, tilt, and zoom) for object recognition [2]. Vidal-Calleja et al. use an active localization for SLAM [16]. However, active vision systems have also been used in other domains, not only for localization. For instance, Schill et al. use active sensing for the exploration of spatial scenes [12]. Except for the latter, all approaches are using active navigation and active sensing, because the localization controls all robot actions. However, in some cases the robot has to perform a global task and the localization can only be improved by active sensing. Seara and Schmidt use an entropy-based method for gaze control to optimize simultaneous localization and obstacle detection [13]. A different approach by Zhou and Sakane uses Bayesian networks to represent relationships between the environment, control actions, and state estimates [17].

There are also task-oriented approaches to learn camera control strategies for robots playing soccer. In [7] the camera control for shooting goals is learned by reinforcement learning using the success rate for rewards. The camera control of a goalkeeper is learned in [5].

2 Robot Platform and Environment

Our experimental platform is the RoboCup edition of the Nao robot [4] made by Aldebaran Robotics as shown in Fig. 1. The robot is equipped with 21 degrees of freedom, a 500 MHz processor, and a camera as main sensor¹. In addition, it provides measurements of joint angles which are combined with accelerometer and gyroscope data for body posture estimation. Image processing and self-localization are performed at the camera’s frame rate, i. e. 30 Hz.

The experimental environment is a RoboCup Standard Platform League field. The field size is $7.4m \times 5.4m$. The only unique features are two colored goals and one center circle. In addition, the field contains several different lines which are non-unique.

The robot runs the software framework of the RoboCup team *B-Human* [11] in the same configuration as it is used during real competitions, e. g. during the team’s win of the RoboCup German Open 2009 and the RoboCup 2009. The self-localization component is an improved version of the one described in [8] that is based on [10], additionally including the *Augmented MCL* resampling approach of [6]. The robot’s vision software is able to perceive a variety of features on the field. In addition to the aforementioned ones, also field line crossings, a ball, and single non-unique goal elements can be extracted from images and be used by the self-localization component.

¹ In fact, the robot has two cameras but only the lower one has been used for our purposes.

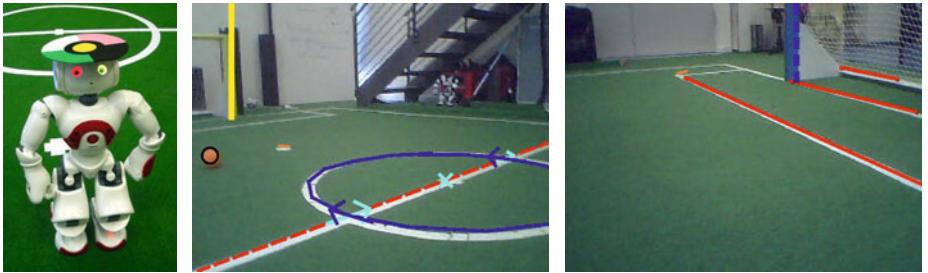


Fig. 1. Left: Nao RoboCup robot used for the experiments. A colored pattern for global tracking has been attached to its head. Middle and right: the robot’s perceptions. Images taken by the robot’s camera, perceived objects – a yellow right goal post, a blue goal post (side unknown), the center circle, the ball, and several lines – are labeled.

3 Active Vision

Self-localization and ball tracking is based on the perceptions retrieved from camera images. The result of the state estimation is highly dependent on the number and type of the visible features. The perceptions extracted from camera images give information about the state of the robot and the ball. By calculating particle weightings from observations, the uncertainty in the belief distribution is reduced. In many cases moving the camera in a fixed pattern does not provide as much information as possible. A camera control system choosing the pointing direction that gives the highest information gain should improve the state estimation.

3.1 Information and Uncertainty

The uncertainty of a belief b about a state x is given by the Shannon entropy:

$$H_b(x) = - \int b(x) \log b(x) dx \quad (1)$$

The information gain (the mutual information) of an action a is equal to the decrease in uncertainty and it can be calculated by the difference between the entropies in cycles:

$$I(x, a) = H_b(x) - H_b(x'|x, a) \quad (2)$$

In our system, state estimation is realized using particle filters. Therefore the belief is represented as a set of samples. These samples only approximate the actual probability density function. The entropy cannot be calculated directly and has to be estimated. There are many approaches for entropy estimation, for example maximum-likelihood or m -spacing estimators. One of these methods is to create partitions in the state space and to calculate the discrete entropy over the sum of particle weightings (probability mass) in each partition (e. g. as

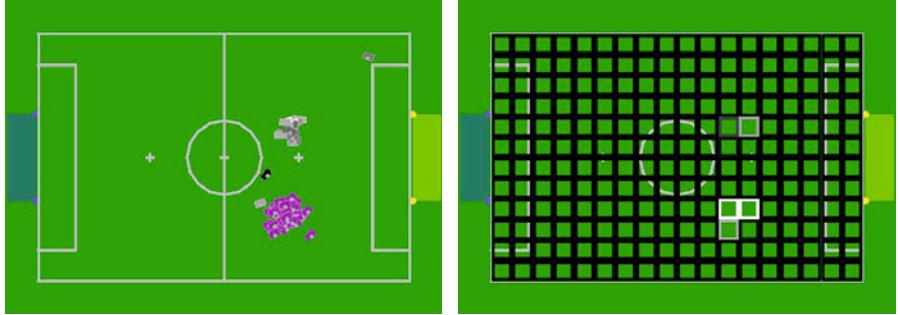


Fig. 2. An example for a belief with samples and the grid used for entropy estimation. The grid dimensions are (18, 12, 16) on the x , y , and rotational axes (the latter is not shown).

in [14]). Similar to that method, simple histogram-based entropy estimation is used in this paper. The belief is approximated by using a grid on the x , y , and rotation axes (cf. Fig. 2). The discrete entropy over all cells can be calculated very efficiently, because the number of cells with a probability greater than 0 is limited by the number of samples. Therefore most cells have no probability and can be ignored. The entropy estimation over a constant grid gives comparatively poor results with a high bias. Nevertheless it is used here because of the limited computational resources.

3.2 Entropy-Based Camera Control

The camera control has to choose the direction with the highest mutual information expected. Thus the entropy expected after executing a given action a in state x is needed for an entropy-minimizing action selection. Equation 3 gives the entropy expected in an probabilistic environment (e.g. used for active localization in [1]). A set of observations is denoted as z . So $p(z|x)$ gives the probability of perceiving z in the state x . The probability of reaching the state x' by executing the action a in state x is written as $p(x'|a, x)$.

$$\begin{aligned} H_b(x'|a) &\approx E_z[H_b(x'|z, a)] \\ &= \int \int \int H_b(x'|z, a)p(z|x')p(x'|a, x)b(x)dzdx'dx \end{aligned} \quad (3)$$

An action selection based on minimizing the entropy only for the immediate next step only produces a greedy exploration. The camera movement created in this way is not the optimal strategy. Due to local maxima, a sequence of other actions could be necessary to produce a higher information gain. However, planning the camera control for more than a single step will result in much higher computational costs due to exponential complexity. Therefore, a camera control system optimizing the information gain only for a single step can be calculated more frequently. Thus, such a control system is more reactive and

will possibly perform better in a dynamic environment. The camera control has to react to unexpected changes in the belief, for example when the robot's position is changed by collisions.

The policy for controlling the camera is given by equation 4. The action with the lowest entropy expected and lowest costs is selected. The costs for each action a in the state x is given by $r(x, a)$ as negative values. The weighting between costs and the entropy expected is given by α .

$$\begin{aligned}\pi(b) &= \underset{a}{\operatorname{argmax}} \alpha(H_p(x) - E_z[H_b(x'|z, a)]) + \int r(x, a)b(x)dx \\ &= \underset{a}{\operatorname{argmax}} \int r(x, a)b(x)dx - \alpha(E_z[H_b(x'|z, a)])\end{aligned}\quad (4)$$

An implementation of this strategy is the Monte Carlo exploration in algorithm 1:

Algorithm 1. Monte carlo exploration (described in [15])

```

1: Monte_Carlo_Exploration( $b$ ):
2: set  $p_a = 0$  for all actions  $a$ 
3: for  $i = 1$  to  $N$  do
4:   sample  $x \sim b(x)$ 
5:   for all control actions  $a$  do
6:     sample  $x' \sim p(x'|a, x)$ 
7:     sample  $z \sim p(z|x')$ 
8:      $b' = \text{Bayes\_filter}(b, z, a)$ 
9:      $p_a = p_a + r(x, a) - \alpha H_{b'}(x')$ 
10:  end for
11: end for
12: return  $\underset{a}{\operatorname{argmax}} p_a$ 
```

By sampling from the various probability distributions, the method approximates the expected entropy for $N \rightarrow \infty$. As the belief is already represented as samples, line 4 only means to choose one sample of the particle filter. In fact a state x consists of the belief and the head joint angles given by sensor readings. Drawing samples from $p(x'|a, x)$ in line 6 can be simplified by the assumption that the state change from x to x' is deterministic. The actions available do only affect the head movement. Many approaches for active localization have to consider uncertainty and the probability of collisions when determining the next state x' for an action a . However by limiting the actions to head movements, the next state x' consist of the same belief for the poses only with changed head joint angles. The state x' is given by the action a .

The last probability function needed for calculating the expected entropy is $p(z|x')$ in line 7. There, the expected observations for a given state x' are created to be used as measurements in the state estimation in line 8. In this paper, all particle weightings are calculated using the expected observation z . These weightings produce an information gain by changing the entropy calculated in line 9.

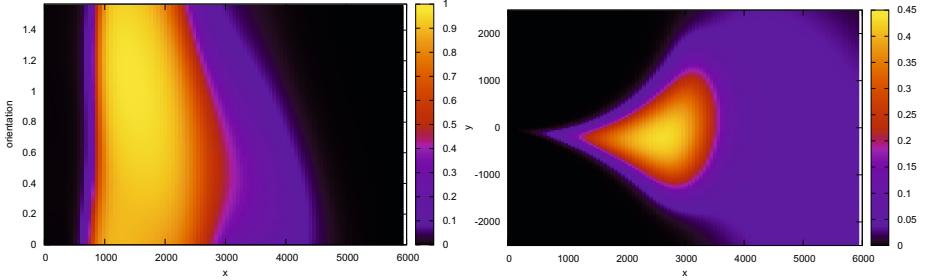


Fig. 3. Sensor model for lines and the right goal post. The color denotes the probability to recognize a line depending on its orientation and distance to the robot (left) and the right goalpost depending on its relative position to the robot (right).

3.3 Expected Observations

The probability function $p(z|x')$ needed for the Monte Carlo Exploration gives the probability for a measurement z in a given state x' . It is necessary to predict the measurements for different actions to calculate utility values for action selection. The raw measurement used for state estimation is the camera image. However the vision modules of the RoboCup team *B-Human* extract features from the camera image. Therefore, the self-localization and the ball tracking are feature-based localizations using the features recognized as observations. These features are field lines, goal posts, the middle circle, corners, and the ball. An observation z consists of a set of visible features. Predicting an observation means to create a set of features that could be perceived in state x' . Due to the known environment, the expected observations can be created from the known field model by iterating through all features and checking which features can be seen. The decision whether a feature is visible or not depends on the feature's relative position (x, y) and orientation ϕ to the camera and the current camera tilt angle θ :

$$(x, y, \phi, \theta) \rightarrow [0, 1] \quad (5)$$

The simplest solution would be to check whether an object lies inside the image and is not too far away. But each feature can have special properties that have an influence to the probability of actually being perceived. For instance, horizontal field lines cannot be seen in as large distances as vertical lines. In general, a better solution for solving this problem is to create sensor models by learning the function in equation 5 with examples created from real perceptions. In this approach, the function was approximated with a feed-forward neural network with 20 hidden nodes and by using backpropagation to minimize the output error. With these sensor models it is possible to decide where certain features are visible. For example, Fig. 3a shows the probability of perceiving a field line depending on the distance and the orientation. The maximum distance varies for different orientations as expected. As another example, Fig. 3b shows the probability for seeing a goal post depending on its x and y coordinates. Here,

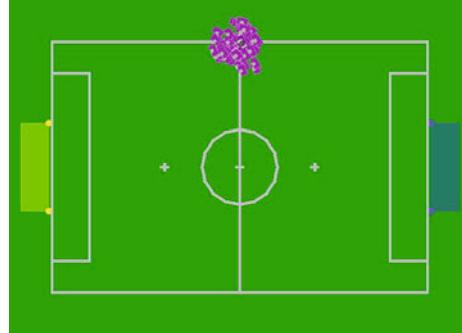


Fig. 4. An example belief. The grey drawing in the middle is the ground truth robot pose.

the orientation is always set to the direction to the right goal post. There must be a small part of the upper bar of the goal visible to detect the side of the goal post. As a result, the colored area is slightly shifted to the right side. Now the expected observations z for a given state x' including any pan/tilt-angles can be created using these sensor models. For the sample set in the self-localization in Fig. 4, the actual mutual information and the expected information gain for the belief and head joint angles are compared in Fig. 5. In that example, the camera was moved from pan angle $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ and back with a constant tilt angle of 20° . The robot is already localized well, so the prediction of the information gain is similar to the actual information gain produced by the perceptions.

The expected information gain for a given belief can be calculated for any camera direction by calculating the average information gain caused by the expected observations for all sample poses. The expected information gain for all reachable camera angles is shown in Fig. 6. The best information is provided by the goals in the upper part. The large red area on the left side is the direction to the center circle.

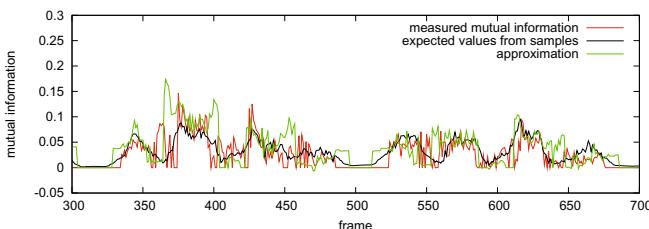


Fig. 5. The mutual information (red) and the expected information gain (black). The values are very similar, because most samples are near the correct robot pose. The actual information gain is only affected by noise. The approximated information gain (green) as described in Sect. 4.

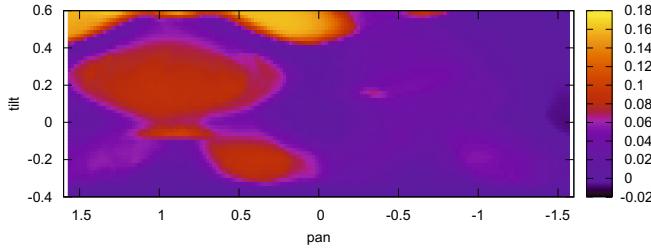


Fig. 6. The entropy expected for all reachable camera angles in one frame from the example in Fig. 4

These values are calculated only for a fixed number of head joint angles defined as actions. The action selection chooses one of these directions as a target position for the head control. The cost of an action is given by the angular distance between the target camera position and the current camera position.

3.4 Feedback

The calculation of the expected information gain is based only on the expected observation for the current belief. However, the expected observations can be wrong due to occlusions or bad sample positions. In Fig. 7, the robot was moved in frame 700. In the following frames the expected values do not match the measured values, so the measured values at an action's position are stored as feedback values and used for some seconds to adjust the expected information gain for this action. This way the action selection does not choose an action with a high expected value, where actually no information has been retrieved a short time before.

3.5 Ball

The calculation of the entropy expected in the localization of the ball is realized with the same method. Because there is only a single type of perception, i. e. a visible ball, an expected observation for a given sample is simply a ball percept

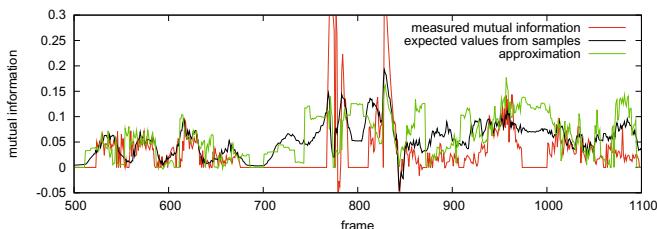


Fig. 7. At the beginning the same data as in Fig. 5, but the robot has been moved in frame 700

at this position. The weightings produced by an observation at a given sample position can directly be obtained by the sample positions and the differences between their distances and angles to the robot. Thus the entropy expected can be calculated directly from the particle positions.

The entropy-based camera control has to optimize the results of the self-localization and ball tracking simultaneously. This is realized by minimizing the sum of both entropies, weighted by a factor for the importance of the ball.

4 Optimizations

The calculation of the action utility values used for action selection described so far is too expensive to be performed for each camera image. The most complex part is to create the expected observations and to calculate the hypothetical particle weightings in the self-localization. In the experiments presented in the next section, these weightings are stored in a look-up-table. This table stores the weightings for a given sample pose produced by the observation at any hypothetical robot pose and tilt angle. By combining the pan angle with the robot rotation, there are seven dimensions left (two poses and the tilt angle). Although the resolution used for the look-up-table is quite low due to the number of dimensions, the approximated entropy values can still be used for action selection (see Fig. 5). There is a high error in the weighting, but the calculation of the expected entropy for an action has been reduced to reading values from the weightings table and calculating the entropy of these values. The calculation of the entropy has been optimized by using fixed-point arithmetics and an approximation of the logarithm using a table for the logarithm to the base 2.

The full calculation of the entropy expected for only one robot pose hypothesis and a given camera direction needs an average time of 41 ms on the Nao. The largest part of this time is needed to create the expected observations (38.5 ms). By using the weights table and the optimized entropy calculation this time has been reduced to less than 0.07 ms.

In the experiments in the following section the active head control uses 44 different actions. Considering that there are 44 actions and 100 particle poses that are possible robot poses, there are 4400 expected entropies to be calculated in every cycle ($4400 \times 0.07 \text{ ms} = 308 \text{ ms}$). Consequently the number of robot pose hypotheses has to be decreased significantly. In practice, samples rarely distribute over the state space but cluster at a few positions, as, e.g., shown in Fig. 2 where the probability distribution has only two major modes. Through extracting the distribution's n_m most significant modes from n_s samples and using them as robot hypotheses for entropy calculation, the time necessary for computing can be reduced by a factor of $\frac{n_s}{n_m}$. Of course, this assumes that the mode extraction is computationally comparatively inexpensive. Thus this task is realized by the *Particle History Clustering* algorithm [9] which robustly provides good results in linear time.

The calculation of a utility value in the ball localization needs 0.015 ms on average. In most cases the belief for the ball is a unimodal distribution. Therefore

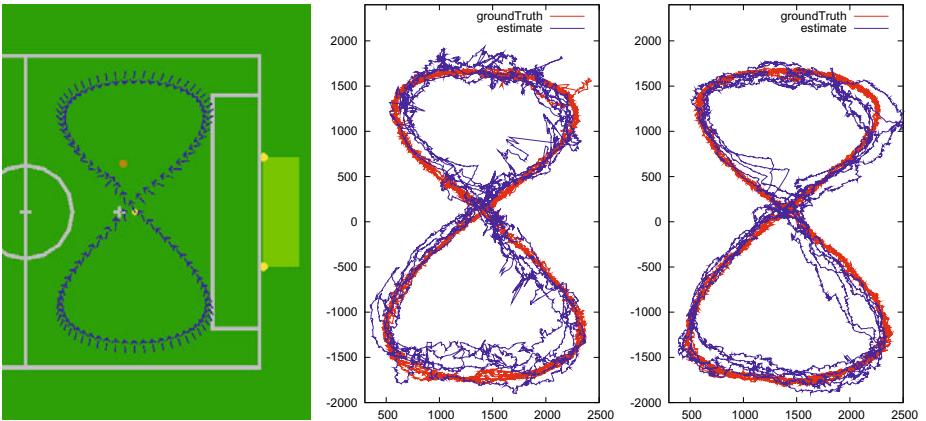


Fig. 8. Left: path used for the experiments. Middle: walked path and the estimated robot position using the passive head control. Right: same for entropy-based head control.

the hypotheses positions used are reduced only to the average sample position, similar to the cluster positions in the self-localization.

Considering an average number of 1.5 clusters in the self-localization and 44 different actions, all utility values are calculated in 5.28 ms. Thus enough processing time is left for the other modules that have to run on a soccer-playing robot (at 30 Hz). All these timings have been measured on a Nao with the operating system and a separate joint control thread running in parallel.

5 Experimental Results

In the experiment, the new active head control is compared to the former head control, used by the striker of the RoboCup team *B-Human*. The former head control points the camera at the ball, at a goal, and at some look-around positions on the left and the right side in regular intervals. The directions to the ball and the goal are based on the current state estimates.

Figure 8 shows the path the robot walks along several times to measure the average errors in the state estimation. The motion control uses the ground truth pose to walk along the path independently from self-localization. As source for ground truth data, a global tracking system has been used. For this purpose, a unique marker has been fixed on the robot's head (cf. Fig. 1) and been tracked by a camera hanging above the field, the software for this purpose is the standard vision system of the RoboCup Small Size League [18]. The system provides the position as well as the rotation (which is fused with the robot's head rotation) of the robot on the field. The self-localization module, described in Sect. 2, has been configured to use 100 samples.

The robot's ground truth position and the estimated state during the experiments are shown in Fig. 8 for the old head control and for the active method.

Table 1. Errors in self-localization and ball tracking

Head control	Self-localization			Ball tracking		
	avg. in mm	stdv. in mm	avg. in rad	avg. in mm	stdv. in mm	seen
passive	163.915	94.8412	0.144	483.547	782.94	37.6%
active	130.062	79.1946	0.105	271.419	360.043	51%

The robot walked along the path 5 times for each experiment. The overall errors in the self-localization and ball tracking are summarized in table 1. The ball importance was quite high in this experiment. By decreasing this value, the self-localization gets better, but the error in ball-tracking will increase again.

6 Conclusions and Future Work

In this paper, we present an approach for an active head control. The method is based on a commonly used entropy-minimizing action selection, but the computationally expensive operations have been executed in a preprocessing step. This allows applying active head control on a robot with low computing power. Thus the action selection takes only a few milliseconds. The error in state estimation in the RoboCup scenario used for the experiments was reduced by 20% in self-localization and by 44% in ball tracking. So both competing state estimations are significantly improved.

A better approximation of the weights with another data structure could improve the results. But nevertheless the computational costs should not be increased too much and an efficient access on the stored weightings is essential for this method to become applied on robots such as the Nao.

Acknowledgements

The authors would like to thank Armin Burchardt for his work at the ground truth system and all members of the team B-Human for providing the software base for this work.

References

- Burgard, W., Fox, D., Thrun, S.: Active mobile robot localization by entropy minimization. In: Proc. of the Second Euromicro Workshop on Advanced Mobile Robotics, pp. 155–162 (1997)
- Denzler, J., Brown, C.: Optimal selection of camera parameters for state estimation of static systems: An information theoretic approach. Tech. rep., University of Rochester (2000)
- Fox, D., Burgard, W., Thrun, S.: Active Markov localization for mobile robots. *Robotics and Autonomous Systems* 25, 195–207 (1998)

4. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: The nao humanoid: a combination of performance and affordability. CoRR abs/0807.3223 (2008)
5. Guerrero, P., Ruiz-del-Solar, J., Romero, M.: Explicitly task oriented probabilistic active vision for a mobile robot. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 85–96. Springer, Heidelberg (2009)
6. Gutmann, J.S., Fox, D.: An experimental comparison of localization methods continued. In: Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002), Lausanne, Switzerland, vol. 1, pp. 454–459 (2002)
7. Kwok, C., Fox, D.: Reinforcement learning for sensing strategies. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), Sendai, Japan (2004)
8. Laue, T., Röfer, T.: Particle filter-based state estimation in a competitive and uncertain environment. In: Proceedings of the 6th International Workshop on Embedded Systems, VAMK, University of Applied Sciences, Vaasa, Finland (2007)
9. Laue, T., Röfer, T.: Pose extraction from sample sets in robot self-localization - a comparison and a novel approach. In: Petrović, I., Lilienthal, A.J. (eds.) Proceedings of the 4th European Conference on Mobile Robots - ECMR 2009, Mlini/Dubrovnik, Croatia, pp. 283–288 (2009)
10. Röfer, T., Laue, T., Thomas, D.: Particle-filter-based self-localization using landmarks and directed lines. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 608–615. Springer, Heidelberg (2006)
11. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-human team report and code release 2009 (2009), http://www.b-human.de/download.php?file=coderelease09_doc
12. Schill, K., Zetsche, C., Parakrama, T.: A hybrid architecture for the sensorimotor exploration of spatial scenes. In: Bloch, I., Petrosino, A., Tettamanzi, A.G.B. (eds.) WILF 2005. LNCS (LNAI), vol. 3849, pp. 319–325. Springer, Heidelberg (2006)
13. Seara, J.F., Schmidt, G.: Intelligent gaze control for vision-guided humanoid walking: methodological aspects. *Robotics and Autonomous Systems* 48(4), 231–248 (2004)
14. Stowell, D., Plumley, M.: Fast multidimensional entropy estimation by k-d partitioning. *Signal Processing Letters* 16, 1930–1936 (2009)
15. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
16. Vidal-Calleja, T., Davison, A.J., Andrade-Cetto, J., Murray, D.W.: Active control for single camera slam. In: ICRA, pp. 1930–1936. IEEE, Los Alamitos (2006)
17. Zhou, H., Sakane, S.: Sensor planning for mobile robot localization - a hierarchical approach using a bayesian network and a particle filter. *IEEE Transactions on Robotics* 24(2), 481–487 (2008)
18. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS(LNAI), vol. 5949, pp. 425–436. Springer, Heidelberg (2010)

A Realistic Simulation Tool for Testing Face Recognition Systems under Real-World Conditions^{*}

Mauricio Correa, Javier Ruiz-del-Solar,
S. Isao Parra-Tsunekawa, and Rodrigo Verschae

Department of Electrical Engineering, Universidad de Chile
Advanced Mining Technology Center, Universidad de Chile

Abstract. In this article, a tool for testing face recognition systems under uncontrolled conditions is proposed. The key elements of this tool are a simulator and real face and background images taken under real-world conditions with different acquisition angles. Inside the simulated environment, an observing agent, the one with the ability to recognize faces, can navigate and observe the real face images, at different distances, angles and with indoor or outdoor illumination. During the face recognition process, the agent can actively change its viewpoint and relative distance to the faces in order to improve the recognition results. The simulation tool provides all functionalities to the agent (navigation, positioning, face's image composing under different angles, etc.), except the ones related with the recognition of faces. This tool could be of high interest for HRI applications related with the visual recognition of humans, as the ones included in the RoboCup @Home league. It allows comparing and quantifying the face recognition capabilities of service robots under exactly equal working conditions. It could be a complement to existing tests in the RoboCup @Home league. The applicability of the proposed tool is validated in the comparison of three state of the art face recognition methods.

Keywords: Face Recognition, Face Recognition Benchmarks, Evaluation Methodologies, RoboCup @Home.

1 Introduction

Face recognition in controlled environments is a relative mature application field (see recent surveys in [1][2][3][4]). However, face recognition in uncontrolled environments is still an open problem [9][10]. Recent journal's special issues [6], workshops [7], and databases [8] are devoted to this topic. Main factors that still disturb largely the face recognition process in uncontrolled environments are [10][11]: (i) variable illumination conditions, especially outdoor illumination, (ii) out-of-plane pose variations, and (iii) facial expression variations. The use of more complex sensors (thermal-, high-resolution-, and 3D- cameras), 3D face models, illumination models, and sets of images of each person that cover various face variations are some of the approaches being used to deal with the mentioned drawbacks, in different application domains [10][11].

* This research was partially funded by FONDECYT under Project Number 1090250.

A very important component in the development of face recognition methodologies is the availability of suitable databases, benchmarks, and evaluation methodologies. For instance, the very well known FERET database [13], one of the most employed face databases that also includes a testing protocol, has been very important in the development of face recognition algorithms for controlled environments in the last years. Some relative new databases such as the LFW (Labeled Faces in the Wild) [8] and FRGC (Face Recognition Grand Challenge) [14][12], among others, intend to provide real-world testing conditions. In applications such as HRI (Human Robot Interaction) and surveillance the use of spatiotemporal context or active vision mechanisms in the face recognition process¹ can increase largely the performance of the systems. However, face recognition approaches that include these dynamic mechanisms cannot be validated properly using current face databases (see database examples in [5]). Even the use of video face databases does not allow testing the use of those ideas. For instance, in a recorded video it is not possible to change actively the observer's viewpoint. The use of a simulator could allow accomplishing this (viewpoint changes), however, a simulator is not able to generate faces and backgrounds that looks real/natural enough.

Nevertheless, the combined used of a simulation tool with real face and background images taken under real-world conditions, could allow to accomplish the goal of providing a tool for testing face recognition systems under uncontrolled conditions. In this case, more than providing a database and a testing procedure, the idea would be to supply a testing environment that provides a face database, dynamic image's acquisition conditions, active vision mechanisms, and an evaluation methodology. The main goal of this paper is to provide such a testing tool. This tool provides a simulated environment with persons located at different positions and orientations. The face images are previously acquired under different pitch and yaw angles², in indoor and outdoor variable lighting conditions. Inside this environment, an observing agent, the one with the ability to recognize faces, can navigate and observe the real face images (with real background information), at different distances, angles (yaw, pitch, and roll) and with indoor or outdoor illumination. During the recognition process the agent can actively change its viewpoint to improve the face recognition results. The simulation tool provides all functionalities to the agent, except the ones related with the recognition of the faces.

This testing tool could be of high interest for HRI applications related with the visual recognition of humans, as the ones included in the RoboCup @Home league. It allows comparing and quantifying the face recognition capabilities of service robots under exactly equal working conditions. In fact, the use of this testing tool could complement some of the real tests that are in use in the RoboCup @Home league.

This article is organized as follows. In section 2, related work in face databases and evaluation methodologies is outlined. In section 3, the proposed testing tool is described. Results of the applicability of the testing tool in the comparison of three state of the art face recognition methods are presented in section 4. Finally, some conclusions and projections of this work are presented in section 5.

¹ In this work we consider the face recognition process as the one composed by the face detection, face alignment, and face recognition stages.

² In-plane rotations can be generated by software (simulator).

2 Related Work

The availability of standard databases, benchmarks, and evaluation methodologies is crucial for the appropriate comparison of algorithms. There is a large amount of face databases and associated evaluation methodologies that consider different number of persons, camera sensors, and image acquisition conditions, and that are suited to test different aspects of the face recognition problem such us illumination invariance, aging, expression invariance, etc. Basic information about face databases can be found in [5][15].

The FERET database [13] and its associated evaluation methodology is a standard choice for evaluating face recognition algorithms under controlled conditions. Other popular databases used with the same purpose are Yale Face Database [16] and BioID [17]. Other database such the AR Face Database [18] and the University of Notre Dame Biometrics Database [19] include faces with different facial expressions, illumination conditions, and occlusions. From our point of view, all of them are far from considering real-world conditions.

The Yale Face Database B [20] and PIE [21] are the most utilized databases to test the performance of algorithms under variable illumination conditions. Yale Face contains 5,760 single light source images of 10 subjects, each seen under 576 viewing conditions (9 poses x 64 illumination conditions). For every subject in a particular pose, an image with ambient (background) illumination was also captured. PIE is a database containing 41,368 images of 68 people, each person under 13 different poses, 43 different illumination conditions, and with 4 different expressions. Both databases consider only indoor illumination.

The LFW database [8] consists of 13,233 images faces of 5,749 different persons, obtained from news images by means of a face detector. There are no eyes/fiducial point annotations; the faces were just aligned using the output of the face detector. The images of the LFW database have a very large degree of variability in the face's pose, expression, age, race, and background. However, due to LFW images are obtained from news, which in general are taken by professional photographers, they are obtained under good illumination conditions, and mostly in indoors.

FRGC ver2.0 database [12] consists of 50,000 face images divided into training and validation partitions. The validation partition consists of data from 4,003 subject sessions. A subject session consists of controlled and uncontrolled images. The uncontrolled images were taken in varying illumination conditions in indoors and outdoors. Each set of uncontrolled images contains two expressions, smiling and neutral.

3 Proposed Testing Tool

The proposed testing tool allows that an observing agent can navigate inside a virtual scenario, and observe a set of N persons. The faces of each of these persons are previously scanned under different yaw and pitch angles, and under different indoor and outdoor illumination conditions. This allows that every time that the agent observes a person's face at a given distance and viewpoint, the corresponding images/observations are composed using a database of real faces and background images, instead of being generated by the simulator.

Considering that the goal of this tool is to test the recognition abilities of the agent, and not the navigation ones, navigation is simplified: the agent is placed in front of each person by the system. After being positioned, the agent analyzes its input images in order to detect and recognize human faces. Depending on the results of this analysis, the agent can change its relative pose. Every time that the agent changes its pose, the simulator composes the corresponding input images. The process is repeated until the agent observes all persons.

3.1 Image Acquisition System

Real face images are acquired at different yaw and pitch angles using a CCD camera mounted in a rotating structure (see diagram in fig. 1a). The person under scan is in a still position, while the camera, placed at the same height than the person's face and at a fixed distance of 140 cm, rotates in the axial plane (the camera height is adjustable). An encoder placed in the rotation axis calculates the face's yaw angle. There are not restrictions on the person's face expression. The system is able to acquire images with a resolution of 1°. However, in this first version of the testing tool, images are taken every 2°. The scanning process takes 25 seconds, and we use a 1280 x 960 pixels CCD camera (DFK 41BU02 model). In the frontal image, the face's size is about 200x250 pixels.

Variations in pitch are obtained by repeating the described processes, with the different required pitch angles are required. In each case, the camera height is maintained, but the person looks at a different reference points in the vertical axis, which are located at 160 cm in front of the person (see fig. 1a). In our experience, pitch angles of -15°, 0°, and 15° give account of typical human face variations.

It is important to remark that the acquisition device does not require any special installation, and therefore it can be used at different places. Thus, the whole acquisition process can be carried out at different locations (street environment, laboratory environment, mall environment, etc.). In our case we use at least two different locations for each person, one indoor (laboratory with windows), and one outdoors (gardens inside our school's campus). See example in fig. 1b.

Background images for each place, camera-height, and yaw-pitch angle combination are taken with the acquisition device, in order to be able to compose the final images to be shown to the agent.

In fig. 2 are shown some examples of images taken with the device.

3.2 Database Description

Face images of 50 persons compose the database. In each case 726 registered face images (121x3x2) are stored. The yaw angle range is -120° to 120°, with a resolution of 2°, which gives 121 images. For each different yaw, 3 different pitch angles are considered. For each yaw-pitch combination, indoor and outdoor images are taken. In addition, background images corresponding to the different yaw-pitch angles, place and camera-height combinations are also stored.

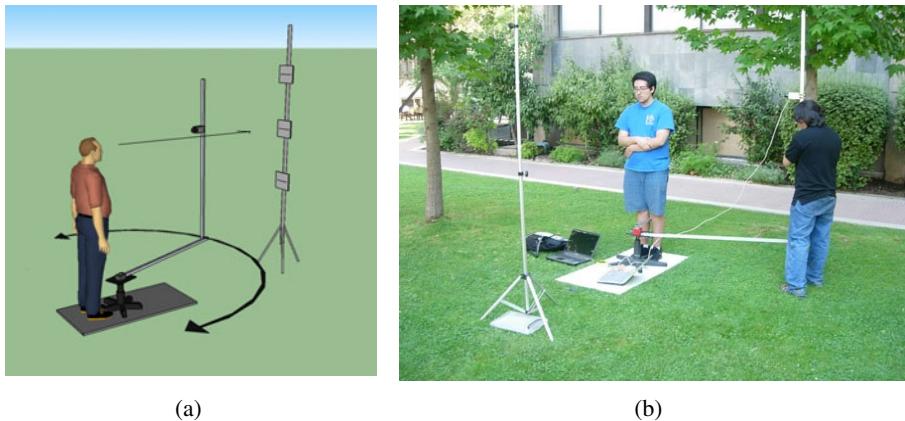


Fig. 1. (a) Diagram of the image acquisition system (b) The system operating in outdoors

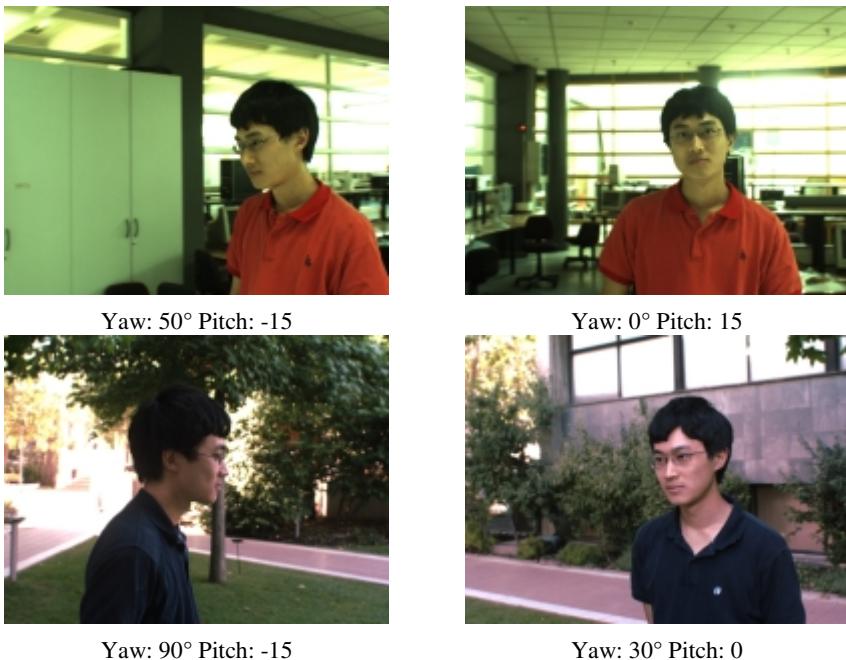


Fig. 2. Examples of images taken using the device in indoors/outdoors first/second row

3.3 Virtual Scenario Description and Agent Positioning

The scenario contains real face images of N persons. An observing agent, the one with the ability to recognize faces, has the possibility of navigating and making observations inside the scenario. Considering that the goal of this tool is to test the recognition abilities

of the agent navigation is simplified: the agent is placed at a fixed distance of 100 cm in front of each person by the system. Persons are chosen in a random order. In this first version of the system, the agent's camera and the observed face are at the same height, and the agent cannot move its head independently of the body. The following variations in the agent's relative position and viewpoint are incorporated before the agent starts to recognize person i :

- The pose of the agent is randomly modified in Δx_i , Δy_i , $\Delta \theta_i$. The maximal variation in each axis, $(\Delta x_{\max}, \Delta y_{\max}, \Delta \theta_{\max})$, is a simulation parameter.
- The face of person i is randomly rotated in θ_i^y (yaw angle), θ_i^p (pitch angle), and θ_i^r (roll angle). The maximal allowed rotation value in each axis, $(\theta_{\max}^y, \theta_{\max}^p, \theta_{\max}^r)$, is a simulation parameter.

After the relative position and orientation between the agent and the observed face are fixed, the simulator generates the corresponding observations (i.e. images) to the agent. This image generation process, more than a rendering process is a image composition process, in which real face and background images acquired using the device described in section 3.2, are used. The out-of-plane rotations are restricted to the available face images in the sagittal and lateral planes, while there are no restrictions for the in-plane rotations. In addition, the system selects at random whether person i is observed under indoor or outdoor illumination conditions.

The agent analyzes the generated images to detect and recognize human faces. Depending on the results of this analysis, the agent can change its relative pose using the following functions:

- *Move($\Delta x, \Delta y$)* : It changes the relative position of the agent in x and y . It is considered that the agent has the ability to perform omnidirectional movements.
- *Turn($\Delta \theta$)* : The agent turns in $\Delta \theta$. The angle's sign gives the turn direction.

Every time that the agent changes its pose, the simulator generates/composes the corresponding images. For instance, fig. 3 shows a given sequence of agent poses, and the corresponding images composed by the simulator. When the agent decides that it already knows the identity of the person, or that it cannot determine it, it sends this information to the simulation tool. Then, the simulator place the agent in front of the next person, and the whole process is repeated. If there is no next person, then the simulation finishes, and the simulation tool write a log file with the statistics of the recognition process.

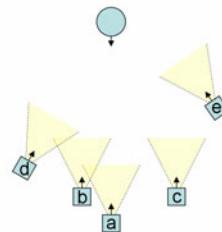
3.4 Testing Methodology

In order to recognize faces properly, the agent needs to have the following functionalities: (i) Face Detection. The agent detects a face (i.e. the face region) in a given image; (ii) Face Pose Estimation. The agent estimates the face's angular pose in the lateral, sagittal and coronal plane; (iii) Active Vision. Using information about the detected face and its pose, and other information observed in the input images, the agent can take actions in order to change the viewpoint of the sensor for improving face's perception; and (iv) Face Recognition. The identity of the person contained in the face image is determined.

In the testing tool these functionalities are implemented by the *DetectFace*, *EstimateFaceAngularPose*, *ImproveAgentPose*, and *RecognizeFace* functions (see fig. 4). The face recognition system under analysis should have at least the *RecognizeFace* function; having the other functions is optional. The testing tool can provide the functions that the face recognition system is not including. In case the testing tool is providing the *DetectFace* and *EstimateFaceAngularPose*, the face detection and face pose estimation accuracy can be fully controlled (the simulator knows the ground truth). They are simulation parameters to be defined in the testing protocol.

(a) => $x = 140, y = 0, \theta = 0$ (b) => $x = 120, y = -20, \theta = -10$ (c) => $x = 120, y = 45, \theta = 20$ (d) => $x = 100, y = -60, \theta = -30$ (e) => $x = 60, y = 65, \theta = 48$

observed face



agent in different positions

Fig. 3. Example of agent's positioning and the images/observations composed by the simulator. The agent is located in (a), and it moves to positions (b)-(e). In each case the input images are shown.

The simulation tool allows using the following modes:

- *Mode 1 - Recognition using Gallery*: The simulation tool generates a face gallery before the recognition process starts. The gallery contains one image of each person to be recognized. The gallery's images are frontal pictures (no rotations in any plane), taken under indoor illumination conditions. This is the standard operation mode, whose pseudo algorithm is shown in figure 4.

Initialization:

```
SetMaxVariationAgentInitialPosition ( $\Delta x_{\max}, \Delta y_{\max}, \Delta \theta_{\max}$ ) ;
```

```
SetMaxVariationFaceRotationAngles ( $\theta_x^y, \theta_y^p, \theta_z^r$ ) ;
```

```
num_recognized_faces=num_false_positives=0;
```

Testing:

```
for (i=0;i<N;i++)
    SetRobotInitialPose();
    SetFaceInitialPose();
    SetIndoorOutdoorIllumination();
    currentImage =GetImage();
    id=RecognizePerson(currentImage);
    if (id==GetPersonID(i))
        num_recognized_faces+=1;
    else if (id!=NO_IDENTIFICATION)
        num_false_positives+=1;
StoreStatistics(num_recognized_faces, num_false_positives);
end;
```

Recognition:

```
RecognizePerson(image)
while(1)
    if(face=DetectFace(image))==NO_IDENTIFICATION)
        return(NO_IDENTIFICATION);
    faceAngularPose=EstimateFaceAngularPose(image);
    if(face.size<MIN_SIZE OR |faceAngularPose.yaw|>MIN_YAW OR
       |faceAngularPose.pitch|>MIN_PITCH)
        ImproveAgentPose(face.position, face.size, faceAngularPose);
        image=GetImage();
    else
        result=RecognizeFace(face)
        if(result.confidence<threshold)
            return(NO_IDENTIFICATION)
        else
            return(result.id)
```

Fig. 4. Pseudo algorithm of testing procedure in mode 1 (recognition using gallery)

- *Mode 2 - Recognition without using Gallery*: There is no gallery. The agent needs to cross two times the virtual scenario. In the first round, it should create the database online. In the second round, the gallery is used to recognize the persons. In both rounds, the agent see the person's faces at variable distance and angles, in indoor or outdoor illumination conditions. The persons pose and the illumination conditions are randomly chosen.

In each of the two described modes it can be activated the option $-m$, which allows observing multiple persons in some images. In this case, the persons were previously scanned together by the image acquisition system.

4 Results

In order to obtain a first validation of the applicability of the testing tool, three unsupervised face recognition methods are compared. In the reported experiments, face detection, face pose estimation, and active vision are provided by the testing tool.

4.1 Face Recognition Methods

Three local-matching face recognition methods are implemented: histograms of LBP (Local Binary Patterns) features [22], Gabor-Jet features with Borda count classifiers [23], and histograms of WLD (Weber Local Descriptor) features. The first two methods have shown a very good performance in comparative studies of face recognition systems [10][23]. The third method is being proposed here, and it is based in the recently proposed WLD feature [24]. In all cases, the methods' parameters are adapted/adjusted using standard face datasets, and not using the face images that the testing tool includes.

Following the results reported in [10], two different flavors of the histograms of LBP features method are implemented, one using the histogram intersection (HI) similarity measure, and one using the Chi square (XS) measure. In both cases face images are scaled to 81x150 pixels and divided into 40 regions to compute the LBP histograms. The two implemented face recognitions systems are called LBP-HI-40 and LBP-XS-40. The implemented Gabor-based method uses 5 scales and 8 orientations, and face images scale to 122x225 pixels, as reported in [10].

Finally, in the case of the WLD based method, after extensive experimentation using the FERET, BioID and LFW databases, the following parameters were selected: histogram intersection and Chi square similarity measures, face images scaled to 93x173 pixels and divided into 40 regions to compute the WLD histograms, 2 dominant orientations ($T=2$), and 26 cells in each orientation ($C=26$).

4.2 Recognition Results

In a first set of experiments, the recognition rate of the different methods is compared under different viewpoint conditions; the yaw angle of the observed faces is uniformly selected (random value) in the range $-/+ \theta_{\max}^y$. The other simulation parameters are kept unchanged ($\Delta x_{\max} = \Delta y_{\max} = \Delta \theta_{\max} = \theta_{\max}^p = \theta_{\max}^r = 0$). In the experiments no

active vision mechanisms are used, and a face detection rate of 100% is considered. Table 1 shows the obtained results. Main conclusions of these experiments are: (i) LBP based methods that use the Chi square similarity measure are more robust to yaw rotations than Gabor and WLD based methods, and (ii) all methods are robust to yaw rotation in the range +/-30°.

Table 1. Top-1 recognition rates under different maximal yaw angles of the observed face (θ_{\max}^y). The other parameters are not varied ($\Delta x_{\max} = \Delta y_{\max} = \Delta \theta_{\max} = \theta_{\max}^p = \theta_{\max}^r = 0$).

Method	θ_{\max}^y								
	5°	10°	15°	20°	25°	30°	35°	40°	60°
LBP-HI-40	1.00	1.00	1.00	1.00	0.95	0.95	0.80	0.85	0.55
LBP-XS-40	1.00	1.00	1.00	0.95	0.95	0.95	0.85	0.75	0.30
GJD-BC	1.00	1.00	1.00	0.95	0.85	0.85	0.75	0.80	0.35
WLD-HI-40	1.00	1.00	0.95	0.95	0.90	0.90	0.85	0.70	0.45
WLD-XS-40	1.00	1.00	0.90	0.90	0.95	0.90	0.75	0.70	0.45

Table 2. Top-1 recognition rates under different maximal yaw and pitch angles of the observed face ($\theta_{\max}^y, \theta_{\max}^p$), different maximal agent's positioning errors ($\Delta x_{\max}, \Delta y_{\max}$), and variable face pose estimation error (pe)

Method	$\theta_{\max}^y = \pm 45$					
	$\theta_{\max}^p = 0$	$\theta_{\max}^p = 0$	$\theta_{\max}^p = \pm 15$			
$\Delta x_{\max} = 20$	$\Delta x_{\max} = 40$	$\Delta x_{\max} = 20$	$\Delta x_{\max} = 40$	$\Delta x_{\max} = 20$	$\Delta x_{\max} = 40$	$\Delta x_{\max} = 40$
$\Delta y_{\max} = 20$	$\Delta y_{\max} = 40$	$\Delta y_{\max} = 20$	$\Delta y_{\max} = 40$	$\Delta y_{\max} = 20$	$\Delta y_{\max} = 40$	$\Delta y_{\max} = 40$
$pe = 40\%$	$pe = 40\%$	$pe = 40\%$	$pe = 40\%$	$pe = 40\%$	$pe = 80\%$	$pe = 80\%$
LBP-HI-40	0.85	0.85	0.80	0.75	0.75	0.70
LBP-XS-40	0.85	0.85	0.80	0.80	0.80	0.75
GJD-BC	0.85	0.80	0.75	0.70	0.70	0.65
WLD-HI-40	0.80	0.85	0.70	0.65	0.70	0.65
WLD-XS-40	0.80	0.85	0.70	0.65	0.70	0.65

In a second set of experiments, the recognition rate of the different methods is compared under more uncontrolled conditions:

- The yaw angle of the observed faces is uniformly selected (random value) in the range +/-45°, and the pitch angle in the range +/-15°. The roll angle is not modified ($\theta_{\max}^r = 0$).
- The position of the observer agent is modified in each axis, by a random value uniformly selected in the range +/-20 or +/-40 centimeters. The agent is not rotated ($\Delta \theta_{\max} = 0$).

The following face detection and pose estimation conditions are considered: (i) Face detection rate of 80% with no false positives, (ii) Face pose estimation with an error, pe , uniformly selected (random value) in the range +/-40% or +/-80% of the estimated value, and (iii) Active vision mechanisms as shown in the procedure of fig. 4.

Table 2 shows the obtained results. Main conclusions of these experiments are: (i) LBP based methods are more robust to the defined uncontrolled conditions than Gabor and WLD based methods, (ii) the agent's initial position error has a low influence on the final performance of the recognition systems, (iii) a maximal error of +/-15° in the pitch angle reduces in ~5% the face recognition rate, and (iv) a pose estimation error increase from 40% to 80% reduces in ~5% the recognition rate.

5 Conclusions and Projections

In this article, a tool for testing face recognition systems under uncontrolled conditions is proposed. The testing tool combines the use of a simulator with real face and background images taken under real-world conditions. Inside the simulated environment, an observing agent can navigate and observe the real face images, at different distances, angles and with indoor or outdoor illumination. During the face recognition process, the agent can actively change its viewpoint and relative distance to the faces in order to improve the recognition results. The simulation tool provides all navigation and positioning functionalities to the agent, except the ones related with the detection, alignment and recognition of faces.

The applicability of the proposed tool is validated in the comparison of three state of the art face recognition methods, histograms of LBP features, Gabor-Jet features with Borda count classifiers, and histograms of WLD features.

In order to be able to share the use of the proposed tool with other researchers of the face recognition community, the following procedure will be implemented:

1. A DLL of the testing tool with a sample of the database containing the images of 10 individuals will be distributed upon request³. The DLL will include a Visual Studio project with an example of use. In a second stage, a Linux library will be also provided. The goal of this DLL is that researchers can make parameters' adjustment and preliminary tests of their face recognition methods.
2. In order to carry out tests using the complete database, researchers will submit a compiled version of their face recognition method, linked to the provided DLL. After testing, results will be sent back automatically.

We are currently writing a technical report where the outlined procedure will be described in detail, as well as the conditions of use of the tool. We are also implementing a website to manage the described procedure.

References

1. Zhao, W., Chellappa, R., Rosenfeld, A., Phillips, P.J.: Face Recognition: A Literature Survey. *ACM Computing Surveys*, 399–458 (2003)
2. Tan, X., Chen, S., Zhou, Z.-H., Zhang, F.: Face recognition from a single image per person: A survey. *Pattern Recognition* 39, 1725–1745 (2006)

³ The full database cannot be made available because of its size.

3. Chellappa, R., Wilson, C.L., Sirohey, S.: Human and Machine Recognition of Faces: A Survey. Proceedings of the IEEE 83(5), 705–740 (1995)
4. Abate, A.F., Nappi, M., Riccio, D., Sabatino, G.: 2D and 3D face recognition: A survey. Pattern Recognition Letters 28, 1885–1906 (2007)
5. Face Recognition Home Page, <http://www.face-rec.org/databases/>
6. Call for Papers Special Issue in Real-World Face Recognition. IEEE Trans. on PAMI, http://www.eecs.northwestern.edu/~ganghua/ghweb/CFP_TPAMI_FR.htm
7. Faces in Real-Life Images Workshop, ECCV 2008 (October 17, 2009), <http://hal.inria.fr/REALFACES2008/en>
8. Labeled Faces in the Wild Database, <http://vis-www.cs.umass.edu/lfw/index.html>
9. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. University of Massachusetts, Amherst, Technical Report 07-49 (October 2007)
10. Ruiz-del-Solar, J., Verschae, R., Correa, M.: Recognition of Faces in Unconstrained Environments: A Comparative Study. EURASIP Journal on Advances in Signal Processing (Recent Advances in Biometric Systems: A Signal Processing Perspective) 2009, article ID 184617, 19 pages (2009)
11. Jones, M.: Face Recognition: Where We Are and Where To Go From Here, Mitsubishi Electric Research Laboratories Technical Report TR2009-023 (June 2009)
12. Face Recognition Grand Challenge, Official website <http://www.frvt.org/FRGC/>
13. Phillips, P.J., Wechsler, H., Huang, J., Rauss, P.: The FERET database and evaluation procedure for face recognition algorithms. Image and Vision Computing J. 16(5), 295–306 (1998)
14. Phillips, P., Flynn, P., Scruggs, T., Bowyer, K., Chang, J., Hoffman, K., Marques, J., Min, J., Worek, W.: Overview of the Face Recognition Grand Challenge. In: Proc. of the IEEE Conf. Computer Vision and Pattern Recognition – CVPR 2005, vol. 1, pp. 947–954 (2005)
15. Gross, R.: Face Databases. In: Li, S., Jain, A.K. (eds.) Handbook of Face Recognition, pp. 301–327. Springer, Heidelberg (2005)
16. Yale University Face Image Database, publicly <http://cvc.yale.edu/projects/yalefaces/yalefaces.html>
17. BioID Face Database, publicly <http://www.humanscan.de/support/downloads/facedb.php>
18. AR Face Database public site, http://cobweb.ecn.purdue.edu/~aleix/aleix_face_DB.html
19. Flynn, P.J., Bowyer, K.W., Phillips, P.J.: Assessment of time dependency in face recognition: An initial study. In: Audio and Video-Based Biometric Person Authentication, pp. 44–51 (2003)
20. Yale Face Database B, <http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html>
21. PIE Database. Basic infomation in, http://www.ri.cmu.edu/projects/project_418.html
22. Ahonen, T., Hadid, A., Pietikainen, M.: Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Trans. on Patt. Analysis and Machine Intell. 28(12), 2037–2041 (2006)
23. Zou, J., Ji, Q., Nagy, G.: A Comparative Study of Local Matching Approach for Face Recognition. IEEE Trans. on Image Processing 16(10), 2617–2628 (2007)
24. Chen, J., Shan, S., He, C., Zhao, G., Pietikäinen, M., Chen, X., Gao, W.: WLD: A Robust Local Image Descriptor. IEEE Trans. on Patt. Analysis and Machine Intell., TPAMI-2008-09-0620 (in press)

Thermal Face Recognition Using Local Interest Points and Descriptors for HRI Applications*

Gabriel Hermosilla, Patricio Loncomilla, and Javier Ruiz-del-Solar

Department of Electrical Engineering, Universidad de Chile
Center for Mining Technology, Universidad de Chile
jruizd@ing.uchile.cl

Abstract. In this article a robust thermal face recognition methodology based on the use of local interest points and descriptors, is proposed. The methodology consists of the following stages: face segmentation, vascular network detection, wide baseline matching using local interest points and descriptors, and classification. The main contribution of this work is the use of a standard wide baseline matching methodology for the comparison of vascular networks from thermal face images. The proposed methodology is validated using a database of thermal images. This work could be of high interest for HRI applications related with the visual recognition of humans, as the ones included in the RoboCup @Home league, because the use of thermal images may overcome limitations such as dependency on illumination conditions and facial expressions.

Keywords: Face Recognition, Thermal Images, Blood Vessels Matching, SIFT Matching, RoboCup @Home.

1 Introduction

Face analysis plays an important role in building HRI (Human-Robot Interaction) interfaces that allow humans to interact with robot systems in a natural way. Face information is by far the most used visual cue employed by humans. There is evidence of specialized processing units for face analysis in our visual system [1]. Face analysis allows localization and identification of other humans, as well as interaction and visual communication with them. Therefore, if human-robot interaction must achieve the same efficiency, diversity, and complexity that human-human interaction has, face analysis should be extensively employed in the construction of HRI interfaces.

Currently, computational face analysis is a very lively and expanding research field. Face recognition, i.e. the specific process for determining the identity of an individual contained in an image area which has been already identified as containing a face (by a face detection system) and already aligned (by a face alignment process which usually includes eye detection), is a functional key for personalizing robot services and

* This research was partially funded by FONDECYT under Project Number 1090250.

for determining robot behaviors, usually depending on the identity of the human interlocutor. Many different face recognition approaches have been developed in the last few years [1]-[5], ranging from classical Eigenspace-based methods (e.g. eigenfaces [6]), to sophisticated systems based on high-resolution images or 3D models. Many of these methods are well suited to specific requirements of applications such as biometry, surveillance, or security. HRI applications have their own requirements (variable illumination conditions, variable face expressions, just one image per person, in many cases), and therefore some approaches are better suited for them. It would be useful for developers of face recognition systems for HRI to have some guidelines about the advantages of some thermal methodologies over others.

In this context, the aim of this work is to propose and evaluate the usage of vascular networks from thermal face images for recognition purposes, and the use of wide baseline methods for matching both, thermal images and vascular networks.

The main contribution of this work is the use, for the first time, of a standard wide baseline matching methodology for the matching of vascular networks from thermal face images. This approach enables to build recognition systems using just one image per subject, which are robust to variable illumination and variable face expressions.

This paper is organized as follows. The related work is described in section 2. The proposed system is explained in section 3. Description of the experiments and results are outlined in section 4. Finally, conclusions are given in section 5.

2 Related Work

Biometric techniques have attracted increasing interest from both research and industrial communities because of the strong immunity to forgery that they can achieve. Face recognition is one of the main topics in the set of biometric applications as it is one of the most human-used, non-intrusive and user-friendly approaches for biometric recognition and have been a very important issue in the computer vision community, generating a lot of related research on image processing and statistical classifiers and achieving encouraging results [1]-[4][6][7]. Visible-spectrum images have high variability because they are produced by reflection in surfaces, which has strong dependence on luminosity and spatial distribution of the light sources which usually have strong differences in time, and their dependence on reflectivity make possible to fool the system using some simple tricks like photographs or dummy faces. These problems encouraged the interest in the use of thermal images for face recognition because of the great resistance to forgery it can achieve and their high immunity to illumination changes and several other sources of variability in the acquisition of images that enables these systems to operate in real non-controlled environments [8][9]. Face recognition systems using thermal images use techniques adapted from visible-spectrum image face recognition [8]-[10], and new techniques that are specific for thermal images [18][19]. A comparison of several techniques for thermal faces is shown in [20]. Algorithms for obtaining vascular networks from thermal images using segmentation and morphologic operators [18][19] are a very useful tool for building face recognition techniques as they provide a very simple and

repeatable imprint representation that is unique for each person, and enable recognition through the matching of these imprints using some specialized or general image matching methodology from the impressive rich variety of techniques that are devoted to this last task.

Wide baseline matching (object recognition) approaches based on local interest points (invariant features) have become increasingly popular and have experienced an impressive development in the last years [14][11][16][21]. Typically, local interest points are extracted independently from both a test and a reference image, and then characterized by invariant descriptors, and finally the descriptors are matched until a given transformation between the two images is obtained. Most employed local detectors are the Harris detector [12] and the Lowe's sDoG+Hessian detector [14], being the Lowe's detector multiscale and the Harris detector single scale. Best performing affine invariant detectors are the Harris-Affine and the Hessian-Affine [17], but they are too slow to be applied in general-purpose applications. The most popular and best performing invariant descriptor [17] is the SIFT (Scale Invariant Feature Transform) [14]. For selecting the local detector and invariant descriptor to be used in a given application it should be taken into account the algorithm's accuracy, robustness and processing speed. Lowe's system [14] using the SDoG+Hessian detector, SIFT descriptors and a probabilistic hypothesis rejection stage is a popular choice, given its recognition capabilities, and near real-time operation. However, Lowe's system main drawback is the large number of false positive detections. This is a serious problem when using it in real world applications.

One of the main weaknesses of Lowe's algorithm is the use of just a simple probabilistic hypothesis rejection stage, which cannot successfully reduce the number of false positives. Loncomilla and Ruiz-del-Solar (L&R) propose a system that reduces largely the number of false positives by using several hypothesis rejection stages [15][21]. This includes a fast probabilistic hypothesis rejection stage, a linear correlation verification stage, a geometrical distortion verification stage, a pixel correlation verification stage, a transformation fusion procedure, and the use of the RANSAC algorithm and a semi-local constraints test. In [21] are compared the Lowe's and the L&R systems using 100 pairs of real-world high-textured images (variations in position, view angle, image covering, partial occlusion, in-plane and out-of-the-plane rotation). The results show that in this dataset the L&R system reduces the false positive rate from 85.5% to 3.74%, by increasing the detection rate by 5%. For this reason we choose to use this system in this work.

3 Proposed Thermal Face Recognition System

The proposed approach uses wide-baseline matching of face vascular networks obtained from thermal images. The vascular networks are obtained through skin segmentation and morphological operators. The image matching stage uses SIFT descriptors and the L&R system for verifying correspondences and generating a final geometrical transformation that relate the vascular networks. A classifier makes a final decision about the recognition. The proposed approach mixes the uniqueness of

the representation obtained through vascular networks, the robustness of the recognition based on local descriptors, and a classifier that make the final decision for providing a new methodology for thermal face recognition. A description of the system is provided in the next subsections.

3.1 System Outline

The system is composed by four stages: face segmentation, vascular network extraction, wide-baseline matching and a statistical classifier. A block diagram of the system is shown in figure 1.

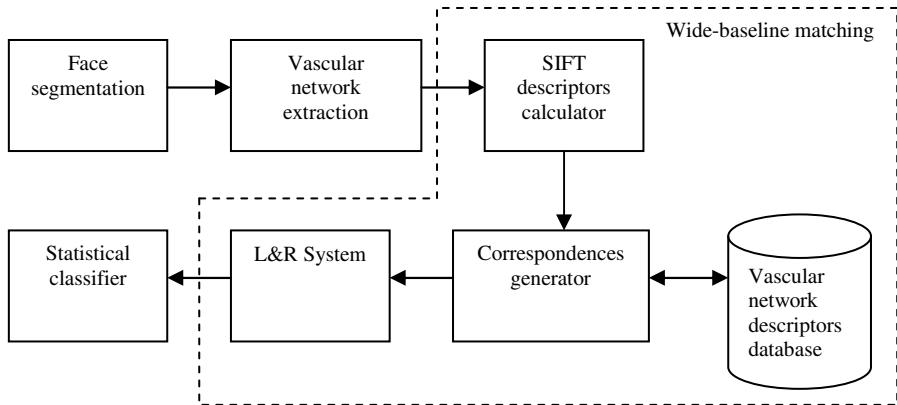


Fig. 1. Diagram of the proposed thermal face recognition system

3.2 Face Segmentation

A skin detection model in the thermal spectrum is created by modelling both, the skin pixel intensity distribution, and the non-skin distribution, as mixtures of Gaussians (MoG). Both the skin and non-skin intensity distributions are modelled as the mixture of four Gaussians:

$$p_{SKIN}(x) = \sum_{i=1}^4 \omega_s \frac{1}{\sqrt{2\pi}\sigma_s} \exp\left(-\frac{1}{2} \frac{(x - \mu_s)^2}{\sigma_s^2}\right) \quad (1)$$

$$p_{NON-SKIN}(x) = \sum_{i=1}^4 \omega_N \frac{1}{\sqrt{2\pi}\sigma_N} \exp\left(-\frac{1}{2} \frac{(x - \mu_N)^2}{\sigma_N^2}\right) \quad (2)$$

A pixel is detected as skin when its intensity has a larger probability of belonging to the skin class than to the non-skin class. The application of this scheme on a face thermal image enables the calculation of a skin mask:

$$I_{MASK}(i, j) = \begin{cases} 1 & \text{if } p_{SKIN}(I(i, j)) > p_{NON-SKIN}(I(i, j)) \\ 0 & \text{if } p_{SKIN}(I(i, j)) < p_{NON-SKIN}(I(i, j)) \end{cases} \quad (3)$$

The skin image is obtained by multiplying the thermal image and the skin mask, as it is shown in (4). General parameters for the Gaussians cannot be obtained as they depend on the response of the particular camera to thermal intensity.

$$I_{SKIN}(i, j) = I_{MASK}(i, j)I(i, j) \quad (4)$$

3.3 Vascular Network Extraction

A smoothed skin image is calculated by filtering the skin image using a 3x3 uniform low-pass filter:

$$I_{SMOOTH} = I_{SKIN} * \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad (5)$$

The smoothed image is eroded and dilated using morphological operations between the image and a 3x3 diagonal cross operator for obtaining a morphological opened image:

$$I_{OPENED} = \left(I_{SMOOTH} \Theta \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \right) \oplus \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad (6)$$

Finally, the vascular network image is calculated by subtracting the smoothed image and the opened one:

$$I_{VASCULAR} = I_{SMOOTH} - I_{OPENED} \quad (7)$$

3.4 Wide Baseline Matching Using Local Interest Points

SIFT descriptors are obtained from the vascular network image using the standard Lowe's methodology [14]. Each SIFT descriptor represents a small patch in the image, and it contains a position (x_0, y_0) , an orientation (θ) , a scale (σ) and a vector of characteristics (v_1, \dots, v_{128}) calculated from the gradients in the patch. Correspondences between two vascular images are obtained by matching pairs of descriptors with similar vectors of characteristics using the Euclidean distance.

Each correspondence between two descriptors has an associated translation (t_x, t_y) , rotation θ , and scale change s that warps one patch into the other. Coherent groups of correspondences that generate similar warping parameters (t_x, t_y, θ, s) are found via a Hough transform. For each coherent group of correspondences a probability test is carried out. Following, the L&R methodology [15][21], additional geometrical and statistical verification tests are applied in order to eliminate wrong groups of correspondences, which do not represent similar patches in both images.

In figure 2 are shown two examples of matches between pairs of thermal images and vascular network images, corresponding to the same person, and to different persons.

3.5 Classification

A final coherent group of correspondences is obtained for each test image by matching it to all gallery images. From all the final groups of correspondences, the most numerous one is selected in order to decide the correct gallery image, and to recognize the identity of the person.

4 Results

Results are obtained using the UChile Thermal Faces Database. This database contains 156 320x240 thermal face images that correspond to 6 images per subject and 26 subjects. The images were using a Cedip Jade UC infrared camera.

The proposed methodology consisting on the matching of vascular networks of thermal images is compared with the matching of plain thermal images (see examples in fig. 2). In both cases the matching is implemented using Lowe's system and the improved L&R system. Thus, the following four recognition systems are implemented and compared:

- *Lowe-Thermal*: Lowe's matching of plain thermal images.
- *L&R-Thermal*: L&R's matching of plain thermal images.
- *Lowe-VascularNet*: Lowe's matching of vascular networks of thermal images.
- *L&R-VascularNet*: L&R's matching of vascular networks of thermal images.

4.1 Database

The UChile Thermal Faces Database was obtained by using a Cedip Jade UC infrared camera. This camera has a spectral range between 8-12 μm and a resolution of 320x240 pixels [22]. A video was recorded for each of the 26 subjects kept in a fixed position. The subjects gesticulated vowels (2,000 frames captured), and the happy/sad/anger expressions (100 frames captured). From the video for each subject, three frame images containing different vowels, and three frame images containing different expressions were selected to build the database. The images are divided into categories depending on the frame number in the video.

- V1: Frame 100 in the vowel's sequence video, 26 images
- V2: Frame 1,000 in the vowel's sequence video, 26 images
- V3: Frame 1,500 in the vowel's sequence video, 26 images
- E1: Frame 45 in the happy expression video, 26 images
- E2: Frame 45 in the sad expression video, 26 images
- E3: Frame 45 in the anger expression video, 26 images

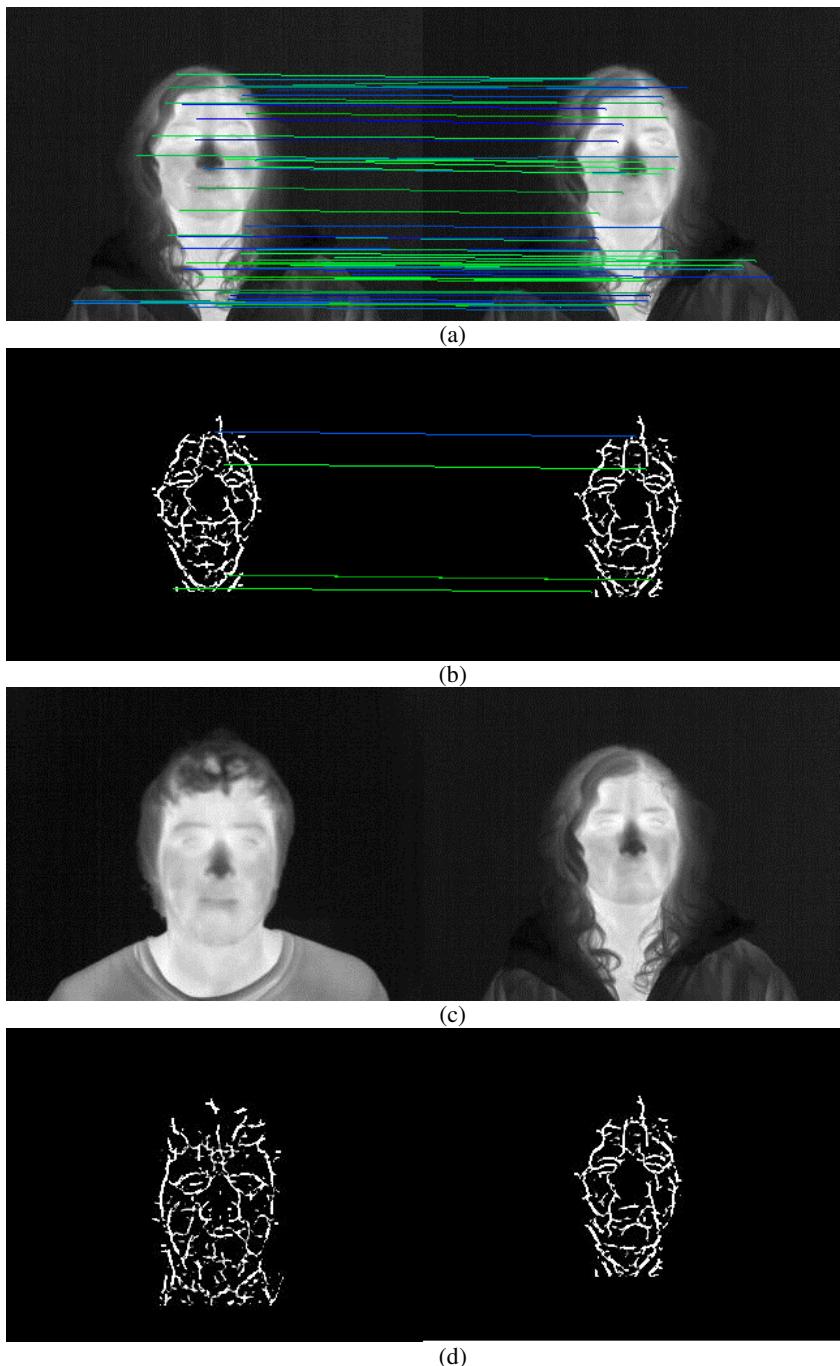


Fig. 2. Matching of pairs of thermal images ((a)-(b)), and the corresponding vascular network images ((c)-(d)). In (a) and (b) the images corresponds to the same person, while in (c) and (d) they correspond to different persons.

4.2 Recognition Results

Cross-validation is used for evaluating the performance of the different methods in the database. From the set of categories {V1, V2, V3, E1, E2, E3} all the possible pairs of categories (A,B) are formed. For each pair (A,B), the images in A are selected as test images, while the images in B are selected as gallery images. Each image in the test set A is recognized in the gallery B, which generate both correct and incorrect recognitions. The proportion of correct recognitions (recognition rate) for all the pairs of categories is shown in tables 1 to 4. The mean and variance of the overall recognition rates for the four methods are shown in table 5.

Table 1. Recognition rates for *Lowe-Thermal* method

Test\Gallery	V1	E1	V2	E2	V3	E3
V1	100	84.6	96.1	80.7	100	96.1
E1	92.3	100	92.3	92.3	96.1	100
V2	100	88.4	100	92.3	100	96.1
E2	84.6	96.1	80.7	100	80.7	100
V3	100	92.3	96.1	76.9	100	88.4
E3	96.1	100	96.1	100	96.1	100

Table 2. Recognition rates for *L&R-Thermal* method

Test\Gallery	V1	E1	V2	E2	V3	E3
V1	100	92.3	100	80.7	100	88.4
E1	96.1	100	96.1	88.4	88.4	100
V2	96.1	100	100	80.7	100	100
E2	88.4	100	80.7	100	84.6	100
V3	92.3	88.4	100	80.7	100	84.6
E3	88.4	100	92.3	96.1	96.1	100

Table 3. Recognition rates for *Lowe-VascularNet* method

Test\Gallery	V1	E1	V2	E2	V3	E3
V1	100	88.4	100	84.6	96.1	100
E1	96.1	100	96.1	100	96.1	96.1
V2	100	92.3	100	92.3	100	96.1
E2	96.1	100	88.4	100	73.0	100
V3	100	84.6	100	84.6	100	92.3
E3	100	96.1	100	96.1	100	100

Table 4. Recognition rates for *L&R-VascularNet* method

Test\Gallery	V1	E1	V2	E2	V3	E3
V1	100	88.4	96.1	76.9	92.3	96.1
E1	96.1	100	92.3	92.3	92.3	88.4
V2	96.1	100	100	88.4	100	100
E2	92.3	100	84.6	100	96.1	96.1
V3	88.4	96.1	100	76.9	100	96.1
E3	100	88.4	92.3	96.1	96.1	100

Table 5. Overall recognition rates of the four methods

Method	Mean	Standard Deviation
<i>L&R-Thermal</i>	95.7	6.2
<i>Lowe-Thermal</i>	94.3	6.1
<i>L&R-VascularNet</i>	94.2	6.8
<i>Lowe-VascularNet</i>	93.9	6.9

4.3 Recognition Analysis

The L&R system applied directly over the thermal image generate the better results; however, the recognition rates for the four methods have comparable means and variances. The difference between the means of the better and the worst method is around 1.8, which is small compared to the variances around 6.5 obtained in the experiments. These experiments show that the transformation from the original thermal images to the vascular network representations preserve information that permit the recognition of the subjects.

5 Conclusions

The main contribution of this work is the use, for the first time, of a standard wide baseline matching methodology for the matching of vascular networks from thermal face images. This work is a preliminary step in the development of face recognition systems using vascular networks detection in thermal images followed by general image matching algorithms. All the variants compared in this work have similar performance for recognizing faces from thermal images. This results shows that vascular network images preserve important discriminative information about the original thermal images. As the vascular network representation has a simple and particular structure, specialized methods can be developed in future works for increasing dramatically the face recognition rates using thermal imaging systems.

As future work we would like to increase largely the size of the database, and to include outdoor illumination. The proposed recognition methodology will be validated in this extended database.

References

1. Sinha, P., Balas, B., Ostrovsky, Y., Russell, R.: Face Recognition by Humans: 19 Results All Computer Vision Researchers Should Know About. Proc. of the IEEE 94(11), 1948–1962 (2006)
2. Zhao, W., Chellappa, R., Rosenfeld, A., Phillips, P.J.: Face Recognition: A Literature Survey. ACM Computing Surveys, 399–458 (2003)
3. Tan, X., Chen, S., Zhou, Z.-H., Zhang, F.: Face recognition from a single image per person: A survey. Pattern Recognition 39, 1725–1745 (2006)
4. Chellappa, R., Wilson, C.L., Sirohey, S.: Human and Machine Recognition of Faces: A Survey. Proceedings of the IEEE 83(5), 705–740 (1995)
5. Face Recognition Homepage (January 2008), <http://www.face-rec.org/>
6. Turk, M., Pentland, A.: Eigenfaces for Recognition. Journal of Cognitive Neuroscience 3(1), 71–86 (1991)
7. Ruiz-del-Solar, J., Verschae, R., Correa, M.: Recognition of Faces in Unconstrained Environments: A Comparative Study. EURASIP Journal on Advances in Signal Processing (Recent Advances in Biometric Systems: A Signal Processing Perspective) 2009, article ID 184617, 19 pages (2009)
8. Socolinsky, D.A., Selinger, A.: A comparative Analysis of face recognition performance with visible and thermal infrared imagery. In: Proc. ICPR 2002, Quebec, Canada (August 2002)

9. Selinger, A., Socolinsky, D.: Appearance-Based Facial Recognition Using Visible and Thermal Imagery: A Comparative Study, Tech. Rep., Equinox Corporation (2001)
10. Desa, S., Hati, S.: IR and Visible Face Recognition using Fusion of Kernel Based Features. In: ICPR 2008, pp. 1–4 (2008)
11. Ferrari, V., Tuytelaars, T., Van Gool, L.: Simultaneous Object Recognition and Segmentation by Image Exploration. In: Pajdla, T., Matas, J. (eds.) ECCV 2004. LNCS, vol. 3021, pp. 40–54. Springer, Heidelberg (2004)
12. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proc. 4th Alvey Vision Conf., Manchester, UK, pp. 147–151 (1998)
13. Lowe, D.: Local feature view clustering for 3D object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, Hawaii, pp. 682–688. IEE Press, New York (2001)
14. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. Int. Journal of Computer Vision 60(2), 91–110 (2004)
15. Loncomilla, P., Ruiz-del-Solar, J.: A Fast Probabilistic Model for Hypothesis Rejection in SIFT-Based Object Recognition. In: Martínez-Trinidad, J.F., Carrasco Ochoa, J.A., Kittler, J. (eds.) CIARP 2006. LNCS, vol. 4225, pp. 696–705. Springer, Heidelberg (2006)
16. Mikolajczyk, K., Schmid, C.: Scale & Affine Invariant Interest Point Detectors. Int. Journal of Computer Vision 60(1), 63–96 (2004)
17. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Machine Intell. 27(10), 1615–1630 (2005)
18. Buddharaju, P., Pavlidis, I.: Multi-Spectral Face Recognition - Fusion of Visual Imagery with Physiological Information. In: Face Biometrics for Personal Identification: Multi-Sensory Multi-Modal Systems, pp. 91–108. Springer, Heidelberg (January 2007)
19. Buddharaju, P., Pavlidis, I., Manohar, C.: ‘Face Recognition Beyond the Visible Spectrum. In: Advances in Biometrics: Sensors, Algorithms and Systems, pp. 157–180. Springer, Heidelberg (October 2007)
20. Hermosilla, G., Ruiz-del-Solar, J., Verschae, R., Correa, M.: Face Recognition using Thermal Infrared Images for Human-Robot Interaction Applications: A Comparative Study. In: 6th IEEE Latin American Robotics Symposium – LARS 2009, Valparaiso, Chile (CD Proceedings) (October 29–30, 2009)
21. Ruiz del Solar, J., Loncomilla, P.: Robot Head Pose Detection and Gaze Direction Determination Using Local Invariant Features. Advanced Robotics 23(3), 305–328 (2009)
22. Jade camera link,
http://www.edevis.de/products/products_ircameras_jade_uc_en.php

On Progress in RoboCup: The Simulation League Showcase

Thomas Gabel and Martin Riedmiller

Machine Learning Laboratory

University of Freiburg, 79110 Freiburg im Breisgau, Germany

{tgabel,riedmiller}@informatik.uni-freiburg.de

Abstract. The annual RoboCup competitions are certainly relevant to present times as they elect the best team of the current year. With respect to RoboCup's well-known 2050 vision, however, it is crucial to assess the general progress being made not just qualitatively, but also in a quantitative manner. Although this is difficult to accomplish in most leagues, the recent development and circumstances in the soccer simulation league led to a unique situation which allowed us to perform an extensive experimental evaluation by means of which we can empirically measure the progress of playing performance made over a number of successive years. The findings we present in this paper, in particular the fact that significant improvements in the level of play can be verified quantitatively in the 2D soccer simulation league, may serve as a showcase for the progress made by the RoboCup initiative in general.

1 Introduction

The 2D soccer simulation league might be called the grandmother of all RoboCup leagues. Its origins date back to Itsuki Noda's first implementations of the Soccer Server, a two-dimensional soccer simulation software in 1994 [1,2] as well as to the Pre-RoboCup event at IROS 1996 in Osaka where simulated soccer agents competed with one another within official competitions for the first time. Ever since the 2D soccer simulation league has seen 14 successive versions of its simulation software and has competed in 13 world championship tournaments as well as in more than 50 major regional competitions. With such a long history within the RoboCup initiative, a natural question to ask is what progress has been made throughout the years.

This is the topic we want to focus on in this paper. In particular, we want to look back and investigate and reflect on the following questions:

- What are the exact conditions that are required to allow for a *meaningful* retrospection?
- If such conditions can be identified, has there been any *provable* progress in performance in soccer simulation 2D during the last few years?
- If so, how close are teams and how reliable are the results of a single game with respect to the noise in the simulation? Has there been something like

convergence in the teams' further development and has perhaps a certain saturation level in their performance been reached?

These questions are extremely hard to answer in other, hardware-based leagues. New technological solutions, new mechanical designs combined with strong rule changes introduced year by year are certainly necessary on the road towards RoboCup's 2050 vision, but they make it hard, if not impossible, to perform a meaningful comparison between the achievements and performance of RoboCup teams over the years. Indeed, sometimes it may have happened that the wheel has been invented twice, whereas at other instances former RoboCup team members minify or disdain contemporary approaches pointing out that certain problems had already been solved in their times. Hence, subjective assessments often superimpose and conceal the true progress.

By contrast, a stable platform allows for a reliable and meaningful evaluation of the true progress made. RoboCup in general, however, is known to be a highly dynamic and changing domain. The development in the soccer simulation league during the previous years, which we will summarize in more detail in Section 2, has created circumstances that must be described as unique in the realm of RoboCup. For a period of five consecutive years, the software platform used in the soccer simulation 2D league remained unchanged. As a consequence, it became possible for the first time to thoroughly and critically evaluate the development and progress of an entire RoboCup league over a period of five successive years. We present the results of a comprehensive analysis concerning these issues in Section 3. Moreover, given the fact that the competitive character of RoboCup competitions is nearly identical in all leagues, we may infer that the progress to be observed in the simulation league can also serve as a showcase for other RoboCup leagues even if their characteristics disallow for a quantitative comparison. Apart from these considerations, we also look at the current character of the competitions in the 2D league and analyze whether saturation in the level of performance has been reached by the teams (Section 4).

2 Evolution of Soccer Simulation

In its early years, the simulation software of the soccer simulation league (Soccer Server) underwent significant changes and extensions. The simulation grew more complex from year to year, e.g. tacklings were introduced, heterogeneous player types were created, players were equipped with a simulated arm and neck, to name just a few of the enhancements (see Figure 1 for an overview).

On the one hand, these year-to-year changes were intended to make the simulation more complex and, eventually, more realistic. Moreover, yearly revisions kept up the pressure on all participating teams, so as to keep pace with the changes introduced. On the other hand, any modification to the simulation raised and still raises the question of backward compatibility. Will team binaries from a previous year still be usable for the current or yet-to-come Soccer Server version? Clearly, when certain features are removed from the simulation from one year to the next, then it is likely that older teams will malfunction given that they

relied on the feature removed. If existing parts of the simulation are altered, the impact on older team binaries will be less drastic, although they are in general expected to be impaired and play worse than they did with the previous Soccer Server version. Finally, the addition of entirely new features to the simulation is least critical. However, assuming that the use of a newly available feature (e.g. the use of heterogeneous players, introduced in 2001) can be beneficial with respect to team performance, a team making use of the new feature is supposed to outperform an older binary not aware of the innovation. In essence, by having annually modified the simulation environment, a meaningful comparison across teams from different years was impossible.

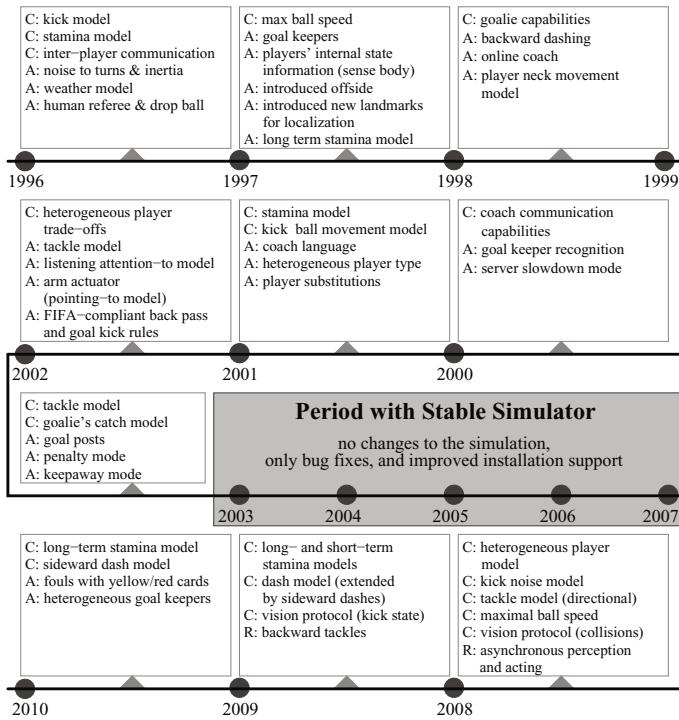


Fig. 1. Evolution of the Soccer Server: The 2D soccer simulation software has undergone various revisions over the years (**Changes**, **Additions**, and **Removals** of features)

Besides the issues of simulator backward compatibility and cross-year comparisons of soccer simulation teams, the question on the long-term perspective of the simulation league was raised during the RoboCup Symposium 2002. To this end, it was envisioned to extend the existing 2D simulation towards a 3D scenario as well as to more realistic robot simulations. After RoboCup 2002, including the mentioned symposium milestone discussion [3], the proposal came up to freeze the 2D soccer simulator and to retain it as a testbed for multi-agent research.

In particular, the goal was to keep the simulated soccer environment stable in order to facilitate measuring scientific progress from year to year. Moreover, the idea was to start developing a new 3D simulator from scratch that should be based on correct physical modelling in a three-dimensional environment.

With almost eight years having passed since then, we can view the following historic developments in retrospect:

- First work on a 3D simulator began in August 2002 (as mailing list archives confirm¹) and the first RoboCup tournament of the newly founded 3D-subleague took place in 2004.
- The transition towards simulating humanoid robots was accomplished in 2007, though this meant abandoning playing 11 versus 11. In fact, as of now there exist multiple 3D robot simulators in parallel [4].
- The 2D simulator remained stable from 2003 onward until the beginning of 2008. Although the number of teams allowed to participate in the annual 2D competitions at RoboCup was strongly restricted, the interest in 2D soccer simulation did not decline.

In 2008, the freezing policy was finally abandoned and, once again, significant changes were introduced into the 2D simulation in that year. There were two main reasons for this change of policy:

- The 3D league had started off adopting agents in the form of spheres allowing for access to abstract commands like kicking, turning, or dashing, not unsimilar to those known from the 2D simulation. With the above-mentioned transition to modelling humanoid robot models in 2007, the 3D league entered a lower level where the simulated robots had to be programmed in terms of controlling motor torques. As a consequence, the performance level of the games dropped significantly [4] as well as the number of players per team, that could be simulated simultaneously, did decline rapidly (2 vs. 2). Thus, at this point, the 2D simulation again represented the only branch of RoboCup focussing on issues like multi-agent coordination, team-play, and strategic reasoning.
- Given the special role resulting from the previous point, there was a growing demand within the simulation community to further develop the 2D simulator. Among other things, it was aimed at making the 2D simulation resemble real soccer, at least from a bird's eye perspective, as much as possible.
- As pointed out, the soccer simulation 2D league had experienced a period of stableness lasting five consecutive years. However, contrary to the original intention, year-to-year progress was not measured in the time window in which the 2D soccer simulation represented a stable multi-agent testbed.

With the paper at hand, we aim at filling the gap mentioned last. Since the policy of keeping a stable 2D simulator was irreversibly revoked in 2008, we consider it important to grasp the opportunity of performing a meaningful evaluation (at least covering the stable period visualized in Figure 1) and prove the progress of this league quantitatively, not just qualitatively.

¹ http://sourceforge.net/mailarchive/forum.php?forum_name=sserver-three-d

3 On Progress and Continuity

In human soccer playing, it is impossible to make sound statements about the development of the playing performance level over years. Although many experts will certainly agree that contemporary soccer exhibits more speed, dynamism, and superior athleticism than a few decades ago, empirical proof on its superior performance is inconceivable.

3.1 RoboCup Leagues' Further Development

With respect to long-term comparisons, one might argue that the situation is better in robot soccer. Robots are patient, can be stored away after a competition, and be unpacked after a few years in order to compete against a contemporary team. This is a nice thought experiment, but experience in RoboCup has proven that such an approach is infeasible. Hence, a formal analysis of the progress of soccer-playing robots made throughout the years is difficult to establish. Nevertheless, most researchers who have been involved in RoboCup activities over several years will unanimously agree that the progress made in all hardware-oriented RoboCup soccer leagues is not to be questioned. Despite this, assessments concerning the exact year-to-year progress remain qualitative in nature.

RoboCup's simulation leagues adopt a special role. No hardware development and maintenance is required and software agents do not age. However, the performance of soccer-playing software agents strongly relies on the simulation of the physical world they are placed in. If the characteristics of that simulation change, a meaningful evaluation of the progress made is rendered impossible. As delineated (Section 2), the simulation league has experienced a highly dynamic history, where the simulators used have undergone tremendous changes over the years. So, statements about the general level of playing performance frequently remained vague and on a qualitative level, without empirical verification. For example, already in 2002 it was claimed that the "overall playing strength of the teams in the tournament was quite impressive" [3]. Moreover, "the playing level of the tournament showed increased and consistent improvement as compared to last year's tournament". Similarly, "the 2003 tournament again showed a big advance in the performance of the teams" [5] and the "level of play of the last twelve teams this year was very mature and close to each other" [6].

While we subscribe to these assessments, we need to stress that no empirical proof for their correctness exists and, in fact, cannot exist. As shown in Figure 1, from 2001 to 2002 and 2002 to 2003 several changes were introduced into the 2D soccer simulation, which is why a meaningful evaluation of the objective progress made in those days is infeasible. By contrast, our focus is on the subsequent period of stability (2003-2007). The results of a large-scale analysis of the progress made during that time interval shall be presented in the next section.

3.2 Empirical Proof of Progress

Platform. As pointed out, of crucial importance to a meaningful comparison is the use of a simulation platform that is equally supported by all teams under

consideration. During the stable period sketched in Figure 1, four different versions of the Soccer Server were used for the RoboCup world championship tournaments in the respective years. Though differing in their version numbers and deviating slightly in platform support and ease of installation, the simulation is entirely identical and allows all team binaries published during that time to work properly with any of the Soccer Server versions released in those years. Therefore, we conducted all empirical analyses using the 2007 version 11.1.0 of the Soccer Server, as this version is fully downward compatible and represents the one that is expected to have applied all relevant bug fixes.

Team Selection. Our aim has been to assess the progress made over the years in soccer simulation 2D. We are convinced that a year’s level of play can be validly read from the performance of that season’s top teams, i.e. the teams placing best during a year’s RoboCup world championships. We refrain, however, from saying that a year’s champion alone is a reliable representative of its volume’s strength. For these reasons, we decided to have each of the years from 2003 to 2007 be represented by all teams that made it into the respective year’s semi-finals (top four teams). In reporting results, we explicitly do not provide the exact team names, but use identifiers of the form “year_place” (e.g. 2007_2 to indicate the runner-up of RoboCup 2007) for the sake of better interpretability. A matching from these identifiers to team names is given in the Appendix.

Experiments. We retrieved the original top four binaries from the five consecutive years under consideration and prepared them for use in the scope of an extensive evaluation involving more than 4000 regular matches. In order to ensure identical prerequisites, we conducted all matches consecutively on exactly the same hardware and used Soccer Sercer version 11.1.0. For the league progress analysis, we let each team face each other team fifteen times in a standard match of 6000 simulation cycles length.

Results. In the top right part of Figure 2, we can see how well each team performed in this comprehensive evaluation. The bar chart shows the share of points each team carved out, when repeatedly facing any of the other 19 teams involved in this study. As usual, a team is awarded 3 points for a victory, 1 for a draw, and 0 for a defeat, thus 100% would indicate winning each match, whereas 33.3% would be equivalent to drawing each match. The 2007 champion obviously comes off best in this regard with a share of 86.0%. Moreover, we can observe that older teams (darker shades of gray) are placed in the second half of this score board-like visualization.

The main part of Figure 2 focuses on the scores (each data point is averaged over 270 matches) the teams yielded when repeatedly playing against all other teams considered in this evaluation. By connecting the data points belonging to representatives of the same year, the steady increase in performance becomes clear. Apparently, the polygons formed are shifted from the chart’s top left to its bottom right region with increasing years (where the average number of goals scored is larger than the number of goals received).

While the previous analysis has focused on individual teams, Figure 3 examines the performance level of different years against one another. Here, we let

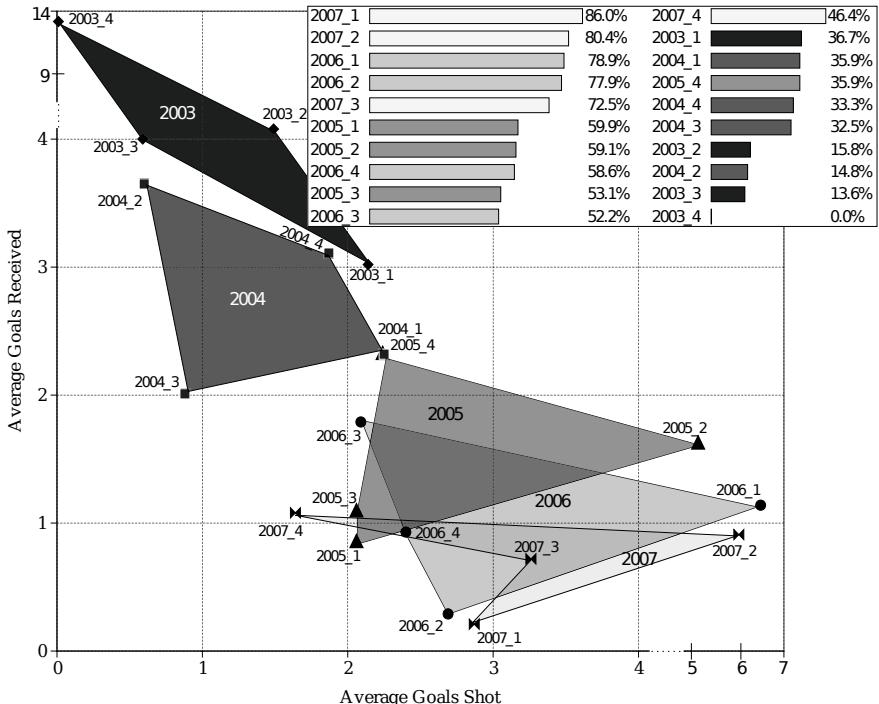


Fig. 2. Top right: Relative strengths of the 20 teams considered in this evaluative study. Main: Average scores yielded by each of these representatives, when repeatedly playing against one another.

the representatives from each season play 240 matches each against the representatives from any other year. The matrix-like result presentation in that figure highlights the fact that in any of the 10 combinations, the season listed in the row was inferior to the column season. Generally, when comparing year x and year y (with $x, y \in \{2003, \dots, 2007\}$), it holds that the difference in the number of points achieved against one another as well as the difference in the average scores becomes more pronounced the larger $|x - y|$.

In addition to the year to year-based comparisons, Figure 4 summarizes the progress analysis of the stable period by visualizing the joint performance of representatives from the same year playing against teams from all other seasons. As a matter of fact, the share of matches won is increasing continuously over the years, while the share of games lost is dropping at the same time. For example, the 2007 representatives lost only 12.9% of their matches against binaries from all recent years. The chart to the right in Figure 4 shows that a similar trend holds for the average scores resulting from this study and, thus, provides further empirical proof for the verifiable progress of soccer simulation 2D made from 2003 to 2007.

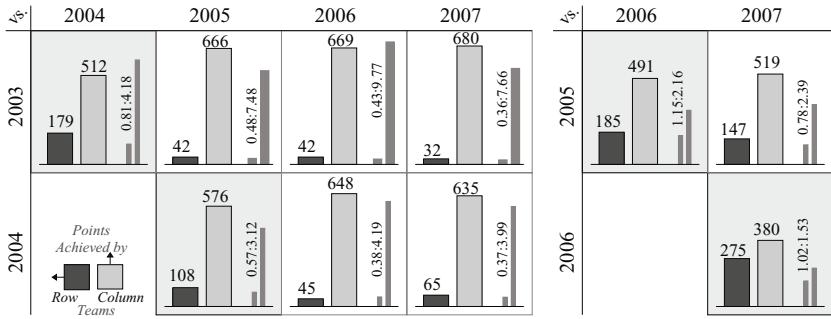


Fig. 3. Representatives from each year of the stable period (2003-2007) played 240 matches against representatives from any other year. This figure summarizes the numbers of points (maximum: 720) as well as average scores yielded in this experiment.

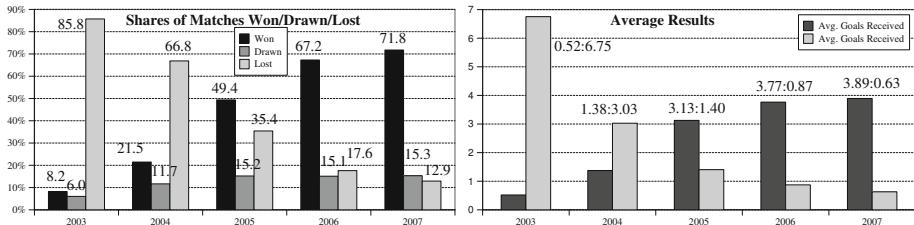


Fig. 4. Joint performance of representatives from the same year when playing against teams from all other seasons: Both average results (over 2400 matches) as well as the development of the shares of matches won/drawn/lost hint to the fact that substantial progress in playing performance had been made

4 On the Character of 2D Competitions

With the development delineated, the question arises whether we can observe some kind of saturation in the further development of the 2D league teams' performance and whether we can see the top teams playing at approximately the same level.

A first answer to this question can be read from the charts in Figure 3. Teams from a successor season do always have a clear edge over the preceding year's teams. While 2004 teams would lose 20.8% of the matches against 2003 representatives and teams from 2005 only 14.7% against 2004 representatives, this number increased slightly to 19.6% as well as 29.2% in 2006 and 2007, respectively. The progress made from year to year, however, has slowed down recently as can be read from the average scores (e.g. 1.02:1.53 for 2006 vs. 2007 teams). A second answer to this question relates once again to the issues mentioned in Section 2. With the re-initiated introduction of changes and extensions to the simulator in 2008, any form of convergence became basically infeasible, since the modifications of the simulation forced the teams to reimplement considerable

parts of their teams or to add functionality that may turn out to be crucial for becoming or staying competitive. There are, however, two further answers to the above question, both of which we will discuss and underpin empirically in the following sections.

4.1 Randomness and Repetability

The question on convergence of performance is closely related to a question that is frequently brought up by spectators during competitions, especially, when it comes to the final matches, “If you replayed this match, would the result be the same?” Of course, the general answer is “no”. But, if we could observe something like convergence in the performance development of the teams, then this “no” should become evidently more pronounced over the years.

Experiments. In order to investigate this question, we focused on the final matches of the seven recent RoboCup tournaments (2003-2009). Under the same conditions as the experiments in Section 3, we let the final matches of those years be replayed for 100 times each, calculating average results as well as corresponding standard deviations (outcomes are plotted in Figure 5).

Results. This series of 700 matches reveals two interesting facts:

- An apparent (and appreciative) observation is that in any year (since 2003) the 2D simulation league found a worthy champion. The analysis shows that, from the perspective of the winning team, the average score over many repetitions of the final match is $x : y$ with $x > y$ for all years (in fact, $\min_{2003,\dots,2009} x - y = 0.31$). Also, it never occurred that the probability for the runner-up to win a match against the champion was higher than the probability for the champion to win. To this end, the closest situation was in 2006, where the runner-up would have even won one out of three matches.
- There is no clear trend of converging performance (i.e. with more and more equally well-performing final match participants), not even during the stable period. At best, one may infer that in 2003 and 2004 the domination of the champion team used to be more pronounced than in later years and that the total number of goals scored in the final had been dropping until 2008. Thus, the development observed does not allow us to conclude that a state of saturation had or has been reached.

4.2 The Effect of “Minor” Changes: A RoboCup 2009 Example

Another indicator for saturation of progress is that significant improvements to a team’s chances of winning the next game can be realized only by investing an exponential amount of effort and would, thus, be impossible to accomplish during competitions. In RoboCup 2009, our team (Brainstormers) experienced a textbook counter example which is unrelated to any recent simulator changes and, hence, descriptive for the state of the league.

Initial Situation. Our team’s defense strategy strongly relies on an effective man-to-man marking. The exact matching from our players to their direct, i.e. to

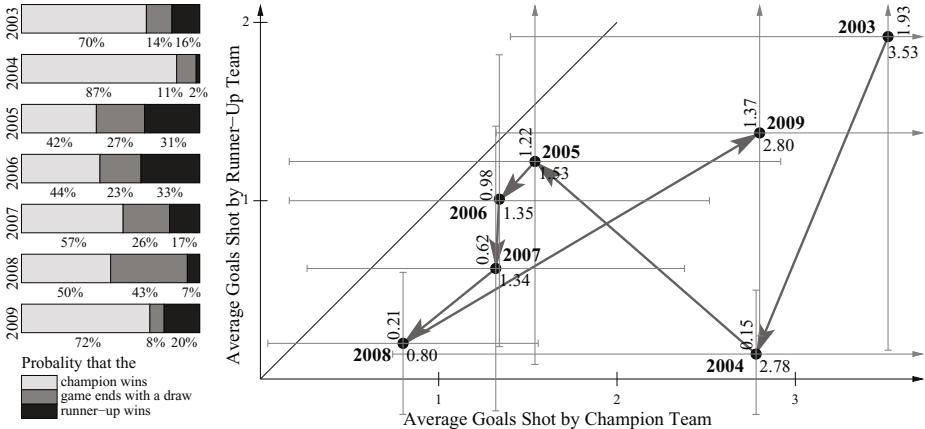


Fig. 5. Replaying the final matches from 2003-2009: Although no clear trend is manifested, in each year the superior team has won the World Champion title

be marked, opponents are computed by the coach program based on an opponent player role determination, using gliding position windows with exponential weighting as well as a greedy graph-matching algorithm. Having determined an appropriate man-to-man marking, the coach communicates the assignment of direct opponents to the players on demand.

Having a look at the distribution of results that is created when the assignment mechanism works properly, we can see (Figure 6, left column) that our team would win/draw/lose 30/30/40% of the matches against the 2009 champion WrightEagle and 32/29/39% against Oxsy (3rd place). Taking into account the average results and their respective standard deviations plotted in that figure, it is obvious that the starting situation could be described as head to head.

Preparation for RoboCup 2009. WrightEagle decided to develop a player role-changing mechanism that turned out to make the described man-to-man marking get out of step. Although these role changes were detrimental to WrightEagle's own performance (e.g. because a defender now had to act as an attacker) and they increased the number of goals received from 1.01 ± 0.975 to 1.45 ± 2.06 , the damage to the Brainstormers' defense strategy was more pronounced. The number of goals shot by WrightEagle more than doubled as can be read from Figure 6 (top row, middle column). Consequently, when both teams faced each other at the semi-final the match ended 0:4.

On-Site Modifications. After having observed and analyzed the above-mentioned semi-final, team Oxsy copied WrightEagle's role-changing strategy in preparation for the matches on the following day. Simultaneously, our team took counter measures and improved the man-to-man marking determination. The effect of Oxsy's overnight clone of WrightEagle's role-changing strategy can be read from Figure 6 (bottom row, middle column). Apparently, applying the mentioned strategy blended very well with Oxsy's general way of playing, which is

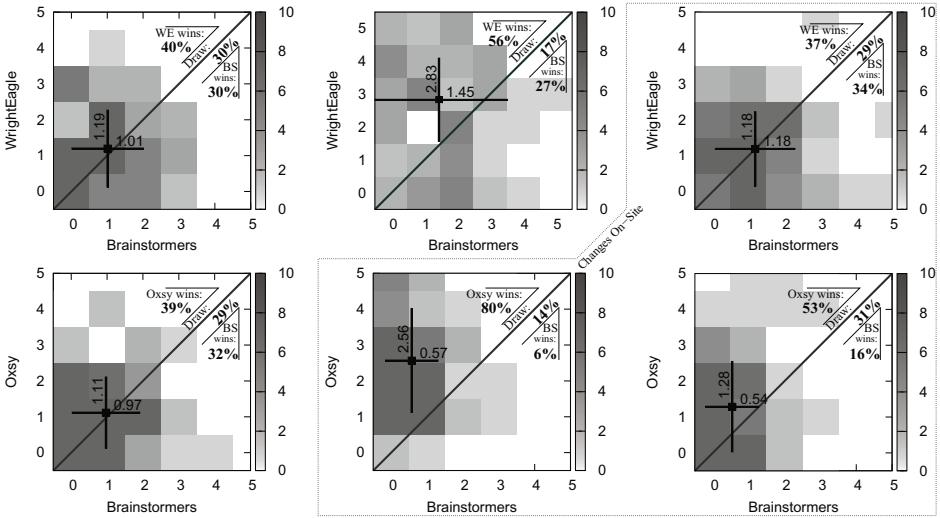


Fig. 6. Average goals shot and received with corresponding standard deviations as well as result distributions of matches between the Brainstormers and WrightEagle (top) and Oxsy (bottom) during RoboCup 2009. The initial situation (left column) is contrasted with the impact of a single winning feature added by WrightEagle and Oxsy (middle column) and counter measures taken by the Brainstormers (right column), with the latter realized during the competition.

why the impact on the average result yielded against the Brainstormers was substantial (improvement from 0.97:1.11 to 0.57:2.56 on average). By contrast, the impact of our team's counter measures (also implemented overnight), which happened to play no further role during the rest of the tournament, are visualized in the right column of Figure 6. Apparently, the impact of the role-changing strategy could be fully annihilated when playing against WrightEagle, and partially when playing against Oxsy.

While the example outlined in this section represents only a snapshot from last year's RoboCup tournament, it serves as evidence to support the claim that decisive and potentially winning modifications to a 2D soccer simulation team's strategy can still be achieved in little time and, in particular, even on-site. This fact is not only a counter example against the saturation hypothesis, but it also stresses the exciting character of the competitions.

5 Conclusion

In this paper, we have targeted questions that relate to the progress made in RoboCup across years. While qualitative assessments of the further development exist in abundance for many leagues, in general it is impossible to make distinctive and verifiable measurements of the playing performance between participants of different years. Starting with the observation that recent circumstances in the

soccer simulation 2D league allowed us to perform a meaningful evaluation, we presented the results of an extensive study that, in a retrospective manner, investigated the further development of the playing strength in soccer simulation.

We found that the progress made during the so-called “stable period” (2003–2007) is astonishing. While admired for their sophisticated play in 2003 and 2004 [5], world champions of those times played at the level of a low-class team a few years later, sometimes losing in the double digits against top teams from 2006 or 2007. In an additional analysis, we found that up to now no saturation has been reached in the level of play of the teams competing in the soccer simulation 2D league. In particular, we found that even minor changes or extensions to a team’s repertoire of capabilities, i.e. modifications that can be realized even on-site during competitions, may suffice to bring about significant and game-winning advantages. Needless to say this feature strongly contributes to the exciting character of the competitions in this league.

Acknowledgements. The authors would like to thank Andreas Hechenblaickner for providing the HLM League Manager sources as well as Hidehisa Akiyama, Ke Shi, and C. Fan for their help in retrieving and refloating old team binaries.

References

1. Noda, I.: Soccer Server: A Simulator of RoboCup. In: Proceedings of the AI Symposium, Japanese Society for Artificial Intelligence, pp. 29–34 (1995)
2. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer Server: A Tool for Research on Multi-Agent Systems. Applied Artificial Intelligence 12, 233–250 (1998)
3. Obst, O.: Simulation League – League Summary. In: Kaminka, G., Lima, P., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 443–452. Springer, Heidelberg (2003)
4. Kalyanakrishnan, S., Hester, T., Quinlan, M., Bentor, Y., Stone, P.: Three Humanoid Soccer Platforms: Comparison and Synthesis. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 140–152. Springer, Heidelberg (2010)
5. Pagello, E., Menegatti, E., Bredenfeld, A., Costa, P., Christaller, T., Jacoff, A., Polani, D., Riedmiller, M., Saffiotti, A., Sklar, E., Tomoichi, T.: RoboCup-2003. New Scientific and Technical Advances. AI Magazine 25(2), 81–98 (2004)
6. Pagello, E., Menegatti, E., Bredenfeld, A., Costa, P., Christaller, T., Jacoff, A., Johnson, J., Riedmiller, M., Saffiotti, A., Tomoichi, T.: Overview of RoboCup 2003 Competition and Conferences. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 1–14. Springer, Heidelberg (2004)

Appendix

	1	2	3	4
2003	UvATrilearn (NED)	TsinghuaAeolus (CHN)	Brainstormers (GER)	Everest (CHN)
2004	STEP (RUS)	Brainstormers (GER)	Mersad (IRN)	TsinghuaAeolus (CHN)
2005	Brainstormers (GER)	WrightEagle (CHN)	TokyoTech (JPN)	STEP (RUS)
2006	WrightEagle (CHN)	Brainstormers (GER)	Ri-One (JPN)	TokyoTech (JPN)
2007	Brainstormers (GER)	WrightEagle (CHN)	HELIOS (JPN)	OPU-Hana (JPN)
2008	Brainstormers (GER)	WrightEagle (CHN)	HELIOS (JPN)	AmoyNQ (CHN)
2009	WrightEagle (CHN)	HELIOS (JPN)	Oxsy (ROM)	Brainstormers (GER)

Odometry Correction for Humanoid Robots Using Optical Sensors

Stefan Czarnetzki, Maximilian Hegele, and Sören Kerner

Robotics Research Institute
Section Information Technology
TU Dortmund University
44221 Dortmund, Germany

Abstract. Odometry measurement is an important concept to update localization information, but is prone to error propagation. Still the method is widely applied to wheeled mobile robots since their motion is quite robust to random error such as slipping. While the concept of odometry can also be applied to humanoid robots the dynamically stable walking generation reinforces sliding motions resulting in unpredictable errors. Therefore this paper proposes a novel approach to measure these sliding errors with the help of optical sensors to either correct the odometry update or perform suitable actions to counteract the error.

1 Introduction

With the development of more capable robotics hardware autonomous robots become able to solve more complex tasks. This opens up new possible application areas such as rescue operations or service activities. To fulfill tasks in these areas of application, robots must be capable of navigating in environments designed for humans or rough terrain. Those environments are particularly challenging for the movement of conventional wheeled autonomous robots. Normal stairs or small objects lying on the floor become insurmountable barriers. For these reasons modern robotic design tries to mimic human appearance with respect to body design, capability of gestures and facial expressions [1]. As a consequence humanoid robots are one of the major topics of robotics research and are believed to have high potential in future applications.

To solve given tasks a mobile robot is required to navigate in known or unknown environments. This kind of goal orientated movement requires some methods of localization to apply path planning algorithms [2]. The planning algorithm needs a global initial position as an input which can be calculated with the help of a wide range of external sensors, such as laser range finders or cameras. In addition the algorithm needs localization updates during the motion execution to supervise the executed path. This information can either be global or differential local updates relative to the last known robot position. Filter algorithms, such as Kalman [2], can be applied to update the current position using the initial position and the differential sensor update. While a differential position update can be derived from every form of global sensor information most global

localization methods are not accurate enough to be useful to calculate small differential updates. Therefore a common concept is to update the current position with the help of movement information. Such motion can either be theoretically calculated by using the actuator input or by measuring actuator output with the help of internal sensors. This concept is called *odometry*.

The next section gives an overview of research on odometry and related work with special focus on possible odometry update errors. In the following sections a concept is proposed to measure such errors and is tested in several experimental setups.

2 Odometry

Using actuator movement to estimate the current position is a common concept in mobile robotics. The position change can either be calculated using the given motor commands or measured with the help of sensors. While in an ideal world both methods would yield the same results they differ in reality due to deviations in motion execution. Thus utilizing the measured motion update is more error-proof and hence preferable. To update the current position sensors can measure either the current speed of the robot or the executed actuator motion. While both principles have been successfully applied to wheeled robots in the past, they can not be adapted to humanoid robots easily.

Acceleration sensors allow the tracking of the current speed of the robot. As long as the robot has no need to move its body to achieve postural stability, which is true when utilizing at least three wheels, the body can be kept in an upright position. Hence for wheeled robots with a rigidly fixed connection of wheels all measured acceleration relates to a position change. Movement of a humanoid robot is different in contrast to a rolling robot. The nature of legged walking requires some movement of actuators that does not result in a direct position change of the robot. While slow movement, which satisfies the static stability criterion, ensures a rather stable position of the upper body, fast movement, which requires dynamic stability, results in a movement of the robot's *Center of Mass* (CoM) leading to a periodic motion of the upper body. While in theory integration over all acceleration changes would result in the change of position, this would require the knowledge of the body orientation in space. Since the orientation of the body can only be estimated and walking movements result in a rather swaying movement, the odometry estimation with help of an accelerometer is not practical for humanoid robots.

The second principle of odometry measurement is to update the position with the help of actuator movements. Since rolling robots are supposed to have ground contact with all wheels at all the time every actuator movement relates directly to a position change of the robot. With the help of wheel encoders calculating the resulting movement is therefore straight forward [3] if the kinematic structure of the wheel positions is known. The walking motion of a two-legged robot requires to periodically lift one leg, resulting in a change of double- and single-support phase [3]. Hence not all motion during the single support phase will result directly

in a change of position, but will alter the support polygon of the robot. Only a shift of the CoM relative to the support polygon leads to a position update of the robot. Thus the combination of the CoM motion relative to the support polygon and expanding the support polygon whenever a single support phase merges into a double support phase results in the movement of the robot during a step. With most legged robots utilizing servo motors which allow to get a position feedback these sensors can thereby be used to calculate the odometry update during the walk resulting in a practical way to measure the relative movement of a humanoid robot.

3 Odometry Error

Due to the incremental nature of odometry this localization method is susceptible to error propagation. A false motion update influences the starting position of the following updates and therefore sums up over time. Errors in odometry measurements can be divided in two types of errors. The first being systematical measurement error while the second arises of actuator movement not resulting in a position update of the robot. Systematical measurement errors can be eliminated by correction or calibration of the sensor information. But predicting when errors of the second class will occur during motion execution is not easily done. Thus they can only be classified as statistical errors. Since every odometry error effects the following prediction update even non-systematical errors can result in a big position estimation error over time. This is especially true for errors in orientation, resulting from incorrectly measured rotational motion, since these sum up to errors in x- and y-position over time [4]. The wheels of a rolling robot are designed to maximize ground friction in order to move. Thus a wheeled robot seldom violates the sliding constraint reducing the impact of the second error class. With good calibration odometry updates for wheeled robots can therefore be seen as quite accurate resulting in a good practical way to update the localization while moving.

While legged robots also try to achieve enough friction during the ground contact to move the robot the impact of the second class of errors becomes more severe. If the friction of the foot is too high slight deviations in putting the foot on the ground can cause a jolt resulting in a fall of the robot. Humanoid robots tend to use the concept of dynamical stability to generate fast walking motions [5]. Thus the material of the robot sole is chosen to be able to slide on the ground when not the complete weight of the robot is supported by this foot. The concept is based on keeping the *Zero Moment Point* (ZMP), which is not taking into account the moments around the vertical axis, inside the support polygon. With faster walking speeds this requires high acceleration of the actuators resulting in forces acting on the robot. When these forces exceed the force the ground friction can hold this results in a sliding motion of the robot. A good illustration is the movement of the swinging foot during the single support phase. The acceleration needed to place the foot on the desired ground position results in a torsional moment around the support foot rotating the robot

around the vertical axis. Since the amount of rotational movement depends on many factors such as the local ground characteristics and the current robot pose the rotation can neither be calculated or predicted and thereby leads to odometry errors of the second class resulting in a deviation in position over a couple of steps. Since the sensor data of both accelerometer and gyroscope are prone to noise due to the swaying walking movement the robot has no sensors capable of measuring this position update. Therefore this paper proposes a novel approach to measure the slipping movement of a humanoid robot described in the following section.

4 Measurement of Odometry Errors

The described motion errors have a great influence on odometry measurement of biped robots. Therefore a way to measure these errors would greatly improve the quality of odometry updates. Since none of the internal sensors is capable of measuring the deviation resulting of the sliding movement the need to apply new sensors arises. The motion that should be measured is the sliding movement of the support foot. This foot is supposed to have contact to the ground during the complete single support phase and should be lifted after the following double support phase. During this single support phase every movement of this foot relative to the floor directly results in a position change of the robot.

Sensors used for optical mice are designed to measure motion updates relative to the ground. The sensor is tracking the position update of the sliding motion while the mouse bottom side has flat contact to the ground. In the past experiments with optical sensors have proven them to be able to support the update of odometry information of rolling mobile robots [6], but since the optical mouse is designed to have ground contact such a sensor either limits the ground clearance of the robot or requires an adapted optical lens system. Modern laser sensors are powerful enough to track both high acceleration and speed on most surfaces. New technologies like Microsoft's *BlueTrack*¹ and Logitech's *DarkField Laser Tracking*² sensor are even capable of tracking movement on uneven and reflecting surfaces such as glass. Since the ZMP, and therefore the *Center of Pressure* (CoP) [5], is supposed to be beneath the standing foot during the single support phase, the foot will be pressed flat to the ground. Hence an optical sensor positioned under the foot surface should be able to measure the motion of the foot during the support phase, if the ground is planar.

A mouse sensor provides a motion update in x- and y-direction by incrementally adding up *counts*. Therefore with a given start position $S(x, y)$ and a measured end position $S'(x', y')$ it cannot be distinguished if the update resulted from a pure translation, a pure rotation or a combination of both. Since a single mouse sensor would not be sufficient to also track rotation the need to combine two sensors arises [6].

¹ <http://www.microsoft.com/hardware/mouseandkeyboard/tracklanding.mspx>

² http://www.logitech.com/images/pdf/briefs/Logitech_Darkfield_Innovation_Brief_2009.pdf

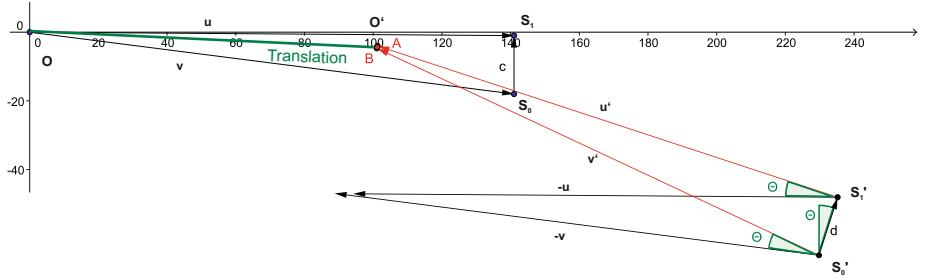


Fig. 1. Foot movement(translation = 100 mm, rotation = 20 deg)

Figure 1 displays example sensor positions $S_0(x_0, y_0)$ and $S_1(x_1, y_1)$ before and after a motion that includes translation and rotation. Since the foot is assumed to be flat on the ground, the z-coordinate is neglected. The position of the sensors in reference to center of the foot coordinate system, labeled in figure 1 as O , fixed and therefore the positions in the robot coordinate system can be calculated by the help of inverse kinematics. The updated sensor positions $S'_0(x'_0, y'_0)$ and $S'_1(x'_1, y'_1)$ after the motion in the old foot coordinate system can be derived by adding the measured sensor motion ΔS_0 and ΔS_1 .

With the new sensor positions the foot rotation can be deduced by calculating the angle θ between the vectors \mathbf{c} and \mathbf{d} .

$$\Theta = \arccos \left(\frac{\mathbf{c} \cdot \mathbf{d}}{|\mathbf{c}| |\mathbf{d}|} \right) \quad (1)$$

The rotation sign results from calculating the cross product of \mathbf{c} and \mathbf{d} .

$$\mathbf{c} \times \mathbf{d} = \begin{pmatrix} c_2 d_3 - c_3 d_2 \\ c_3 d_1 - c_1 d_3 \\ c_1 d_2 - c_2 d_1 \end{pmatrix} \quad (2)$$

The vectors \mathbf{u} and \mathbf{v} are the given position of the Sensors S_0 and S_1 in the foot base coordinate system. Transforming these with the help of the rotation matrix for the calculated rotation Θ results in the vectors \mathbf{u}' and \mathbf{v}' .

$$\mathbf{u}' = \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \mathbf{u} \quad (3)$$

$$\mathbf{v}' = \begin{pmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{pmatrix} \mathbf{v} \quad (4)$$

Subtracting these vectors from the new sensor positions S'_0 and S'_1 , respectively, results in the new center of the foot coordinate system O'

$$S'_0 - v' = O'_v \quad (5)$$

$$S'_1 - u' = O'_u \quad (6)$$

The resulting \mathbf{O}'_v and \mathbf{O}'_u are theoretically identical. In reality both will differ due to errors in sensor output. The new center of the foot coordinate system will therefore be arithmetically averaged from both sensors.

$$\mathbf{O}' = \frac{\mathbf{O}'_u + \mathbf{O}'_v}{2} \quad (7)$$

The foot translation \mathbf{T} is calculated by subtracting the old from the new foot coordinate system center.

$$\mathbf{T} = \mathbf{O}' - \mathbf{O} \quad (8)$$

Combining the rotation θ and translation \mathbf{T} transforms the old foot pose p in the new foot pose p' .

$$p' = R(\theta) \cdot p + \mathbf{T} \quad (9)$$

5 Evaluation

To evaluate the concept presented in this paper, experiments were conducted using a robot of the type *Nao* designed and build by Aldebaran Robotics³. The robot has no optical sensors integrated in the feet and therefore has to be equipped with such a setup to perform the experiment. From the wide variety of optical sensors the ADNS-5030 produced by *Avago* is chosen, being specially designed for the use in wireless optical mice. Thus the rather low energy intake of 15.2 mA making it ideal for usage in a mobile system without stressing the power supply. The sensor has a resolution of 2000 Dots per Inch (DPI) which is high enough to measure even small motions. Initial performance tests have been done with the help of a robotic manipulator. The results of the experiment prove the sensor to perform well on different surfaces, such as paper, carpet, concrete, linoleum and laminate flooring. The measurement accuracy is not affected by speeds up to 1.1 m/s and accelerations up to 0.9 m/s. All used sensors are individually calibrated by the experimental outcome of these tests as a reference to match the measured counts to the moved distance. When lifted above 0.3 mm off the ground the sensor did not measure any motion while moving, assuring that the sensor of the swinging foot will measure any motion. In this situation this is desirable since the measurement would otherwise incorrectly affect the motion update. Due to the tests classifying the sensor performance to be adequate and the fact that the optical lens system of a laser sensor is much larger and therefore more complicated to integrate in a humanoid robot foot, the LED sensor ADNS-5030 is chosen over more powerful optical laser sensors.

The ADNS-5030 is utilized in a custom designed board. Since the robot *Nao* is not self build and its internal interface and layout is not published the sensor board can not be integrated in the foot and has therefore to be attached externally. This is done with the help of a kind of overshoe, attaching the board directly in front of the foot (see figure 2). The material of the shoe is chosen

³ <http://www.aldebaran-robotics.com/>

to reflect characteristics of the sole of *Nao*. The information collected by the sensors is transmitted to *Nao* with a wireless connections based on the *ZigBee*⁴ standard connected to the USB port making a transmission possible within the 20 ms time frame given by the used framework.

5.1 Odometry Sensor Setup

With reference to section 4 it becomes obvious that the position of the sensors influence the quality of the measured motion. Even if being highly accurate under ideal circumstances the sensor output will be flawed when used on none ideal surfaces by adding random noise to the measurement.

If the robot is turning it is logically supposed to rotate around the current CoP being the point which has currently the highest static friction. To ensure stability the CoP is preferably positioned in the middle of the foot. Hence positioning the sensors distant to the center will result in a bigger distance traveled by the sensor lessening the absolute influence of noise. But as can be seen in equation 1 the measured rotation is highly influenced by vector c which relates directly to the position of the sensors relative to each other. Therefore a first experiment will evaluate two different sensor setups. The first configuration, from here on denoted as *Configuration 1*, can be seen in figure 2(b) on the left. The two sensors are positioned within the same board placing them 17 mm apart from each other. In contrast an additional sensor board is attached to the heel of the robot, seen in figure 2(b) as *Configuration 2* on the right, resulting in a distance of 232 mm.

To prove the theoretical concept proposed in section 4 and to evaluate the influence sensor noise in respect to different sensor positions the robot is equipped with both sensor configurations and a combined translation and rotation is performed. The sliding motion is executed manually to ensure ground contact for both sensor setups during the complete motion. Three separate experiments are performed repeatedly, figures 3, 4 and 5 showing chosen representative results. Experiment 1 and 2 demonstrate motion being purely rotational respectively translational. Figure 3 illustrates clearly the impact of the sensor distances on the measured rotation. While the measured results of *Configuration 2* match the real rotation, the impact of the sensor noise on *Configuration 1* due to the short distance becomes obvious. The results of the second experiment (see figure 4) demonstrate the same tendency. While the translation is not directly influenced by the distance between the sensors the reason for the second configuration performing significantly better becomes clear when recapitulating the calculations demonstrated in section 4. To calculate the translation, the foot rotation has to be calculated before. Therefore the rotation noise influences the translation even when no rotation took place in reality.

The third experiment combines both motions, a rotation and a translation in x direction, to one sliding movement. The results of the rotation calculations are demonstrated in figure 5(a), the following translation calculation can be found

⁴ IEEE 802.15.4.

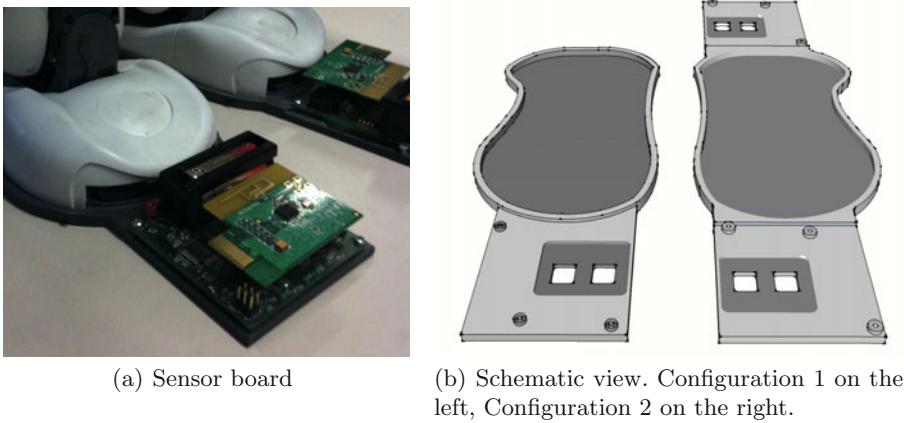


Fig. 2. Sensor placement

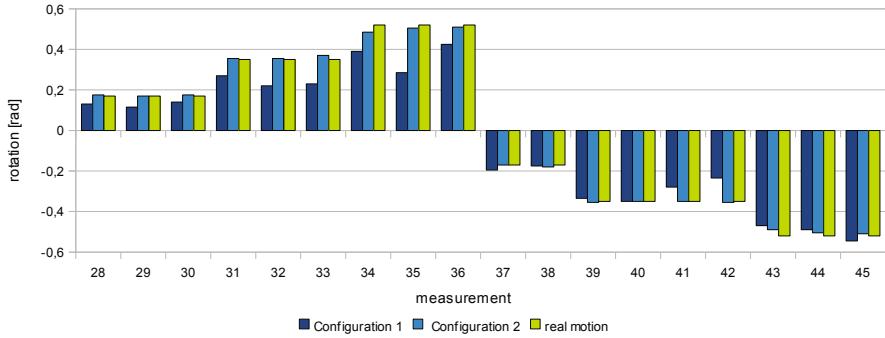
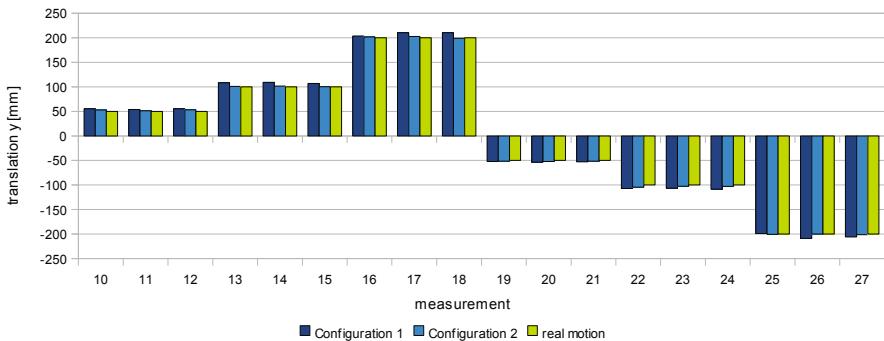
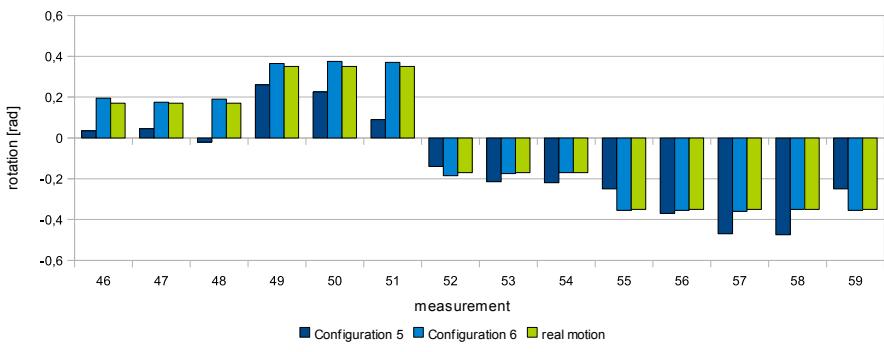


Fig. 3. Experiment 1 - Manual rotation

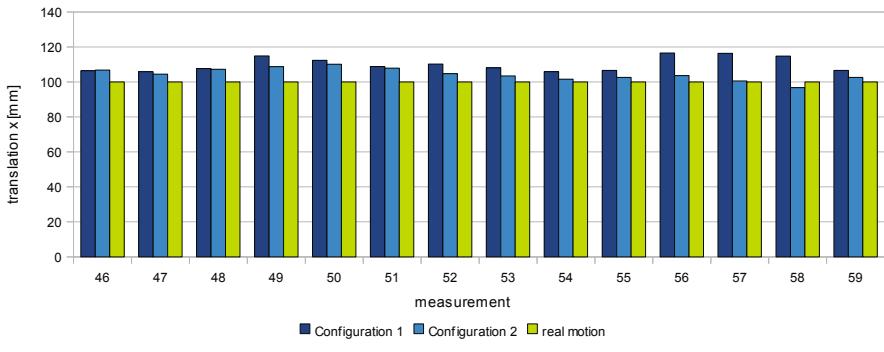
in figure 5(b). The outcome of the rotation measurement again puts the second configuration ahead of the first, but with a higher variation due to the noise interaction with the translation motion. As a result the translation also yields inferior results but still measures the motion to a satisfying accuracy.

5.2 Walking

Since the first experiment proves *Configuration 2* to be superior all following experiments only utilize this sensor setup. The described experiments support the conclusion that the sensor setup allows to measure the slipping motion, and thereby the odometry error, during a walk. Thus the last experiment, described in this paper, demonstrates the results of the sensor measurement of an autonomous walk. While the additional weight of the sensor boards influences the walking of *Nao* the parameters of the utilized walking engine [7] are adapted to ensure a stable motion.

**Fig. 4.** Experiment 2 - Translation in y-direction

(a) Rotation measurement



(b) Translation measurement in x-direction)

Fig. 5. Experiment 3 - Manual combined motion

The Measurement set number 86 of figure 7 is illustrated as a more detailed example walk in figure 6. As expected a rotation of the support foot can be observed during every single support phase, caused by the movement of the

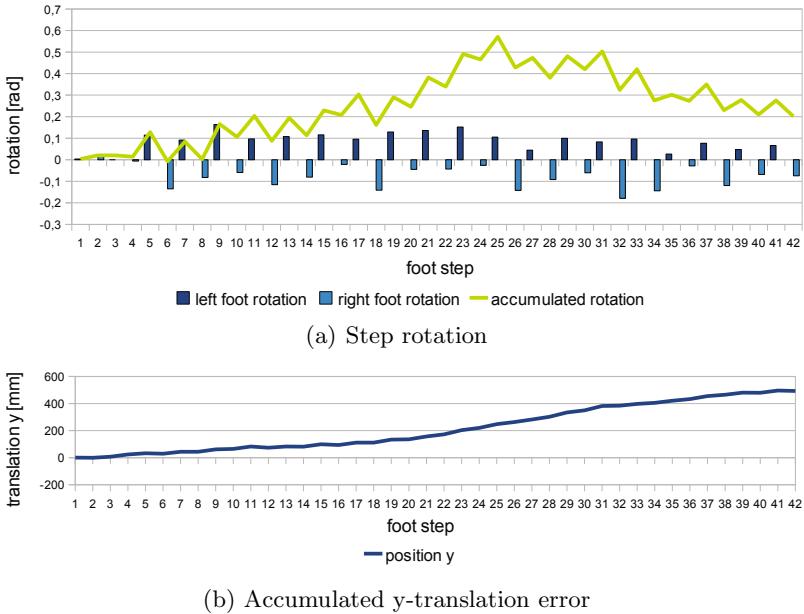


Fig. 6. Example walk in x direction

swinging leg. The direction of the rotation depends on which leg is swinging, but the amount of rotation differs on every step. Therefore the accumulated rotation increases in positive direction causing a deviation of the walk in y-direction which cannot be observed by the odometry. The measured deviation of 49.2 cm does not match the observed real deviation of 65.0 cm but clearly helps to correct the odometry update.

Figure 7 illustrates a representative sample measurement of the same walking experiment on different floor types. Most interesting is the analysis of the resulting y-direction error in figure 7(a). Since a validation of each step rotation is not possible due the lack of a reference measurement, the resulting deviation in y-direction over time of the walk is the best proof of the rotation measurement quality. As figure 7(a) illustrates every walking test results in of y-deviation over the executed time proving that the influence of the observed rotation is significant. The results of the sensor measurement, however, show a wide distribution with some results even measuring a drift in the opposite direction. This result is contrary to these of the previous experiments and therefore requires discussion. Due to the incremental nature of the experiment the source and influence of a measurement error cannot be derived from the outcome. A small error on the first rotation measurement can lead to a large drift over the course of the walk, while the same error made during the last step would have no influence at all. But the observation of the experiment allows some conclusions that explain the variance of results.

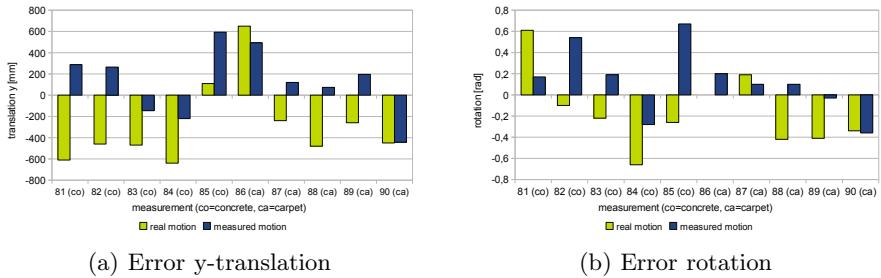


Fig. 7. Experiment 4 - Walking in positive x-direction with speed 100 mm/s

First of all as can be seen in figure 6(a) the occurring rotation during the walk can be located around 0.1 rad. Even though Experiment 1 has proven that the measured motion update for this dimension is satisfactory under ideal circumstances such small updates make the measurement more susceptible to sensor noise. The movement generated by the walking engine results in a much bumpier motion than the manual motion, resulting in less ideal conditions. Due to the utilized closed loop sensor control a countermeasure to assure stability can even result in a rocking motion lifting a part of the foot off the ground shifting the CoP and thereby the rotation center. This can result in one sensor being lifted higher off the ground than its height-tolerance resulting in a false rotation measurement. Since the rocking motion is worse when accelerating the robot from a standing position this explains part of the deviation.

6 Conclusion

The experiments prove that updating odometry of humanoid robots with information of optical sensors is a big advantage in principle. The sensors can measure motions unobservable by the conventional methods to measure odometry allowing an error correction of the motion update. This helps to improve especially local localization e.g. when trying to approach an object. An online compensation of the error by adaptation of the walking pattern could not be tested due to the variation of sensor output during autonomous walking.

The results of the experiments prove the importance of suitable sensor placement with placing the two sensors further apart being more favorable. But the observations also suggest that it is more beneficial to place the sensors closer to the CoP by moving them nearer to the center of the foot. This would result in less lifting of the sensor during the rocking walking motion and thereby would improve the measurement. Unfortunately this influence could not be verified experimentally due to the nature of the experimental platform allowing no sensor placement inside of the feet. The experiments were conducted with the help of the sensor ADNS-5030. It has been chosen due to its benefits for the application in smaller mobile robots, such as the utilized robot *Nao*. The walking experiment however revealed some flaws of this choice. The nature of two legged

dynamic walking requires sensors with a larger height and overall better noise tolerance. Sensors for modern gaming mice feature both attributes, but especially recent development of new technologies such as *BlueTrack* and *DarkField Laser Tracking* are promising. They are believed to yield much better results since they claim to work on virtually any surface. But since the sensors are not commercially available for developers testing these sensors has to be done in future work.

References

1. Kanda, T., Ishiguro, H., Imai, M., Ono, T.: Development and evaluation of interactive humanoid robots. *Proceedings of the IEEE* 92(11), 1839–1850 (2004)
2. Thrun, S., Burgard, W., Fox, D.: *Probabilistic Robotics*. MIT Press, Cambridge (2005)
3. Siegwart, R., Nourbakhsh, I.R.: *Introduction to Autonomous Mobile Robots (Intelligent Robotics and Autonomous Agents)*. MIT Press, Cambridge (2004)
4. Borenstein, J., Feng, L., Borenstein, C.J.: Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation* 12, 869–880 (1996)
5. Vukobratovic, M., Borovac, B.: Zero-moment point – Thirty five years of its life. *International Journal of Humanoid Robotics* 1(1), 157–173 (2004)
6. Bonarini, A., Matteucci, M., Restelli, M.: Automatic error detection and reduction for an odometric sensor based on two optical mice. In: ICRA, pp. 1675–1680 (2005)
7. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems* 57(8), 839–845 (2009); *Humanoid Soccer Robots*

SSL-Humanoid

RoboCup Soccer Using Humanoid Robots under the Global Vision

Tadashi Naruse¹, Yasuhiro Masutani², Noriaki Mitsunaga³, Yasunori Nagasaka⁴,
Takashi Fujii⁴, Masato Watanabe⁵, Yukiko Nakagawa⁶, and Osamu Naito⁷

¹ Aichi Prefectural University

² Osaka Electro-Communication University

³ Kanazawa Institute of Technology

⁴ Chubu University

⁵ Toyota National College of Technology

⁶ RT Corporation

⁷ University of Tokyo

info@robocup-ssl-humanoid.org

Abstract. We propose a new RoboCup league called “SSL-Humanoid”. In the SSL-Humanoid league, the global vision system used in the RoboCup soccer Small Size Robot League (SSL) is also employed, and two teams of up to five humanoid robots play on a field whose size is half of the SSL field. This league aims to advance AI research for humanoid robots such as cooperation among robots, tactics and strategies. This contribution is possible due to the substitution of local to a global vision system. This is expected to attract new participants who have been interested in RoboCup but did not attend so far. We propose to organize this league as a sub-league of the original SSL. In this paper, we describe background of the proposal, research topics, road map toward 2014 and a summary of games played in 2009. Finally, we introduce team ODENS’ system as an example of an SSL-Humanoid team.

1 Introduction

Fourteen years have passed since the RoboCup initiative [1] started in 1996. In those years, the technology used in the RoboCup soccer Small Size Robot League (SSL) [2] has developed rapidly to a great extent thanks to the global vision system. SSL games can now entertain audience as interesting robot soccer games. In terms of research and development, it seems that the SSL already matured quite rapidly. We believe that a core feature of the SSL is the global vision, which enables the stable detection of robots and ball positions.

In 2007 the authors unofficially proposed a new RoboCup league in which humanoid robot teams compete with each other under a global vision environment [3]. The idea of the new league arose from a local discussion when the authors gathered during the 2007 RoboCup Japan Open. The goal of the new league is to explore new research topics while keeping the characteristic features of the SSL. This offers the opportunity to easily participate in the RoboCup to those who are interested in it.

In the new league, humanoid robots are used instead of conventional wheeled robots of the SSL, and the global vision system used in the SSL is also employed. We call this new league “SSL-Humanoid”.

This proposal aims to introduce a league which establishes new research topics and to promote the world of RoboCup for new participants. We believe that the SSL-Humanoid will give young talents opportunity to actively participate in the RoboCup and bring a large amount of technical achievement towards the final goal in 2050.

In the following sections, first, we describe the background why we proposed the SSL-Humanoid. Then, we describe a way of running the competition, a number of possible research topics and expected effects of the new league, as well as a road map. Rules used in the 2009 demonstration are also outlined. Furthermore, we briefly discuss the demonstration games conducted at the RoboCup Japan Open and the RoboCup Competition in 2009. Finally, we introduce team ODENS’ system as an example of an SSL-Humanoid team and conclude the paper.

2 Background

2.1 Current Small Size and Humanoid Robot Leagues

Teams of humanoids have soccer games with local vision in the Humanoid Robot League [4] and the Standard Platform League [5]. In both leagues, cameras and image processing units must be embedded in each robot. So far, great efforts were necessary to embed a local vision system in the small robot hardware. In our new league, such efforts are unnecessary due to the use of an off-board global vision system. Researchers can concentrate rather on their Artificial Intelligence (AI) research, such as cooperation among robots and strategy development to win the soccer games.

Meanwhile, the wheeled robots and the global vision of the SSL technically matured in recent years. Games in the SSL became more and more exciting each year. In 2009, the SSL vision [6] was proposed and tested. SSL vision is a common platform used to give teams position and orientation of each robot and ball in real-time. This system releases the burden of developing an image processing system for each team. Several games were played with SSL vision in 2009 and verified as working well.

However, the technological development of the robots seems to be turning into an arms race. This development makes it difficult for new teams to participate in the SSL because of the technical and economical difficulties. These problems especially arise when new teams try to catch up the latest advances. Therefore, we propose a new league, which aims to overcome the problems and make further advances in research and technological development.

2.2 Advances in Humanoid Robots for Hobby Use

Recently, small size humanoid robots are gaining popularity among hobbyists due to falling prices. Using these biped robots, many people enjoy soccer and Robo-One fighting games. Most of the participants currently operate their robots using remote radio control, while a fraction of people let their systems work autonomously. It seems that

most of participants have interests in creating and controlling the robot but not in automatic control. We believe that by giving incentives and proper tools to support autonomous robots in the new league we will attract a larger audience which did not join due to the inherent technical difficulties so far. The new league will be the key to capture their interests and provide the most suitable opportunity for them.

2.3 Intelligent Space and Humanoid Robots

Intelligent Space (IS) is an environment in which vision sensors, auditory sensors, and Radio Frequency (RF) tags are embedded [7]. Intelligent Spaces are actively studied since it can be one of the core tools to realize a symbiotic life platform between humans and robots. The benefit of this approach is that one can reduce the number of sensors locally embedded in each robot. We believe that in the future many robots will be used in the IS, which helps to considerably reduce the number of sensors necessary in the robots. Then, the robot in the IS might be the best approach to enable humanoid-like robots to play an important role in our everyday environment.

3 RoboCupSoccer Using Humanoid Robots under the Global Vision

3.1 The Objective of SSL-Humanoid

In the SSL-Humanoid, cameras and computers, which perform the image processing and the behavior decision, can be placed outside of the soccer field. We can utilize the key technologies such as image processing systems and radio systems that were developed in the SSL. Moreover, it is possible to use hobby type humanoid robots in the SSL-Humanoid. These lower both technological and economical barriers at the same time. We can establish a new way to participate in RoboCup for those who gave up to participate in the humanoid league due to economical issues as well as for those who just enjoy playing with humanoid robots in their spare time.

In addition, the SSL-Humanoid league shares the same idea as the intelligent space approach and the system can be regarded as one of the intelligent systems. We believe that many application fields will benefit from the SSL-Humanoid's results, most notably the research on symbiotic life of humans and robots.

3.2 Starting SSL-Humanoid

SSL-Humanoid will start as a sub-league of the SSL. We use half of the SSL field as the new playing field for the time being and share over-field cameras and markers (team markers and sub markers) with SSL. Each team may use three to five robots in a game.

In order to ensure the fairness of the competition and provide an easy-to-participate environment, the organizing committee will provide over-field cameras and the image processing unit, i.e. the SSL-Vision [6] itself or a similar one. Markers specified by the rules[3] should be placed on the top of each robot. The output of the image processing unit is a pair of position and orientation of each object on the field. These are delivered

Table 1. Roadmap of the SSL-Humanoid

Year	Description
2009	Demonstration games were played. Each robot had markers and each team used its camera and its radio system.
2010	All teams share one over-field camera and use the SSL-Vision.
2012	Unify the radio system.
2013	Start using corner pole cameras.
2014	Start a marker-less class and three dimensional vision class .

to the computer of each team through a network. To avoid conflict of radio communication, the committee will clearly define the type of radio system allowed to use. The use of software libraries such as the Robot Technology Middleware (RTM) [8] is encouraged to promote the participation of new teams. It is also useful for the accumulation of know-how.

On the other hand, teams are still allowed to use their own image processing system in order to develop an improved global vision system if they choose to do so. One example of such systems we consider is a vision system that processes the images given by the cameras slantingly mounted at poles around four corners of the game field. Such arrangement become important when a wider field will be used. Of course, such a system will be a next generation of the shared SSL-Humanoid vision system if it is approved.

3.3 Roadmap

We propose a concrete timetable in order to smoothly shift to new stages as shown in Table 1. To clarify the system images, those of the SSL-Humanoid in 2009 and around 2014 competitions are shown in Figure 1. The year 2009 was a preparation phase. Using the technology developed in the SSL, the demonstration games of the SSL-Humanoid were conducted.

We will introduce shared vision in 2010. The image data captured by the cameras and/or their processing results are shared between teams. The shared vision for the SSL-Humanoid will be modified to make it possible to recognize the three dimensional robots rather than two dimensional markers. Unified radio and marker-less class are introduced in 2012 and 2014, respectively. The marker class will be phased out in 2014. In the marker-less class, three-dimensional recognition of the robots is necessary. In order to acquire three dimensional information, we will introduce cameras slantingly mounted on poles around the corners of the field in 2013.

3.4 Research Directions

There are two main research directions in the SSL-Humanoid, i.e. AI and robotics research using global vision, and developing higher-level global vision.

AI and Robotics Research Using Global Vision. The local vision of the robots in the Humanoid League determines the entire performance of their robots. However, there

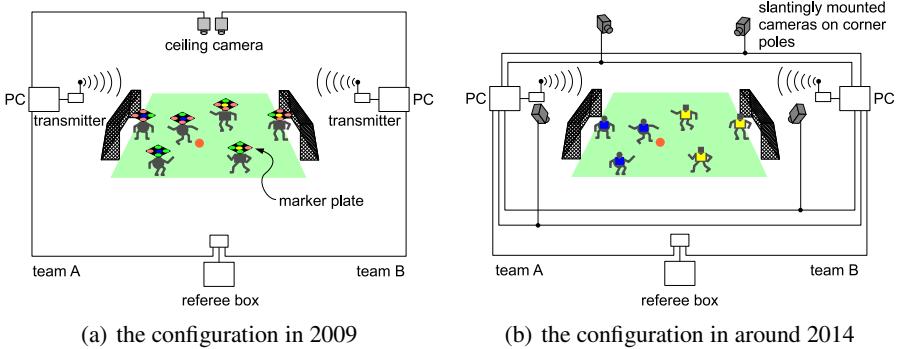


Fig. 1. These figures show configurations of SSL-Humanoid in 2009 and around 2014. In 2014, robots play without markers on their head. The global vision system consists of over-field cameras in 2009 and slantingly mounted cameras in 2014.

are many technical difficulties such as restrictions of viewing field, resolution and processing speed since whole local vision system must be embedded in the robot.

Global vision does not have these difficulties, which helps in the promotion of humanoid robot research. Participants can focus on the research of mechanics, motion control of humanoid robots, the research of strategies and tactics of multiple agents as well as cooperation.

Developing Higher-Level Global Vision. From the view point of the vision system, the features of the SSL are “two dimension” and “markers” on the top of the robots. So, the global vision system of the SSL is not so difficult to develop. However, we have to go one step further toward the final goal, where “three dimension” and “no marker” become main features. This means that the new global vision system should recognize the three dimensional humanoid robots which have no markers on their top. SSL-Humanoid will give the best opportunity for developing such global vision system.

The goal of the global vision in the SSL-Humanoid is to recognize what kind of posture (joint transformation) and motion robots make. In order to correctly observe three dimensional objects, it is necessary to view the objects from a slanted direction not directly above. Cameras around the corners of the field will play this role. Another goal is the unification of all camera images to acquire 3-D information in real-time. The ultimate goal is to construct an automatic referee system and a play-by-play broadcasting system for all soccer games.

3.5 Research Topics

Main research topics in the SSL-Humanoid are the following:

Motion Control of Humanoids. In the current humanoid league, the performance of the robots seems to be restricted by weight, volume, power consumption and processing speed of the image processing unit embedded in the robot. The global vision will allow fast and light-weight robots, which will promote the research and learning of skills.

Cooperation Among Humanoid Robots. Not only the research on strategy and tactics based on the global information but also the research on cooperation between humanoid robots will be promoted due to the use of global vision.

Recognition and Understanding of Humans Based on Images. The occlusion problem among humanoid robots is one of the most important problems to be solved in the SSL-Humanoid vision. Recognition and understanding technology developed in the new league can also be applied to the field of human actions' recognition and understanding. The SSL-Humanoid can contribute to the solution of these problems.

Building Intelligent Space. In the SSL-Humanoid, it is easy to build an Intelligent Space (IS) since the global vision is embedded in the environment and humanoid robots are playing a soccer game by using the information provided by the IS. This means that the new league can take a leading role in the IS research by providing good test benchmarks for its development. Typical research topics in this area are an effective arrangement of sensors, information synthesis from multiple (heterogeneous) sensors, and compensation methodology for processing/communication delays under an environment where robots move dynamically.

4 Competitions Held in 2009

4.1 Outline of the 2009 Rules

A great portion of the SSL-Humanoid rules is in common with the rules of the SSL, which eases a transition from the SSL to the SSL-Humanoid. The 2009 rules are partially outlined here. Full version of the 2009 rules is available on the website [9]. The readers are encouraged to visit the website for further details.

1. The Game field

- The field size is 4050×3025 [mm] (half of the SSL field).
- The playing surface is a green felt mat or the like.
- The size of each goal is 1000×600 [mm] (width \times height).

2. The Ball

- An orange tennis ball is used.

3. The Number of robots

Each team consists of up to three robots, where one of those may be the goalkeeper.

4. The Structure of robots

- A robot has two legs and two arms.
- The height is in the range between 200 and 400 [mm].
- The weight is 4 [kg] or less.
- The number of joints is at least 5 for each leg and at least 3 for each arm.
- A marker plate, with a size of 120×120 [mm], must be mounted on the top of each robot.



Fig. 2. An SSL-H game in the RoboCup Japan Open 2009 Osaka

Table 2. Standings of the SSL-Humanoid in the RoboCup Japan Open 2009. Note, RoboDragons' games were forfeit games.

		S				

5. Cameras, Radio and Autonomy

- The over-field cameras and the external computers are allowed.
- Radio communication between robots and external computers is allowed.
- Robots must be controlled by computers. Remote control by humans is prohibited.

6. The Referee

A (human) referee controls the game.

7. The Assistant Referee

A referee box similar to the one used in the SSL is employed to control the game. A system referee (an assistant referee) controls the referee box, acts as timekeeper and keeps a record of the match.

8. The Duration of the Match

The match lasts for two equal periods of five minutes, unless otherwise mutually agreed between the referee and the two participating teams. The half-time interval should not exceed five minutes. Each team may have up to four timeouts in the game. A total of 10 minutes is allowed for all timeouts.

Table 3. Standings of the SSL-Humanoid in the RoboCup 2009

4.2 RoboCup Japan Open 2009 in Osaka

From 8 to 10 May 2009, the RoboCup Japan Open was held in Osaka. In this competition, we established official games of the SSL-Humanoid for the first time as a sub-league of the SSL. Four teams participated: KIKS (Toyota National College of Technology), ODENS++ (Osaka Electro-Communication University and Osaka Prefectural College of Technology), Owaribito-CU (Chubu University) and RoboDragons (Aichi Prefectural University). We used the SSL's field alternatively with the SSL games and shared the PCs and cameras with the SSL's. Total of six games were played during the period. Figure 2 shows a scene from one of the games and Table 2 shows the standings. Figure 3 shows examples of a robot and basic skills seen in demonstration games in 2009.

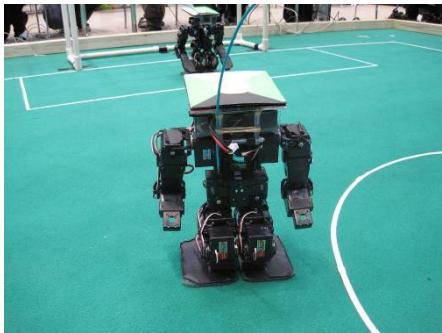
In the competition impressive games were played, although it was the first competition. Highlight movies of the games are available on the web site of SSL-Humanoid[3]. Each of the participated teams built up their humanoid system and implemented basic and advanced skills such as kicking, shooting, clearing a shot, and struggling for the ball in rather short period. It seems very hard to implement them under the local vision system, although both teams had enough experience in the SSL games. Based on these experiences, we are certain that the SSL-Humanoid will successfully promote the research topics as described in section 3.5.

4.3 RoboCup 2009 in Graz Austria

From 28 June to 5 July 2009, the RoboCup was held in Graz Austria. In the competition, we had demonstration in the SSL on 4 and 5 July, the final days. Four teams participated: KIKS (Toyota National College of Technology), ODENS (Osaka Electro-Communication University), Owaribito-CU (Chubu University) and RoboDragons (Aichi Prefectural University). We used a field which was not used in finals. Six round-robin matches were played. Figure 3 shows some scenes from the demonstration games and Table 3 shows the standings. In the competition, impressive games were played as well and spectators were very much interested in the games. Highlight movies of the games are also available on the web site[3].

4.4 Osaka Electro-Communication University's Team

Here, we introduce the team ODENS from Osaka Electro-Communication University. Figure 4 shows the system configuration of the team ODENS. A marker plate is attached



(a) An SSL-Humanoid robot



(b) Attacking into the opponent side



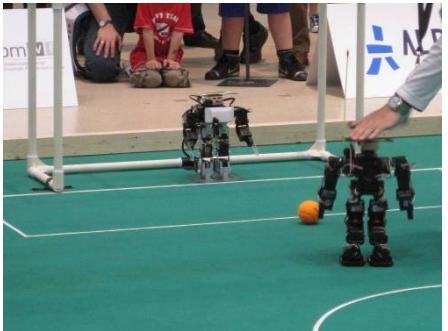
(c) Which robot gets the ball?



(d) Trying Shot



(e) Defence



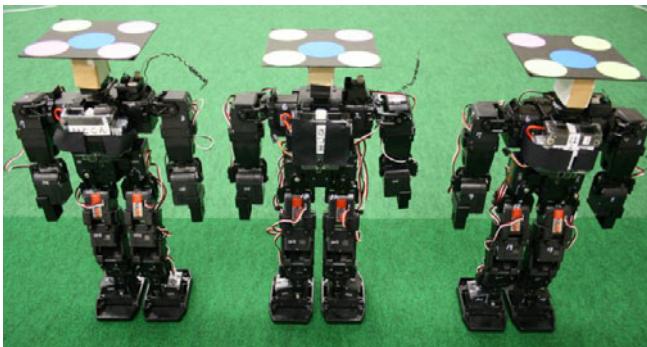
(f) Penalty Kick

Fig. 3. Typical actions of humanoid robots in the RoboCup 2009 demonstration games ((e) was taken in the RoboCup Japan open 2009)

to the head of each humanoid robot. Over the field, a camera is attached to the torus. It takes the image of the robots and the ball on the field and sends it to the server PC every 16 msec. In the server PC, positions and orientations of the robots and the ball are computed from the image, then they are sent to the client PCs through the network. In the client PCs, robot motions are computed through a decision making program. In each robot, motions such as a forward-walk, a backward-walk and a kick are implemented in advance. Therefore, motion number is sent to each robot through the radio. This process is executed repeatedly to realize the soccer game autonomously by humanoid robots.

Table 4. Specifications of the camera and PCs

camera	maker & model	Basler Vision Technologies, A311fc (658×492pixel,60fps)
	lens	Tamron Co., Ltd 12VM412ASIR
server PC	maker & model	Seiko Epson Corporation, NJ3100
	CPU	Intel Core2Duo P8700 2.53GHz
	RAM	2GB
	IEEE1394	Sonnet Technologies, Inc. FWUSB2-E34
	OS	Windows XP
client PC1	maker & model	Panasonic Corporation, CF-W7
	CPU	Intel Core2Duo U7700 1.33GHz
	RAM	1GB
	OS	Ubuntu 8.10
client PC2	maker & model	Panasonic Corporation, CF-W5
	CPU	Intel Core Solo U1300 1.05GHz
	RAM	512MB
	OS	Ubuntu 8.10

**Fig. 4.** Humanoid robots employed in team ODENS

Humanoid robots used are MANOI AT01 robots, product of KYOSHO Corporation. They are shown in figure 4. Motors driving foot joints are high torque RC servo motors, KRS-4013HV, products of Kondo Kagaku Co. Ltd, which are replaced from original ones. Also, the radio receiver (25MHz band) and gyro / acceleration sensor, which are also the products of Kondo Kagaku Co. Ltd, are embedded in the robot. The battery is a lithium polymer battery¹, 3 cells, 1350 mAh. The specifications of camera and PCs of team ODENS are shown in Table 4. The client PC1 controls two robots and the client PC2 controls one robot.

¹ Misusing the battery may cause the battery to get hot, rupture, or ignite and cause serious injury.

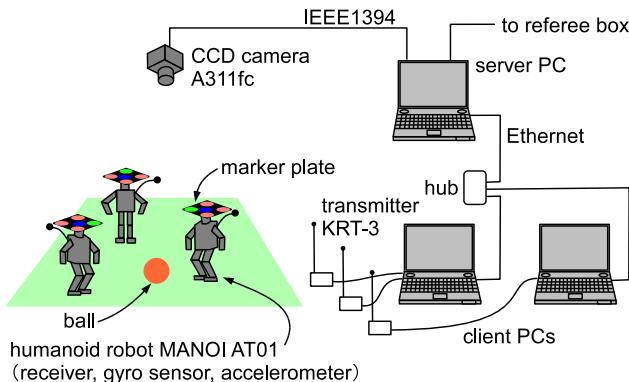


Fig. 5. System configuration of team ODENS

5 Concluding Remarks

We proposed a new league “SSL-Humanoid” in this paper. In the league, humanoid robot teams compete with each other under a global vision system, and discussed the research directions and topics. Under the stable global vision, AI research will be promoted greatly. During the SSL-Humanoid games held in the Japan Open and other international events, it was shown that the robots were moving smoothly in the field and various actions including kicks and defenses were successfully demonstrated. Higher-level cooperative plays will be implemented within a few years.

The SSL-Humanoid started in the Japan Open 2009. We hope that the new league is approved and will start in the RoboCup international event as an SSL’s sub-league from 2011. We encourage everybody interested in this to join and participate in the new SSL-Humanoid league.

Acknowledgement

Authors would like to thank Mr. Bjoern Rennhak for correcting and improving English of our paper.

References

1. RoboCup, <http://www.robocup.org>
2. RoboCup Small Size Robot League,
<http://small-size.informatik.uni-bremen.de>
3. RoboCup SSL Humanoid,
<http://robocup-ssl-humanoid.org/>
4. RoboCup Humanoid League,
<http://www.tzi.de/humanoid/bin/view/Website/WebHome>
5. RoboCup Standard Platform League,
<http://www.tzi.de/spl/bin/view/Website/WebHome>

6. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 425–436. Springer, Heidelberg (2010)
7. Lee, et al.: An agent for intelligent spaces: functions and roles of mobile robots in sensored, networked and thinking spaces. In: IEEE Conference on Intelligent Transportation System (ITSC) (1997)
8. Ando et al.: RT-Component Object Model in RT-Middleware Distributed Component Middleware for RT (Robot Technology). In: IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) (2005)
9. RoboCup, SSL-Humanoid Competition Rules,
<http://robocup-ssl-humanoid.org/SSL-H-Rule2009.pdf>

Localization with Non-unique Landmark Observations

N. Ergin Özetur and H. Levent Akın

Boğaziçi University, Department of Computer Engineering, 34342 Istanbul, Turkey
`{nezih.ozkucur,akin}@boun.edu.tr`

Abstract. In the probabilistic robot localization problem, when the associations of observations with the landmarks in the map are given, the solution is straightforward. However, when the observations are non-unique (e.g. the association with the map is not given) the problem becomes more challenging. In the Standard Platform League (SPL) and other similar categories of RoboCup, as the field setups evolve over years, the observations become less informative. In the localization level, we have to seek solutions with non-unique landmark observations. In this paper, we established the probabilistic model of the problem and showed the difficulty of optimal solution. After that, we introduce our importance sampling based approximate solution and implicit hypothesis pruning. We give results from simulation tests in the SPL setup using corners and goal bar observations and discuss characteristics of our approach.

1 Introduction

In the probabilistic robot localization problem, when the associations of observations with the landmarks in the map are given, the solution is straightforward [1]. We refer to such observations as *unique* observations. However, when the observations are *non-unique* (e.g. the association with the map is not given) the problem becomes more challenging, since the possible associations should now be considered as random variables.

In the Standard Platform League (SPL) and other similar categories of RoboCup, the objects can be recognized along with associations in the real world by color codes and/or shapes. However as the field setups evolve over the years, the observations during competitions become less informative and inferring the identities from sensors gets harder. We, thus, have to seek solutions to the localization problem in case of non-unique landmark observations. In this paper, we address this problem and propose a solution based on *Sequential Importance Sampling/Resampling* (SIS/Resampling) [2] with multiple hypotheses.

In some RoboCup Leagues, several approaches for ambiguous and decreasing number of landmarks such as [3,4,5,6,7] have been proposed. The solution techniques differ, but the general trend is about resolving ambiguities before applying the localization algorithm by using as much information and heuristics as possible. In [3], the ambiguity is resolved using a compass. In [5], the observations are identified by using autonomously extracted extra features from the image. The most generic solution is in [7], where the line observations form constraints and the problem is modeled as a constraint satisfaction problem. One other approach to our problem is *Multi-Hypothesis Tracking*

(MHT) [1], where the possible associations of observations form the hypotheses and each hypothesis is simultaneously tracked with a Kalman Filter [8]. In MHT, complexity increases with ambiguities so pruning is a crucial step which is generally based on some problem dependent heuristics or similar assumptions. In our approach, we maintain multiple hypotheses as samples (along with the robot pose) and they are implicitly pruned in the resampling step. As we will show, our method is generic and does not rely heavily on problem specific heuristics or exploits.

In the rest of this paper, in Section 2, we first establish the probabilistic model of the problem with non-unique landmarks. With this model, we show why this variant of the problem is harder and requires approximate solutions. After that, we introduce our approximate solution based on SIS/Resampling and specific optimizations based on RoboCup setups. In Section 3, we report our results based on experiments and discuss different characteristics of our approach. In Section 4, we give conclusions and directions for further improvements and research.

2 Proposed Approach

A robot moving on a field with noisy sensory inputs can be modeled as a dynamic system with nonlinear evolution functions (i.e. a nonlinear dynamical system). Let x_t denote the robot pose, u_t denote the control input (odometry readings) and z_t^i denote the observation from the i th landmark in the map at time t . The directed graph model for this problem is given in Figure 1.

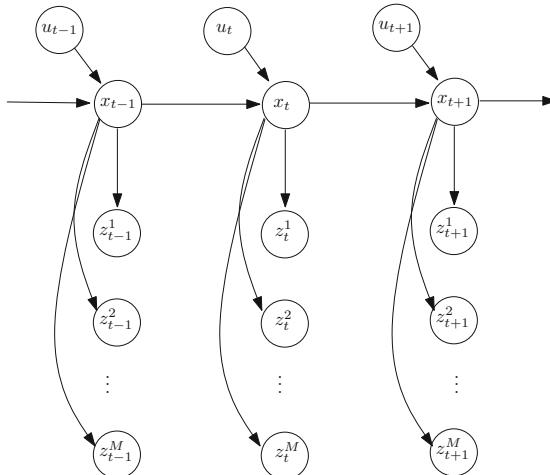


Fig. 1. Directed graph model of localization problem with unique landmarks

The model is of the form

$$\begin{aligned} x_{t+1}|x_t, u_{t+1} &\sim N(x_{t+1}; f(x_t, u_t), Q) \\ z_t^i|x_t &\sim N(z_t^i; g(x_t, i), R) \end{aligned}$$

where $f(x, u)$ basically calculates the transformation of the pose based on odometry and is a nonlinear function and Q is the covariance matrix of noise on u_t . The measurement function $g(x, i)$ calculates the expected distance and bearing of a landmark based on the location of i^{th} landmark on the map. Matrix R is the covariance of the noise on observations.

The exact inference in this model is difficult due to nonlinearities, but approximate solutions based on Extended Kalman Filter (EKF) [8] and Particle Filter PF [2] exist and have many successful applications.

In the model described above we assumed that z_t^i is observed from landmark i in the map. Now, let us restrict the information of observation index. Assume z_t^i denote the observation from the i^{th} landmark subset in the map instead of single i^{th} landmark. In the soccer field setting, one example would be that a *yellow goal bar* is observed but we do not know if it is the *left yellow goal bar* or the *right yellow goal bar*. In this case, the *yellow bar* group represents a subset in the map. In Figure 2 such an observation sample is given. We identify bars when the upper bar is observed, however when the robot is close to the bars, it cannot observe the upper bar.

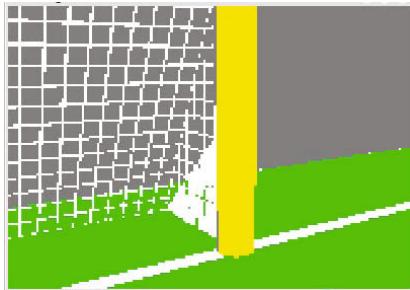


Fig. 2. Non-unique bar observation in the color-segmented image

Assume that there are K such groups. Now let us define a discrete label vector as $l_t = \{l_t^1, l_t^2, \dots, l_t^K\}$, where l_t^i denotes the index of the i^{th} non-unique observation in the landmark group. We can also add one more value to l_t^i to denote missing observations (e.g. *yellow bar* is not observed). The value of l_t^1 might be 0 for absence, 1 for z_t^1 is *left yellow goal bar* and 2 for z_t^1 is *right yellow goal bar*. (For simplicity in the method description, we assume at most one landmark can be identified from a group at any instant. However the extension is trivial and our implementation is generic.)

The new discrete label vector is a random variable. Let L be the number of possible combinations of the label vector, then we can represent the distribution as $p(l_t) = \{\alpha_{t,1}, \dots, \alpha_{t,L}\}$, where $\alpha_{t,i}$ denotes the probability that $l_t = i^{\text{th}} \text{ combination of } K \text{ variables}$. The new model with non-unique landmarks is given in Figure 3. According to the model, the observations depend on labels which only depend on previous labels. This may seem counter intuitive, because in the (often used) naive data association method, where a non-unique observation is associated to the closest landmark in the map, the label appears to depend on the observation and the pose. However what is actually

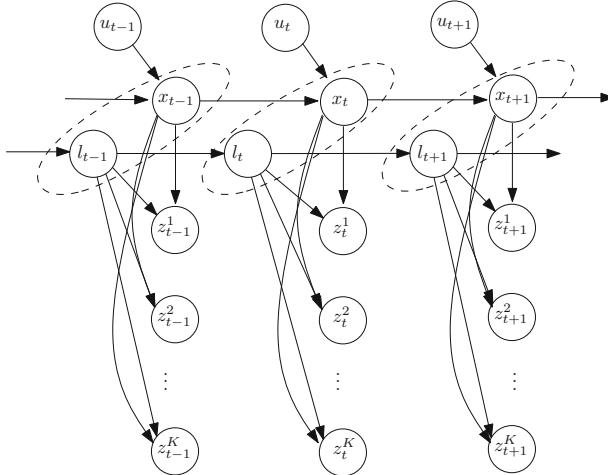


Fig. 3. Directed graph model of localization with non-unique landmarks

calculated is $p(l|z, x) = \frac{p(z|l, x)p(l)}{\sum_l p(z|l, x)p(l)}$ (focusing on a single instance) which is perfectly explained by our model. The label variable is modeled as dependent on the previous label; this enables us to use the information that observing the same label in consecutive steps is more likely.

In this new model, the state to be estimated is a mixture of the continuous robot pose and the discrete labels. This type model is called as *switching dynamic system* [9] or more specifically *switching observation model* [10]. It is called so because the observation model $p(z|x)$ changes (switches) with a discrete random value. The posterior distribution to be estimated then is

$$\begin{aligned} p(x_t, l_t | z_{1:t}, u_{1:t}) &\propto p(z_t | x_t, l_t) p(x_t, l_t | z_{1:t-1}, u_{1:t}) \\ p(x_t, l_t | z_{1:t-1}, u_{1:t}) &= \int_{x_{t-1}} \sum_{l_{t-1}} p(x_t | x_{t-1}) p(l_{t-1} | l_t) p(x_{t-1}, l_{t-1} | z_{1:t-1}, u_{1:t-1}) \end{aligned}$$

If we assume Gaussian distributions on the initial, observation and transition models, this posterior can be estimated using Kalman filter equations. However at time t , the posterior would be $p(x_t, l_t | z_{1:t}, u_{1:t}) = \sum_{i=1}^{L^t} \beta_i N(x_t; \mu_i, \Sigma_i)$ where β_i is a weight parameter which sums to 1. This distribution is a mixture of Gaussians and the number of components grow exponentially over time (L^t at time t) as shown in Figure 4. This is the primary reason of difficulty in our switching model. One approach to overcome this complexity is Multi-Hypothesis Tracking (MHT) method with pruning [1]. The idea in pruning is to discard mixture components with low weights (or with other heuristics) and maintain a reasonable amount of components.

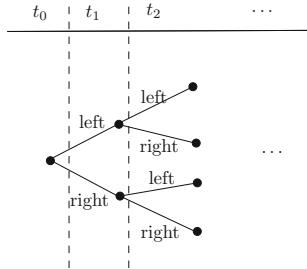


Fig. 4. Number of mixture components grow exponentially over time (Assume $L = 2$)

2.1 Approximation with Sequential Importance Sampling / Resampling

We solve the complexity problem implicitly by representing the mixed state with a particle set and predict/update with importance sampling/resampling. Let the particle set at time t be $X_t = \{x_t^{\{i\}}, l_t^{\{i\}}, \omega_t^{\{i\}}\}$, $i = 1, \dots, N$ where $\omega_t^{\{i\}}$ is the importance weight of the i^{th} particle. To generate samples from the target posterior $p(x_t, l_t | z_{1:t}, u_{1:t})$, we will use the prior $p(x_t, l_t | z_{1:t-1}, u_{1:t})$ as the proposal distribution. This is the well known particle filter algorithm (Figure 5). The N_{eff} is the effective sample size and is a metric to decide performing the resampling step [2].

Compared to the Multi Hypothesis Tracking algorithms, this algorithm also supports multiple hypotheses and pruning. At time t different particles represent different hypotheses (pose and labels). The new hypotheses are introduced in the sampling step (line 8). In the resampling step, bad hypotheses are pruned (if there are any). As we will show in Section 3, the label transition probabilities $p(l_t | l_{t-1})$ are important for adaptivity of the algorithm. Local spatial relations between landmarks can be coded in this transition.

The complexity of this algorithm is $O(NK)$ where N is the number of particles and K is the number of groups. The complexity is the same with the unique landmark case (K would be number for unique landmarks in this case) but an equivalent adaptivity would require more particles.

3 Experiments and Results

We conducted our experiments in the SPL setup of year 2009. The platform is Aldebaran Nao humanoid robot [11]. It has two cameras positioned vertically where the active camera can be switched on demand. The testing environment is the Webots simulator [12].

Our vision module can perceive goal bars, L type line intersections and T type line intersections. It cannot identify intersections but can identify goal bars if the upper bar is in the field of view. However at close range, the upper bar is not in the field of view and the bars cannot be identified. In such settings, there are four landmark groups as *YellowBar*, *BlueBar*, *LCorners* and *TCorners*. The groups *YellowBar* and *BlueBar* include two landmarks each, group *LCorners* includes eight landmarks and group *TCorners* includes six landmarks. We can estimate the distance of the bars, but the distance estimation for the corners is not accurate, so all of the corners are bearing only landmarks. The field setup and landmarks are given in Figure 6.

Algorithm *PF_NonUnique*

```

1: Sample  $x_0^{\{i\}} \sim p(x_0)$ 
2: Sample  $l_0^{\{i\}} \sim p(l_0)$ 
3:  $\omega_0^{\{i\}} = \frac{1}{N}$ 
4: for  $t = 1$  to  $T$  do {Each time step}
5:    $X_t \leftarrow \emptyset$ 
6:   for  $i = 1$  to  $N$  do {For each particle}
7:     Sample  $x_t^{\{i\}} \sim p(x_t|x_{t-1} = x_{t-1}^{\{i\}})$ 
8:     Sample  $l_t^{\{i\}} \sim p(l_t|l_{t-1} = l_{t-1}^{\{i\}})$ 
9:      $\omega_t^{\{i\}} = \omega_{t-1}^{\{i\}} \prod_{j=1}^K p(z_t^j|x_t^{\{i\}}, l_t^{\{i\}})$ 
10:     $X_t \leftarrow X_t \cup \{x_t^{\{i\}}, l_t^{\{i\}}, \omega_t^{\{i\}}\}$ 
11:   end for
12:   Normalize weights
13:    $N_{eff} = \frac{1}{\sum_{i=1}^N (\omega_t^{\{i\}})^2}$ 
14:   if  $N_{eff} < N_{thr}$  then {Resampling conditions met?}
15:     Resample
16:      $\omega_t^{\{i\}} = \frac{1}{N}$ 
17:   end if
18: end for

```

Fig. 5. Algorithm for Particle Filter with non-unique landmarks**Fig. 6.** SPL 2009 field setup and our landmarks

3.1 Experiment 1

Our first experiment is based on a simple scenario for demonstrating multiple hypotheses in the particle set and domination of the true hypothesis. In this scenario, the robot stands still just outside the yellow penalty box, facing the yellow goal, and scanning with its head continuously. The robot observes the right and left yellow bars

sequentially and does not know the identity of the currently observed bar. This scenario is given in Figure 7. There are two possible hypotheses in this scenario, and in the first few observations, both hypotheses are maintained in the belief state. After observing both bars, for each observation both hypotheses are sampled (each observation can be left or right) but the true hypothesis dominates in the resampling state since the wrong hypothesis results with lower weights. In Figure 8, the particle set is given at the beginning and after convergence.

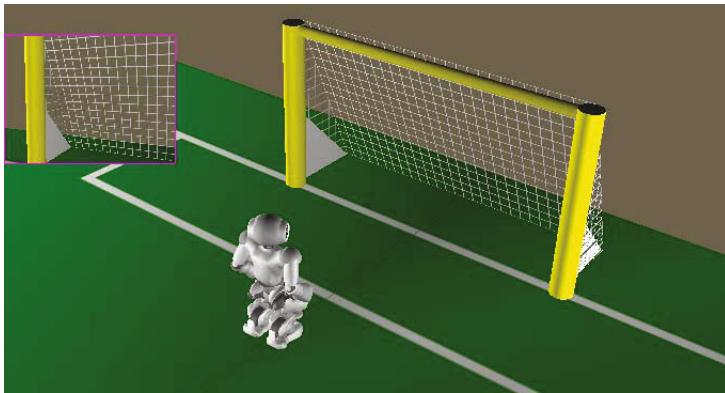


Fig. 7. Scenario for the first experiment

3.2 Experiment 2: Effect of the Transition Model

In the second experiment, we aimed to demonstrate the effects of label transition model $p(l_t|l_{t-1})$. We know that one T corner and two L corners on the left hand side of the yellow penalty box are close to the left yellow bar. This implies that after observing the left yellow bar, there is a high probability of observing those corners and low probability of observing any other corners. In the transition model, we can represent this information. In this experiment, the robot starts in front of and facing the yellow goal, turns around and moves toward the blue goal while scanning with its head continuously. It observes almost all of the landmarks in this scenario.

In experiment 2a, the transition model is naive and only represents identity transition (it samples the same label with high probability and randomly selects another label with low probability). In experiment 2b, the transition model represents the hints due to the spatial relations described above. In experiment 2a, the particles frequently converge to wrong positions and fail to adopt different landmarks. In experiment 2b, pruning of the wrong hypotheses is more successful. A snapshot of particles for both runs are given in Figure 9 where we can see the wrongly localized particles with wrong hypotheses. We can see the path error in a sample 600 step run of both experiment 2a and 2b in Figure 10. The wrong hypotheses dominated more frequently with the naive transition model than the informative transition model. The mean squared error (MSE) and variance of the particles in the corresponding runs are given in Figure 11. We can see

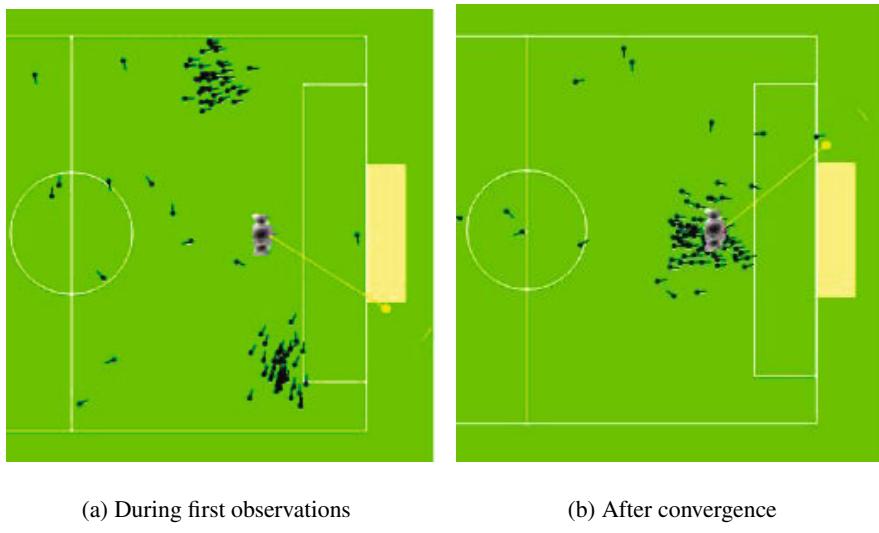


Fig. 8. Snapshot of particles in the first experiment

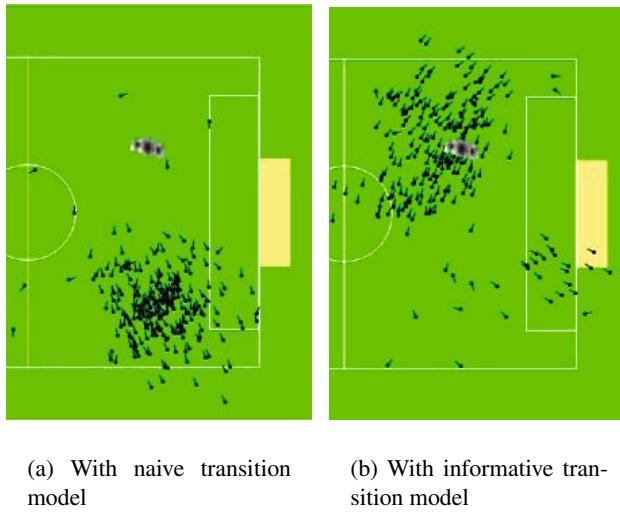


Fig. 9. Snapshot of particles in the second scenario

the effect on the MSE (especially in the beginning, the informative model is more adaptive). However the effect on the variance is smaller because most of the landmarks are non-unique and the wrong hypotheses are consistent in itself.

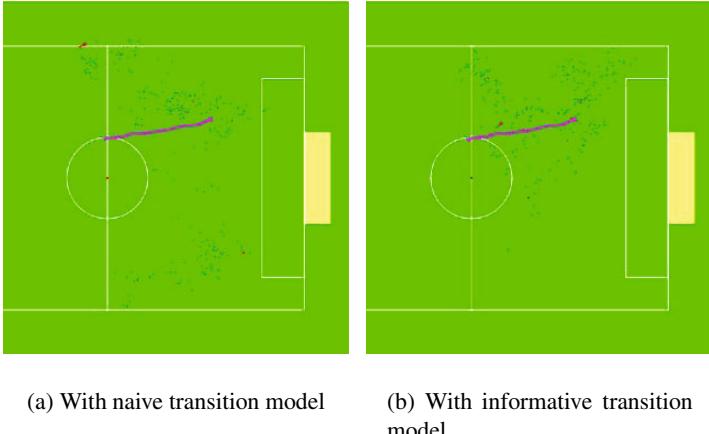


Fig. 10. PaLth errors in the second scenario. Line is the ground truth position and dots are estimated poses.

We used 100 particles in the first experiment and 250 particles in the second experiment. Our generic implementation is fast enough to run without affecting the frame per second processing rate of the overall system.

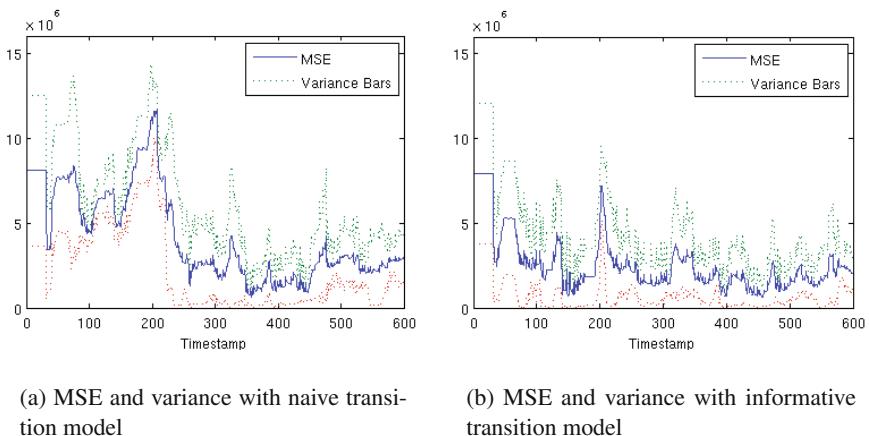


Fig. 11. Path errors in the second experiment

4 Conclusions

Localization with non-unique landmarks is challenging and optimal solutions have high complexity. In this paper, we modeled the problem as a switching observation

model and proposed an approximate solution based on sequential importance sampling/resampling. As opposed to other approximations based on heuristics in the pruning of hypotheses, our algorithm does not employ direct use of heuristics but performs implicit pruning in the resampling step due to the nature of the model.

As we showed with experiments, the transition model is important for the adaptivity of the algorithm. We hand coded the spatial relations of landmarks in the label transition model, however a better and more robust approach would be training of the transition model with the supervised training data including true identities.

Acknowledgments

This research has been partially supported by by Turkish State Planning Organization (DPT) under grant DPT 2007K120610, the Scientific and Technological Research Council of Turkey (TUBITAK) 2211 National Graduate Scholarship Program, and by Boğaziçi University Research Fund project 09M105.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
2. Doucet, A., Godsill, S., Andrieu, C.: On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing* 10(3), 197–208 (2000)
3. Strasdat, H., Bennewitz, M., Behnke, S.: Multi-cue localization for soccer playing humanoid robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 245–257. Springer, Heidelberg (2007)
4. Herrero-Pérez, D., Martínez-Barberá, H.: Robust and efficient field features detection for localization. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 347–354. Springer, Heidelberg (2007)
5. Sturm, J., van Rossum, P., Visser, A.: Panoramic localization in the 4-legged league. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 387–394. Springer, Heidelberg (2007)
6. Jüngel, M., Risler, M.: Self-localization using odometry and horizontal bearings to landmarks. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 393–400. Springer, Heidelberg (2008)
7. Göhring, D., Mellmann, H., Burkhard, H.-D.: Constraint based belief modeling. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 73–84. Springer, Heidelberg (2009)
8. Welch, G., Bishop, G.: An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA (1995)
9. Oh, S.M., Rehg, J.M., Dellaert, F.: Parameterized duration mmodeling for switching linear dynamic systems, pp. II: 1694–1700 (2006)
10. Caron, F., Davy, M., Duflos, E., Vanheeghe, P.: Particle filtering for multisensor data fusion with switching observation models: Application to land vehicle positioning. *IEEE Transactions on Signal Processing* 55(6-1), 2703–2719 (2007)
11. Aldebaran-Nao, <http://www.aldebaran-robotics.com/eng/nao.php>
12. Webots. Commercial Mobile Robot Simulation Software, <http://www.cyberbotics.com>

MR-Simulator: A Simulator for the Mixed Reality Competition of RoboCup^{*}

Marco A.C. Simões¹, Josemar Rodrigues de Souza^{1,2},
Fagner de Assis Moura Pimentel¹, and Diego Frias¹

¹ Bahia State University (ACSO/UNEB), Salvador, BA, Brazil

² Integrated Center of Manufacture and Technology (SENAI-CIMATEC), Salvador, BA, Brazil
`{msimoes, josemar}@uneb.br,`
`{diegofriass, fagnerpimentel}@gmail.com`

Abstract. Simulations play an important roll in the development of intelligent and collaborative robots. In this paper we describe the development of a simulator for the RoboCup Mixed Reality (MR) competition. MR uses a mix of simulated and real world to foster intelligent robotics development. Our simulator "virtualizes" the players within the MR hardware and software framework, providing the game server with the player-positional information usually supplied by the central vision system. To do that the simulator, after receiving the action commands from each agent (player) must track its expected trajectory and behavior in collision events for all the players. We developed fast algorithms for simulating the robot motion in most usual cases. The simulator was validated comparing its results against the real environment and proved to be realistic. This tool is important for setting-up, training and testing MR competition teams and could help to overcome current difficulties with robot hardware acquisition and maintenance.

1 Introduction

Simulators are important tools for development and support of RoboCup. Can be used in various leagues with the purpose of testing the performance of teams prior to competition in the real environment. The RoboCup simulation league is divided into three distinct competitions: 2D, 3D and Mixed-Reality (MR). The first two competitions are fully simulated while the last has a virtual environment (simulated) in which physical robots interact [1]. The virtual game environment comprising the field, the ball and football beams is simulated by a server which communicates with software agents [1] providing to them the position of each player in the field and trajectory of the ball. Each software agent processes that information and take decisions for the best movement and behavior of the corresponding player in each time step. Action commands are then transmitted by infrared interface to the physical robots. This paper presents the substitution of physical robots by a software module that we have called MR-simulator.

* This project is partially funded by UNEB, and CNPQ. The authors thank the great help and provision of Juliana Fajardini Reichow in elaboration of this article.

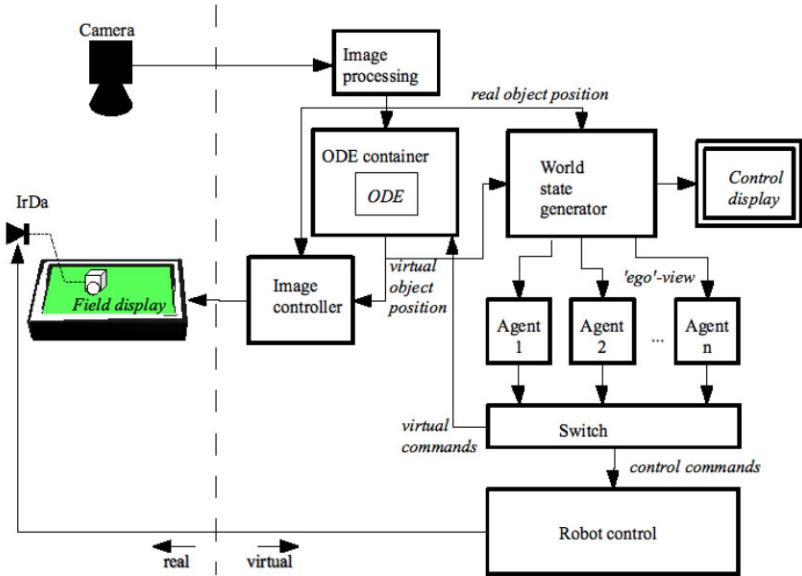


Fig. 1. Structure of RoboCup Mixed Reality system. Figure taken from [1].

The infrastructure of MR competition consists of: (a) camera and vision-tracking (VT) system which guarantee the capture of robots position and movement direction in the field, (b) infra-red (IR) transmitters located in the corners of the field driven by a Robot Control (RC) system and receivers in the robots which guarantee the command of the robots, and (c) a projection system comprised by an LCD monitor and software that projects the soccer field, as well as the moving objects: ball and robots, in the LCD screen. The LCD screen is put in horizontal position allowing the physical robots to move over its flat surface. The real robots interact each other and with virtual objects displayed on the screen [1]. In this infrastructure, the server (MR-SoccerServer [2]) is responsible for the simulated environment and virtual objects, and supervises the soccer game. The robots are controlled by software agents, the infrared transmitters and the camera are the actuators and sensors in real environment (Fig. 1). MR-Simulator communicates with the MR-Soccerserver emulating the IR-RC and VT tracking systems.

Among the features of MR-Simulator there are: simulation of IR-RC and VT (connection and messages transmission) systems, real robots simulation (calculation of trajectories and collisions) and repositioning of the robots in the field. The main features are explained in section 3. Section 2 contains a brief review of the RoboCup Mixed Reality competition history and describes some mixed reality aspects. In section 4 we discuss the results of tests with the MR-Simulator. The conclusions and future works are addressed in section 5.

2 The RoboCup Mixed Reality Competition

The Mixed Reality competition is part of the simulation league of RoboCup. It was proposed in RoboCup 2006 under the name Physical Visualization (PV) [3]. Its goal is to provide a bridge between simulated and physical leagues, using the concepts of mixed reality [1,4,5].

Mixed reality is defined as a mix between real and virtual, or the overlap of the virtual world with the physical. It is divided into augmented reality and augmented virtuality in Milgram's real-virtual continuum [6,7]. Augmented reality is the insertion of virtual objects in the real world, while augmented virtuality is the insertion of real objects into the virtual world. The mixed reality that occurs in the competition MR is augmented virtuality, where robots (real objects) are inserted in a virtual environment composed by field, ball and football beams simulated by the server of the competition.

The MR complete environment set is formed by a high resolution camera, infra-red transmitters, a flat screen or projector, where field and virtual objects are seen, and the robots [1]. Robots are identified on the field by the camera, and their positions and orientations are sent to the VT, and then to the server. The server sends data about all real objects on the field to the clients, which process their decisions and pass the desired wheel velocities (varying from 0 to 130mm/s, positive, negative) back to the server, so it can send them as commands to the robots on the field through the RC system. Inside the RC, commands are discretized based on Table 1, and sent to robots through the infra-red transmitters. The micro-robot currently used in MR competitions has small size (28x25x27 mm) having an almost cube shape (Fig. 2).

Since its creation the MR competition has been gradually evolving, both in software and hardware. However, still remain some platform development challenges. For example VT, RC and even the robots used are too much unstable. We have observed that VT can lose robot's positions due to calibration errors or improper lighting. In some cases

Table 1. Speeds's Discretization. Values taken from [8].

Code speed in mm/s		Code speed in mm/s	
0	0	16	44.80
1	25.61	17	47.15
2	27.17	18	49.77
3	28.54	19	52.65
4	29.72	20	55.81
5	30.76	21	59.24
6	31.71	22	62.95
7	32.59	23	66.96
8	33.48	24	71.33
9	34.39	25	76.19
10	35.39	26	81.78
11	36.51	27	88.59
12	37.77	28	97.48
13	39.21	29	110.16
14	40.84	30	130.43
15	42.70	-	-

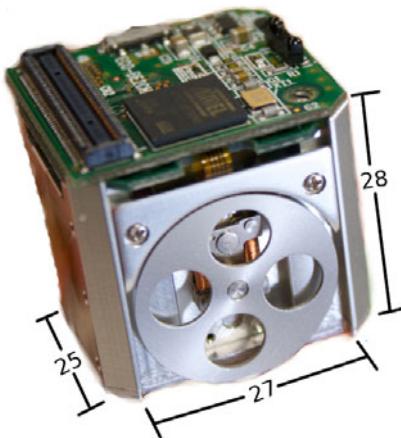


Fig. 2. MR micro-robot and its dimensions (in milimeters)

RC it was not able to pass commands to all robots due to infra-red interference. However, this problem has been solved by increasing the number of infra-red transmitters around the field. Moreover, robots have both hardware and software problems: robots can translate and/or execute erroneously sent commands. In spite of the effort of the participating teams in solving those problems, since 2009 the MR was presented only as a Demo Competition on RoboCup.

3 The MR-Simulator

The motivation for developing the MR-Simulator emerged from the difficulties to work with such still unstable mixed platform. MR soccer team developers need to test tactics and strategies in the field in order to improve the defensive and offensive performance of their teams, and that is not really possible with the current infrastructure. Authors believe that MR-simulator provides an stable environment for team training and testing, as well as could allow the expansion of the competition providing a cheaper environment. Using MR-Simulator it is possible to dispense robots, infra-red transmitters (RC), camera (VT) and big sized monitors or projectors, which can be an alternative for beginners and/or for preliminary tests. Besides the relatively high cost of such infrastructural items there is also an economy in time and resources needed for support and maintenance of such infrastructure, when using MR-Simulator. The MR fully simulated environment makes possible a faster spread of the competition, widening educational applications possibilities, one of the league's primary goals - together with gradual platform development [9,10,11].

Nowadays, we can find good simulators that could be used in the MR competition, for example Webots [12] and Spark [13]. Both are based on the same simulation engine Open Dynamics Engine [14] used by MR-SoccerServer. ODE provides software objects to model physical objects and joints and to handle collisions which makes it suitable for

simulating also 3D competition. However, in our case we decided to develop a simulator free of dependencies with any simulation library. Two main reasons justified that decision: (1) ODE is still incomplete and needs substantial improvements in the documentation and, (2) we aimed to construct simplified kinetic models that better simulates the movement and collisions of the robot-players in the real MR soccer environment.

MR-simulator is more focused on competitor's needs and expectations. Spark, for instance, offers many features that are unnecessary for MR, and is in constant evolution, which requires frequent readjustments of the competition structure in order to maintain the compatibility. It should be said that a first attempt was done trying to use the 2D competition server (Soccer Server [15]) for MR simulations, however it was difficult to control agent's behavior under the classical noisy environment of the 2D competition.

3.1 Simulation of the Environment

In order to simulate VT and RC, virtual modules were created for message exchange with the server: the virtual vision tracking (V-VT) and the virtual Robot Control (V-RC). Messages sent by V-VT are based on an initial robot position and on trajectory and collision models, explained in section 3.2. V-VT sends messages containing virtual position and orientation of robots to the server, which passes client commands to the V-RC. V-RC discretizes the velocities according with Table 1 and sends them to the virtual robots. Trajectory and collision models are used to compute robot's next positions and restarts the cycle, until the game ends, as can be seen on Algorithm 1. In our implementation the MR-SoccerServer interface was not modified, that is, the server behaves in the same way that when communicates with real VT and RC.

3.2 The Robot's Simulation

As the camera is located on top of the field, the height of the robot is neglected in the simulation. The robot is modeled as a flat figure of 25 x 27 mm.

The robots are simulated with virtual robots which have name, ID, orientation, size and wheels speed, based on data passed by the agents. The robots simulation is made with the trajectory and collision models.

Algorithm 1. Main Loop of MR-Simulator

```

while !EndGame() do
    if RC.ReceiveRobotsVel() then
        for Robots.begin : Robots.end do
            UpdateVelRobots()
            ModelCollisionRobot – Robot()
            ModelCollisionRobot – Wall()
            ModelTrajectory()
        end for
    end if
end while

```

Trajectory Model. Our model considers two discretized and coupled time scales for generality and numerical stability purposes. Consider first a succession of cycle times $t_i = t_0, t_1, \dots, t_n$, such that $t_{i+1} = t_i + \Delta T_c$, at which the velocities $v_{L,i}$ (left) and $v_{R,i}$ (right) of the wheels of the robot can be updated via the RC system. Here ΔT_c stands for the “cycle time” which depends on the server configuration. Within a time interval $I_i = [t_i, t_{i+1})$ the velocities of both wheels are kept constant and equal to the velocity set at $t = t_i$, that is, $v_L(t) = v_{L,i}$ and $v_R(t) = v_{R,i}$ for $t \in I_i$. However, the position of the wheels are updated at smaller time steps $\tau_k = \tau_1, \tau_2, \dots, \tau_m$ such that $\tau_1 = t_i$ and $\tau_m = t_{i+1}$. That is, we set $\tau_{k+1} = \tau_k + \Delta T_s$, for $k = 1, 2, \dots, m$ where $m = \Delta T_c / \Delta T_s$ is an integer greater or equal 1. ΔT_s is a prefixed “simulation time step” adjusted for obtaining the desired precision in the simulation and is subject to a natural constraint $\Delta T_s \leq \Delta T_c$. Therefore, during a cycle time interval m simulated displacements of the robot must be calculated. Let’s denote as $(x_{L,k}, y_{L,k})$ and $(x_{R,k}, y_{R,k})$ the position of the left and right wheels, respectively, of the robot at a given simulated time τ_k in a x, y plane domain representing the football field. Let a_0 be the distance between wheels and f a correction factor. At the beginning of each server time interval I_i , $i = 1, 2, n$ set $(x_{L,1}, y_{L,1})$ and $(x_{R,1}, y_{R,1})$ according with the current position of the left and right wheel, respectively, sent by the V-VT system. Then, using the velocities $v_{L,i}$ and $v_{R,i}$ sent by the agent for this cycle, compute

$$e_y = y_{R,1} - y_{L,1}$$

$$e_x = x_{R,1} - x_{L,1}$$

$$c_t = e_y / a_0$$

$$s_t = -e_x / a_0$$

$$m = \Delta T_c / \Delta T_s$$

and then for $k = 1, 2, \dots, m$ do:

1. Calculate predicted wheel position in the next simulation time step

- $x_{L,k+1} = x_{L,k} + v_{L,i} c_t \Delta T_s$
- $x_{R,k+1} = x_{R,k} + v_{R,i} c_t \Delta T_s$
- $y_{L,k+1} = y_{L,k} + v_{L,i} s_t \Delta T_s$
- $y_{R,k+1} = y_{R,k} + v_{R,i} s_t \Delta T_s$

2. Compute correction terms

- $e_y = y_{R,k+1} - y_{L,k+1}$
- $s_y = \text{sign}(e_y)$
- $e_x = x_{R,k+1} - x_{L,k+1}$
- $s_x = \text{sign}(e_x)$
- $e_a = \sqrt{e_y^2 + e_x^2} - a_0$
- For velocity dependent correction set $f_R = v_{R,i} / (v_{R,i} + v_{L,i})$ else $f_R = f$
- $f_L = 1 - f_R$

3. Calculate corrected wheel position in the next simulation time step

- $x_{R,k+1} = x_{R,k+1} - s_x e_a f_R$
- $x_{L,k+1} = x_{L,k+1} + s_x e_a f_L$
- $y_{R,k+1} = y_{R,k+1} - s_y e_a f_R$
- $y_{L,k+1} = y_{L,k+1} + s_y e_a f_L$

After each cycle MR-simulator plots the virtual robot placing the left and right wheels at coordinates $(x_{L,k}, y_{L,k})$ and $(x_{R,k}, y_{R,k})$, respectively. When the last simulation time interval ends, send to the V-VT the position of the centroid of each robot calculated as the mean of the wheel coordinates, that is $X = 0.5 * (x_{L,m} + x_{R,m})$ and $Y = 0.5 * (y_{L,m} + y_{R,m})$.

Collision Model. The collision model is divided into robot-wall collisions and robot-robot collision. In both types of collision it uses the concept of slip, a reduction or increase in the actual speed of the wheels depending on the type of collision. When MR-simulator detects an immediate future collision scenario, that is, when the body of a robot-player is expected to be not entirely placed within the field (robot-wall collision case) or when it is expected to intersect other robot body (robot-robot collision case) in the next simulation time τ_{k+1} , the kinetics is altered by using modified wheel velocities $v_{L/R,k}^*$ instead of $v_{L/R,k}$, that are the velocities set by the agent without taking into account the collision. To do that, the actual velocity of the wheels are calculated as $v_{L/R,k}^* = (1 - s_{R/L,k})v_{L/R,k}$ where $s_{R/L}$ is an slip factor that formally varies between a negative lower bound and an upper bound greater than or equal to one and is different for each wheel depending on the collision conditions.

Negative slip causes wheel acceleration with respect to its velocity prior collision due to impulse transfer from a colliding object. This occurs when the colliding object was moving faster than the collided object in a non-opposite direction. For example for a slip value equal to -1 , the actual wheel velocity is the double of the expected velocity of the wheel prior collision. A null slip does not change the velocity of the wheel, but as the slip increases toward one the wheel velocity decreases, stopping when slip is equal to one. This happens when the robot collides with an object moving slower in the same direction or with an object moving and colliding from an opposite direction. Therefore, when a wheel is expected to stop its slip factor is set to one, independent of the velocity set by the agent at the beginning of the cycle. That is the case when a robot collides with a virtual field limiting fence. This way the robot is limited to be within the field. Finally when the sleep exceeds one it implies a reversion of the expected rotation of the wheel. The actual reverse velocity depends on the magnitude of the slip. For example, when the slip is equal to two, the actual velocity of the wheel has the same expected value but in opposite direction.

The total slip (sum of the left and right wheel slips $s_k = s_{L,k} + s_{R,k}$) was set as a function of the absolute value of the resultant velocity after collision, which is proportional to the remaining kinetic energy of the colliding objects assumed to stay together forming a single object after collision. In the case of robot-wall collision it depends only on the velocity of the robot. The total slip is then distributed between left and right wheels which depends on the collision geometrical conditions, considering torque.

During collisions the velocities of the wheels of the colliding objects are updated at each simulation time τ_k , that is the wheel velocity is set equal to the modified velocity. Such modified velocity is then used for the next simulation step and a collision status is sent to the agents of the colliding robots to update de finite state machines. The non-collision status is reestablished when the collision algorithm does not predict physical contact of the robots in the next time step.

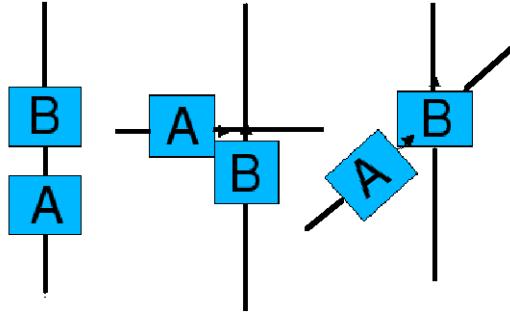


Fig. 3. a) Parallel b) Perpendicular c) Oblique

When simulating robot-robot collision we firstly identify the type of collision considering three categories: (1) parallel, when the robot collides with the front or back side, (2) perpendicular, when collides with a lateral (left or right) side, and (3) oblique when collide with one of its corners, as illustrated in Figure 3. The latter case is currently treated as a parallel collision when the colliding corner bump with another robot's front or back side, and as perpendicular when it collides with the left or right side of another robot. The parallel and perpendicular collisions are treated separately, always analyzing two robots a time: a colliding robot (A) and a collided robot (B).

For example, to find the collision point at robot A we must verify if any of the four vertex of robot B (considering the orthogonal projection of the 3-dimensional robot onto the x, y plane) will be located within the region occupied by the robot A in the next simulation time step. This should be done also for robot B with respect to robot A. In Figure 4 we show an schematic representation of a collision. $V = (x_v, y_v)$ is the colliding vertex and (x_r, y_r) are the coordinates of the collision point, with respect to a reference system located in the center of mass of the collided robot. In that figure $b = a_0$ is the size of the robot and θ is the angle of rotation of the reference system. Therefore a vertex (x_v, y_v) is inside a robot with centroid located at x_c, y_c if it satisfies equations 1 and 2 [16].

$$-a_0/2 \leq x_r \leq a_0/2 \quad (1)$$

$$-a_0/2 \leq y_r \leq a_0/2 \quad (2)$$

where

$$x_r = \sqrt{(x_v - x_c)^2 + (y_v - y_c)^2} \cos(\tan^{-1}(\frac{y_v - y_c}{x_v - x_c}) - \theta)$$

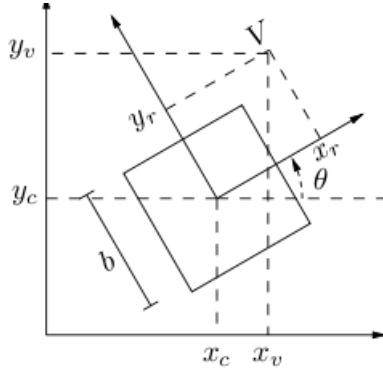
and

$$y_r = \sqrt{(x_v - x_c)^2 + (y_v - y_c)^2} \sin(\tan^{-1}(\frac{y_v - y_c}{x_v - x_c}) - \theta)$$

With these formula we check whether some vertex of A collides with B and/or some vertex of B collides with A, defining the collision type point using the rules in Table 2.

Table 2. Defining the collision point

Vertex of A bumps into B	Vertex of B bumps into A	Collision point
True	False	where A intersects B
False	True	where B intersects A
True	True	middle point of A and B vertices
False	False	no collision case

**Fig. 4.** Relevance of a point to a square. Figure taken from [16].

When both robot A and B have a vertex colliding into the other robot it indicates parallel or perpendicular collision. Otherwise, an oblique collision is taking place.

After a robot-robot collision state is predicted (the simulator predicts that collision will happen in the next simulation time) or confirmed (the collision began in a previous simulation time but the robots are still in contact) we calculate the after-collision velocity C as the average vectorial velocity of the two robots A and B (3).

$$C = \frac{V_A + V_B}{2} \quad (3)$$

The total slip, S , in parallel collision case for both robots, A and B, are defined as the absolute difference of the colliding robot speed ($V_X = V_A$ or V_B) and the after-collision velocity C with respect to the robot speed, that is.

$$S_X = \frac{\|V_X - C\|}{\|V_X\|} \quad (4)$$

Parallel Collision. The total slip S_X of each robot $X = A$ or B is distributed between left and right wheels, considering the mechanical torque (T), that tends to rotate or flip objects. The torque in this case is defined by the vectorial product

$$T_X = r_X \times (V_A - V_B) \quad (5)$$

where r_X is a vector pointing from the centroid of robot $X = A$ or B to the collision point, calculated as described above.

Let us denote by Y and \bar{Y} two different states of the robot wheels which depends on the relative position and motion direction of the colliding robots, in such a way that when one wheel of the robot is in state Y the other necessarily is in state \bar{Y} . The wheel that is in state Y is always that being in the right side of the robot with respect to the direction of motion. Thus, when the robot moves forward (positive velocity) the right wheel is in state Y and the left wheel in \bar{Y} state. When the robot moves backward (negative velocity) the left wheel is in state Y and the right wheel is in state \bar{Y} . The slip calculation described below depends only on the state of the wheel.

The slip of the wheel in state Y of robot $X = A$ or B is defined by

$$S_{Y,X} = \frac{1 + \|T_X\|}{2 \|T_{max}\|} \quad (6)$$

where T_X is given by equation 5 and T_{max} is the maximum expected torque given by

$$T_{max} = r_{max} \times (V_A + V_B) \quad (7)$$

where r_{max} is the distance of the centroid of the robot to a corner. The slip of the other wheel of the robot X (being in state \bar{Y}) is then given by

$$S_{\bar{Y},X} = 1 - S_{Y,X} \quad (8)$$

The modified speed of the wheel being in a state $W = Y$ or \bar{Y} of the robots $X = A$ and B is in this case calculated as

$$v_{W,X}^* = (1 - S_{W,X}) C \quad (9)$$

and the modified robot velocity is then given by

$$V_X^* = 0.5 (v_{Y,X}^* + v_{\bar{Y},X}^*) \quad (10)$$

Perpendicular Collision. It was observed that in most cases when a perpendicular collision occurs the colliding robot (A) and the collided robot (B) depict different behaviors. Robot A rotates over its own axis following the movement of robot B which rotates

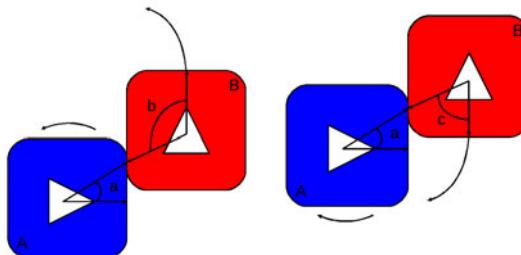


Fig. 5. Perpendicular Collision. 'a' is the collision angle of robot A, and 'b' and 'c' are the collision angles of robot B for positive and negative velocity, respectively.

around robot A, as shown in Figure 5. The orientation of robot B defines to which side robot A will turn. This rotating coupled motion usually continues until robot A retreats.

To simulate this movement within the slip model, we need to take into consideration two variables: (1) Robot's orientation, identifying which robot collided with the front or back side (A) and which collided with a lateral side (B), and (2) the angle of collision β for each robot. β is the angle spanned by the vector joining the center of mass of the robot and the collision point (middle point of the projected contact surface of the robots) and the vector of motion originated also in the center of mass, as indicated in Figure 5. Angles spanned clockwise with respect to the motion direction are assumed to be positive. Counterclockwise are considered negative angles.

For colliding robot A the slips are determined as linear functions of the collision angle $\beta_A \in [-\pi/4, \pi/4]$. The slip of the wheel in state Y is defined by equation 11,

$$S_{Y,A} = 1 - \frac{\sigma}{2} \left(1 + \frac{\beta_A}{\pi/4} \right) \quad (11)$$

where $\sigma = \text{sign}(V_A \times V_B) = \{-1, +1\}$. Notice that $S_{Y,A} \in [0, 2]$. The slip of the opposite wheel (in state \bar{Y}) is calculated by equation 12

$$S_{\bar{Y},A} = S_{Y,A} + \sigma. \quad (12)$$

For illustration, in the left part of Figure 6 we represented six cases corresponding to angles $\beta_A = \{-\pi/4, 0, \pi/4\}$ (from top to bottom) and $\sigma = \{-1, +1\}$ (left and right). The collision point is represented with a blue spot inserted in the collided robot surface (side). The arrows in the wheels represent the modified velocities after collision. Prior to collision both wheels were moving with the same velocity. In the right part of the same Figure 6 we plotted the slip of both wheels (Y , \bar{Y}) for positive and negative σ as a function of β , using Eq. 11 and 12.

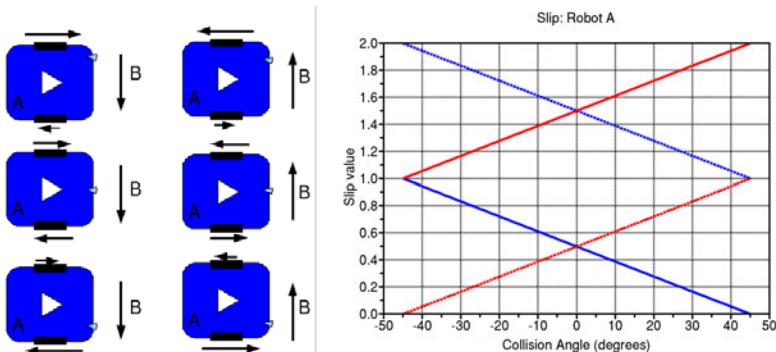


Fig. 6. LEFT: Wheel velocities of robot A in 3×2 typical perpendicular collision cases. From top to bottom the collision angles are $\beta_A = \{-\pi/4, 0, \pi/4\}$. In the left column cases $\sigma = -1$ and $\sigma = +1$ in the right column cases. RIGHT: Slip setting as a function of β_A , for positive (blue) and negative (red) σ . Continuous and dashed lines correspond to Y and \bar{Y} wheels, respectively.

The slip of the wheel W in the collided side of robot B was modeled with a quadratic function of the absolute value of the collision angle $|\beta_B| \in [\pi/4, 3\pi/4]$ in the form

$$S_{W,B}(\beta_B) = c_0 + c_1 |\beta_B| + c_2 |\beta_B|^2, \quad (13)$$

while the velocity of the wheel in the other side, \bar{W} , was not modified in this case, that is, we set $S_{\bar{W},B} = 0$. The coefficients in Eq. 13 were calculated to satisfy $S_{W,B}(\pi/4) = 1$, $S_{W,B}(\pi/2) = 1/3$ and $S_{W,B}(3\pi/4) = 0$.

For illustration we represented in the left part of Figure 7 the velocities of the wheels of the collided robot B in six typical perpendicular collision cases. In the right part the slip of the wheel in the bumped side is plotted as a function of $|\beta_B|$ using Eq. 13.

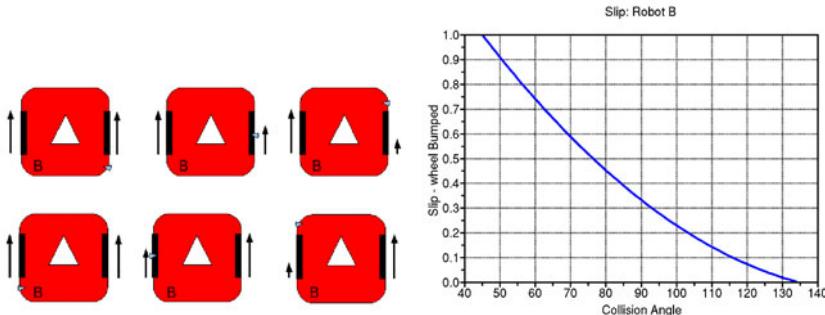


Fig. 7. LEFT: Wheel velocities of robot B when collided from the right (top) and from the left (bottom) for collision angles $\beta_B = \{\pm 3\pi/4, \pm\pi/2, \pm\pi/4\}$ varying from left to right. RIGHT: Bumped wheel slip as a function of $|\beta|$.

4 Discussion of Results

MR-Simulator was tested as the software modules were released as follows: VC and RC virtual modules connection; VT and RC virtual modules message transmission; Trajectory model; Collision model; and Robots repositioning in field.

We tested using an autonomous client, the BahiaMR [17], to verify the behavior of the team using fully simulated environment comparing its behavior with that observed using real mixed reality environment (videos of competition official matches).

4.1 Connection Tests

Tests of connection and transmission of messages from virtual modules and repositioning were a jectory done successfully, checking whether connections were made correctly, or messages delivered flawlessly at the right time and if the robots were repositioned in the correct positions and orientations.

4.2 Kinetic Model Test

The kinetic model that predicts the trajectory of the robot was tested using a code written in Scilab [18]. The kinetic model test program consisted on the evaluation of the accuracy of wheel positions for different simulation time steps. As a result a simulation time step ΔT_s of 0.001 seconds was chosen. After approval of the kinetic model a C++ version was created and built within the MR-simulator.

The collision module was directly written in C++ within the MR- simulator environment. The test and parameter adjustment program consisted on: (1) Evaluation of dependence on collision intensity; (2) Calculation of the total slip factors with different options; and (3) Calculation of wheel slip factors on parallel and perpendicular collisions. As a result the formulas described above were chosen.

4.3 Test with Clients

The tests with the client were made by analyzing its behavior in a totally simulated environment. The clients were firstly programmed to perform simple trajectory movements. These movements were observed compared with those showed in the competition videos. Then, robots were programmed to perform movements that resulted in collisions. The collisions were made in various angles and with different robot speeds. The movements performed by clients in the tests were go forward, go back, go to the ball, turn around the axis (spin) and make curves to trajectory model and execute parallel, perpendicular and oblique's collision in different directions, angles and speeds to collision model.

The result was a behavior similar to those seen in the videos. In these tests, we observed great stability even when the field size and number of robots per team was changed. That makes it possible to simulate the game regardless of field size and number of robots, different from what happens today in the competition, which runs 4x4 to 6x6 robots matches over a 42" flat screen. In Figure 8 it is possible to observe simulations with 5 players per team and with 11 players per team.

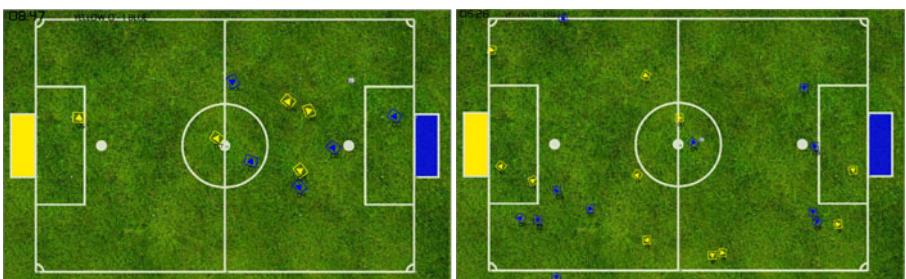


Fig. 8. Left: simulation with 5 players per team Right: simulation with 11 players per team

5 Conclusions and Future Work

This paper presented the MR-simulator, a tool to aid team development of RoboCup Mixed Reality competition and facilitate league's expansion. We described its features and mathematical models used to compute robot's trajectory and collision events. The MR-Simulator has been used since the last official competition, RoboCup 2009, held in June and July 2009 in Graz, Austria.

In future works we will address the implementation of a graphical interface for the simulator itself, to serve as viewer and improve the way of repositioning the robots. Moreover, we plan to develop features to control the robot manually in a debug mode; nowadays robots are controlled only by autonomous clients. Controlled noises, usual on real environments will also be added, in a configurable way, affecting messaging, positioning and robots movement (e.g. wheels locks) in order to simulate more realistic scenarios.

References

1. Gerndt, R., Schridde, C., da Silva Guerra, R.: On the aspects of simulation in the robocup mixed reality soccer systems. In: Workshop at International Conference on Simulation, Modeling and Programming for Autonomous Robots (2008)
2. Silva, G., Cerqueira, A.J., Reichow, J.F., Pimentel, F.A.M., Casaes, E.M.R.: MR-SoccerServer: Um Simulador de Futebol de Robôs usando Realidade Mista. In: Workshop de Trabalhos de Iniciação Científica e de Graduação (WTICG), pp. 54–63 (2009)
3. Guerra, R., Boedecker, J., Yamauchi, K., Maekawa, T., Asada, M., Hirosawa, T., Namekawa, M., Yoshikawa, K., Yanagimachi, S., Masubuchi, S., et al.: CITIZEN Eco-Be! and the RoboCup Physical Visualization League (2006)
4. Stapleton, C., Hughes, C., Moshell, J.: Mixed reality and the interactive imagination. In: Proceedings of the First Swedish-American Workshop on Modeling and Simulation, pp. 30–31 (2002)
5. Azuma, R.: A survey of augmented reality. *Presence-Cambridge Massachusetts-* 6, 355–385 (1997)
6. Milgram, P., Takemura, H., Utsumi, A., Kishino, F.: Augmented reality: A class of displays on the reality-virtuality continuum. In: Telemanipulator and Telepresence Technologies, vol. 2351. SPIE, San Jose (1994)
7. Milgram, P., Kishino, F.: A taxonomy of mixed reality visual displays. *IEICE Transactions On Information And Systems E Series D* 77, 1321–1321 (1994)
8. SourceForge: Mixed reality (2010),
<http://mixedreality.ostfalia.de/index.php/topic,73.0.html>
(retrieved January 28, 2010)
9. da Silva Guerra, R., Boedecker, J., Mayer, N., Yanagimachi, S., Hirosawa, Y., Yoshikawa, K., Namekawa, M., Asada, M.: Introducing physical visualization sub-league. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) *RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI)*, vol. 5001, pp. 496–503. Springer, Heidelberg (2008)
10. da Silva Guerra, R., Boedecker, J., Mayer, N., Yanagimachi, S., Hirosawa, Y., Yoshikawa, K., Namekawa, M., Asada, M.: Citizen eco-be! league: bringing new flexibility for research and education to robocup. In: *Proceedings of the Meeting of Special Interest Group on AI Challenges*, vol. 23, pp. 13–18 (2006)

11. da Silva Guerra, R., Boedecker, J., Mayer, N., Yanagimachi, S., Ishiguro, H., Asada, M.: A New MiniRobotics System for Teaching and Researching Agent-based Programming. In: Computers and Advanced Technology in Education, Beijing (2007)
12. Michel, O.: WebotsTM: Professional mobile robot simulation. Arxiv preprint cs/0412052 (2004)
13. Obst, O., Rollmann, M.: Spark-a generic simulator for physical multi-agent simulations. Computer Systems Science and Engineering 20(5), 347 (2005)
14. Smith, R., et al.: Open Dynamics Engine. Computer Software (2010), <http://www.ode.org> (retrieved January 28, 2010)
15. Noda, I., Noda, C., Matsubara, H., Hiraki, K., Frank, I.: Soccer server: A tool for research on multiagent systems. Applied Artificial Intelligence (1998)
16. Pedrosa, D.P.F., Yamamoto, M.M., Alsina, P.J., de Medeiros, A.A.D.: Um simulador dinamico para mini-robos moveis com modelagem de colisoes. In: VI Simposio Brasileiro de Automação Inteligente (Setembro) (2003)
17. Simões, M., da S Filho, J., Cerqueira Jr. A., Reichow, J., Pimentel, F., Casaes, E.: BahiaMR 2008: Descrição do Time (2008)
18. Scilab: Scilab home page (2010), <http://www.scilab.org/> (retrieved January 28, 2010)

Learning Footstep Prediction from Motion Capture

Andreas Schmitz, Marcell Missura, and Sven Behnke

University of Bonn,
Computer Science VI, Autonomous Intelligent Systems
Roemerstr. 164, 53117 Bonn, Germany
{schmitz,missura,behnke}@ais.uni-bonn.de
<http://ais.uni-bonn.de>

Abstract. Central pattern generated walking for bipedal robots has proven to be a versatile and easily implementable solution that is used by several robot soccer teams in the RoboCup Humanoid Soccer League with great success. However, the forward character of generating motor commands from an abstract, rhythmical pattern does not inherently provide the means for controlling the precise location of footsteps. For implementing a footstep planning gait control, we developed a step prediction model that estimates the location of the next footstep in Cartesian coordinates given the same inputs that control the central pattern generator. We used motion capture data recorded from walking robots to estimate the parameters of the prediction model and to verify the accuracy of the predicted footstep locations. We achieved a precision with a mean error of 1.3 cm.

Keywords: footstep planning, dynamic walking, machine learning, motion capture.

1 Introduction

Central pattern generator based methods and inverse kinematics based methods are two successful approaches to implement controlled dynamic walking for bipedal robots, even though they differ in their core aspects. Central pattern generator based methods [1,2] generate an abstract, periodic signal stream which is translated to motor commands resulting in rhythmical weight shifting and leg swinging motions. Inverse kinematics based solutions [3,4,7] precalculate trajectories in Cartesian coordinates for significant body segments, such as the pelvis and the feet. These trajectories are converted to motor commands by solving the inverse kinematics with the given trajectories as constraints. In the latter case the footstep locations are known in advance: they are determined by the intersections of the foot trajectories with the ground. In the former case, however, footstep locations are not inherently obtainable, since they are indirect results of amplitudes and frequencies of abstract signal patterns.

Our goal is to predict the footstep locations of a central pattern generated walk and to use the predictions to implement a more precise, footstep planning gait

control. We present two different approaches to estimate the footstep locations and compare their performance in experimental results.

The remainder of the paper is organized as follows. After reviewing related work, a brief introduction to our gait engine is given in Section 3. Then an overview of the footstep prediction algorithm is presented in Section 4, leading to the detailed descriptions of the two different approaches we implemented: a forward kinematic based approach in Section 5 and a motion capture based approach in Section 6.

2 Related Work

Footstep planning is a fairly new research topic and feasible solutions are scarce in comparison. The most prominent proposals in [8,9,10] and also [11] are based on the A* algorithm. By imposing a strong discretization on the state space and using only a small, discrete set of actions, these online solutions plan a few steps ahead and are able to deal with dynamic environments. Uneven floor plans are also considered, so that the footstep plans can include stepping over obstacles and climbing stairs. An intriguing alternative solution has been recently shown in [12]. Here, a short sequence of future footsteps is considered to be a virtual kinematic chain as an extension of the robot. Their location is determined using inverse kinematics. The configuration space and the action space are not discretized, but the algorithm is computationally expensive. A computationally more promising method that can plan in a few milliseconds, if the environment is not too cluttered, has been suggested in [13]. The idea is to solve the footstep planning problem mostly with a path planning algorithm. Actual footstep locations are only given in key points, where the walking speed of the robot has to be zero, for example when stepping over an obstacle. The majority of the footstep locations are layed out along the planned paths by the motion generator developed for HRP-2 [3,6,5]. The closest related work is [10], where an A* based footstep planning algorithm is adapted for the humanoid robot ASIMO. As the walking algorithm of ASIMO is not precisely known, the authors were forced to reverse engineer a footstep prediction algorithm from observations with a motion capture system.

3 Central Pattern Based Dynamic Walking

In this chapter we introduce the basic concepts of our central pattern based gait generation method in a simplified level of detail. We concentrate on the core modules that are important to understand the footstep prediction model.

3.1 Leg Interface

The leg interface is a low level abstraction layer that allows intuitive control of a leg with three parameters. The leg angle Θ_{Leg} defines the angle of the leg with respect to the trunk, the foot angle Θ_{Foot} defines the inclination of the foot with

respect to the transversal plane, and the leg extension η defines the distance between the foot and the trunk (Figure 1). The output of the leg interface are joint angles for the hip, knee and ankle. The leg interface allows independent control of the three parameters and encapsulates the calculation of coordinated joint angles.

$$L(\Theta_{Leg}, \Theta_{Foot}, \eta) = (\Theta_{Hip}, \Theta_{Knee}, \Theta_{Ankle}) \quad (1)$$

The leg can be bent in roll, pitch, and yaw directions as illustrated in Figure 1 (right). The three directional components can be controlled independently from each other. Most importantly, the foot is rotated around its own center and not around the trunk. More detailed information can be found in [1].

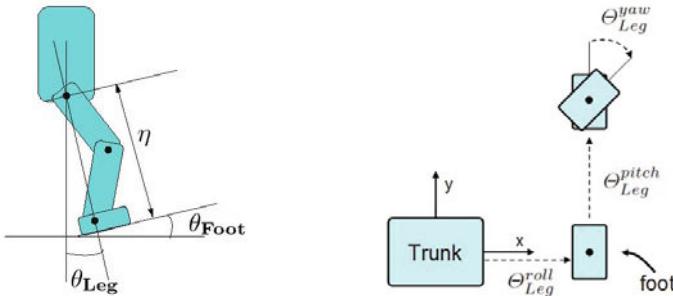


Fig. 1. The leg interface allows independent control of three abstract parameters: the leg angle Θ_{Leg} , the foot angle Θ_{Foot} , and the leg extension η (left). The leg can be bent independently in roll, pitch, and yaw directions (right).

3.2 Central Pattern Generator (CPG)

The CPG generates patterns of rhythmic activity governed by a periodic internal clock called gait phase $-\pi \leq \phi < \pi$. The patterns encode the waveforms of the leg interface parameters. In particular, the leg extension is activated with a sinusoidal function, whereas the phase of the left leg is shifted by π with respect to the right leg (Figure 2, left).

$$P_w = \sin(\phi) \quad (2)$$

The antidromic shortening and extending of the legs causes a rhythmic lateral shift of the body weight alternatingly freeing a leg from its support duty. This leg can be swung. In concert with the leg extension signal P_w , the CPG generates a second activation to swing the free leg.

$$P_s = \sin(\phi - \frac{\pi}{2}), \quad -\pi \leq \phi < 0 \quad (3)$$

$$P_s = 1 - \frac{\phi}{\pi}, \quad 0 \leq \phi < \pi \quad (4)$$

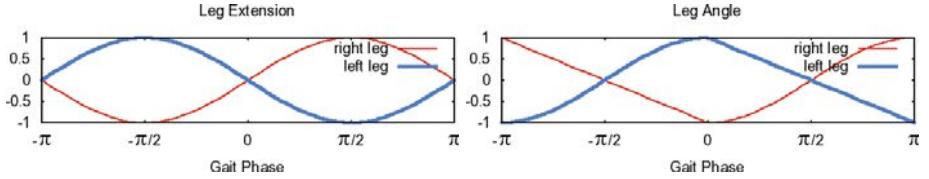


Fig. 2. The CPG waveforms encode the leg extension (left) and leg swing (right) patterns

As shown in Figure 2 right, the leg is swung forward with a sinusoidal motion and pushed back with a linear motion in the support phase. Support exchange is expected to occur at gait phase $\phi = 0$ from right to left and at gait phase $\phi = -\pi$ from left to right.

3.3 Omnidirectional Gait Control

Walking direction and step size are controlled by modulating the amplitude of the leg swing activation P_s with a gait control vector $g \in [-1, 1]^3$ and applying it to the roll, pitch and yaw component of the leg angle Θ_{Leg} . Omnidirectional walking is achieved by applying the swing signal in all three directions simultaneously with different intensities. For example, a mixture of the pitch and yaw components will result in a curved walk forward, where the yaw intensity determines the curvature of the path. The modulated signals are then transformed by a configuration vector $c \in \mathbb{R}^3$, which is a mapping from CPG signal space to leg angle space expressed in radians. c can be used to adapt the very same CPG patterns to robots of different sizes, for example from the KidSize and the TeenSize class and to fine tune individual robots. In summary, these equations describe the generation of the gait trajectory:

$$\Theta_{Leg}^{roll} = P_s \cdot g_x \cdot c_x + |g_x| \cdot c_x \quad (5)$$

$$\Theta_{Leg}^{pitch} = P_s \cdot g_y \cdot c_y \quad (6)$$

$$\Theta_{Leg}^{yaw} = P_s \cdot g_z \cdot c_z + |g_z| \cdot c_z \quad (7)$$

At the end of the gait control chain the leg interface converts the leg angles and extensions to joint angles, as depicted in Figure 3.

Notably, the equation in sagittal direction (6) differs from the lateral and yaw directions (5,7). In sagittal direction the legs are encouraged to swing fully from front to back. Positive and negative leg angles are likewise allowed. In lateral direction, however, the legs would collide. To avoid negative leg angles a positive value is added to the leg roll angle proportionally to the lateral component of the gait control vector causing the legs to spread out when walking in lateral direction. As a result, two different step sizes occur: a long step, when the leading leg is swung in the direction the robot is moving and a short step, when the other leg is pulled in to meet the leading leg at a leg angle close to zero. The same is true for the yaw direction.

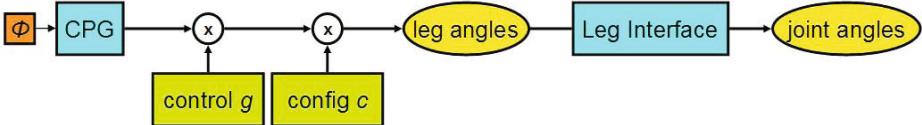


Fig. 3. The gait trajectory generation chain. The CPG is always active and produces periodic activation signals clocked by the gait phase ϕ . After modulation by the gait control g , the signals are mapped to leg angle space with the configuration c . The leg interface converts the leg angles to joint angles that are passed to the robot.

4 The Step Prediction Models

For the implementation of a step planning algorithm, a forward model

$$F(g_x, g_y, g_z, \phi) = (p_x, p_y, p_\theta) \quad (8)$$

is required that maps a gait control vector g to an expected footstep location and orientation $p \in \mathbb{R} \times \mathbb{R} \times [-\pi, \pi]$ in Cartesian coordinates. We define p to be the location and orientation of the footstep in the local coordinate frame of the support foot. As outlined in the previous Section, a gait control vector g can produce two different step sizes. The gait phase ϕ has to be included as parameter for reasons of disambiguation. The value of ϕ is determined by keeping track of the support leg and knowing at which gait phase the next step will occur.

We developed two strategies to obtain this mapping. As an analytic approach we used a kinematic model of the robot to calculate the forward kinematics from given joint angles. Alternatively, we collected training data from a motion capture device and used linear regression to learn the mapping F . Both approaches are described in detail in the following sections.

5 Kinematic Model Approach

The kinematic model requires a precise skeleton of the robot that we acquired from the CAD construction blueprints. Predictions are made by applying the joint angles in the moment of a step to the kinematic model and calculating the position and orientation of the swing foot relative to the support foot in Cartesian space (Figure 5). To get hold of the joint angles in the moment of a step for a specific gait control vector g , we set the appropriate gait phase ϕ (0 or $-\pi$) and execute the gait trajectory generation chain (Figure 3). The CPG will output the signals that it would produce in the moment of a step. Using the configuration c of a specific robot and the gait control vector g , we acquire the desired joint angles.

Due to their analytic nature, the predictions can be calculated very efficiently. However, some sources of error are inevitable. Inaccuracies in the skeleton cannot be completely avoided as well as the fact that the robot does not perfectly obey the commanded joint angles. There is always mechanical wear, backlash in the

gears and undesired elasticities that cause the physical system to deviate from theory. We decided to take an alternative approach and learn from data collected from the actual physical system. This approach is presented in the next Section.

6 Machine Learning Approach

As an alternative course, we collected ground truth data with a motion capture system to learn from the footsteps as they really happen. Two KidSize robots were equipped with reflective markers in groups of three or four on the head, the hip, and the feet, as shown in Figure 4. With both robots we recorded approximately five minutes of more or less random walking speeds and directions, trying to explore the entire gait control space. The output of the motion capture device are trajectories of the reflective markers, which we synchronized with a recording of the gait control vector. In data post processing, we calculated the centroids of every marker group to represent the head, the hip and the feet with only one point. The centroids were used for further post processing. From the hip marker group we calculated the orientation of the robot with respect to the global vertical axis. The orientation describes in which direction the robot is facing in the world coordinate frame, but this is not necessarily equal to the walking direction. Using the feet marker groups we also computed the orientation of both feet relative to the global orientation of the robot.



Fig. 4. Groups of reflective markers were used to indicate the head, the hip and each of the feet

To produce the training data, single steps had to be identified. We used the feet centroids to extract two features: the height of the feet h_l and h_r and the velocities of the feet v_l and v_r calculated from two consecutive frames. A step is recognized when the feet have approximately the same height and the same velocity:

$$|h_l - h_r| + |v_l - v_r| < 0.01 . \quad (9)$$

Altogether we identified approximately 3000 footsteps and matching gait control vectors. Figure 5 shows a visualization of the marker cloud, the kinematic model fitted into the cloud, the orientation of the trunk and the feet, and some of the extracted footstep locations on the floor.

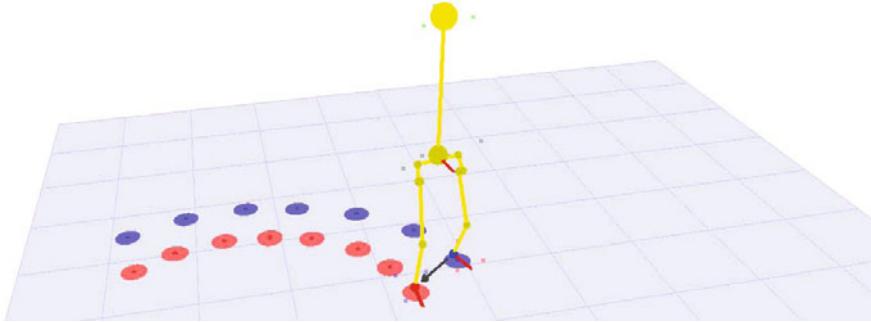


Fig. 5. Visualization of the data obtained from the motion capture device: the marker cloud, the kinematic model fitted into the cloud and performing a step, the orientation of the robot and the feet, and some of the extracted footstep locations on the floor. The arrow between the feet illustrates the vector we extract from the kinematic model.

As mentioned in Section 3.1, the leg interface implements independent control of the footstep location and orientation. This allows the assumption that instead of the three dimensional function F (8), three independent one dimensional functions can be learned. However, when performing a step, the fixed frame of reference is the support foot and not the coordinate frame of the trunk. The g_z component of the gait control vector applies a rotation to the footstep location by an angle α and consequently p_x and p_y both depend on g_z . This is shown in Figure 6 (a). To tackle this problem, we introduce a footstep $q = (q_x, q_y, \alpha)$ in the reference frame of the trunk (Figure 6 (b)). q_x and q_y are defined as the distances between the feet in x and y directions and α is the orientation of the swing foot with respect to the trunk. In this frame of reference q_x depends only on g_x , q_y only on g_y , and α only on g_z . These are the mappings that we learn. We salvage (q_x, q_y, α) from the identified steps in the motion capture data. Knowing the reference frame of the trunk from the hip marker group, we calculate (q_x, q_y) from the difference between the foot coordinates of the fitted skeleton. α is equal to the swing foot orientation that we calculated from the foot marker group. We found the support foot angle and the swing foot angle to be symmetrical enough, so that they do not have to be modeled separately. The trunk oriented footstep q is then transformed to the support foot oriented footstep p with a step transform function T as depicted in Figure 6 (c) and (d). (q_x, q_y) is rotated in the frame of the support foot by the angle α . Then, the swing foot is rotated around its own origin again by the angle α . The transform function T is given by:

$$(p_x, p_y) = (q_x, q_y) \cdot R(\alpha) \quad (10)$$

$$p_\theta = 2\alpha \quad (11)$$

where R denotes a rotation matrix.

Moreover, we implemented another simplification of the learning task. Figure 7 shows a decomposition of function F . Given g and ϕ as input, the gait control chain (Figure 3) can be used to calculate the leg angles at the moment of the next

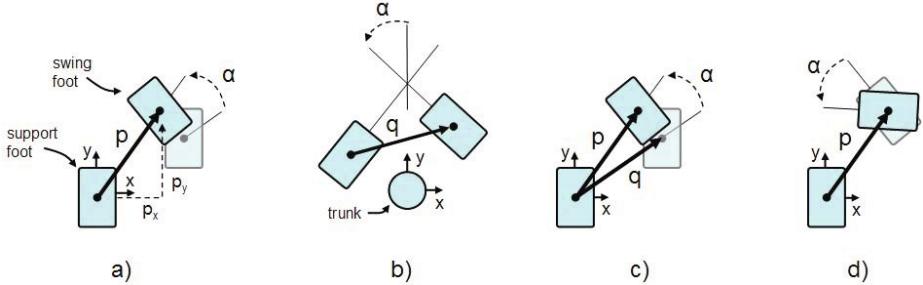


Fig. 6. The g_z component of the gait control vector rotates the footstep location by the angle α around the support foot and influences p_x and p_y (a). The footstep q in the trunk frame, however, is independent from the rotation (b). q is mapped to the support foot related footstep p with a rotation by α (c) and an additional rotation of the swing foot again by the angle α (d).

step. The step function S is the actual physical step. It maps the leg angles to a footstep q in the reference frame of the trunk. The step transform T translates q to the footstep p in the reference frame of the support foot. Only S needs to be learned and can be approximated by three simple, independent, one dimensional functions:

$$q_x = q_x(\Theta_{Leg}^{roll}), \quad (12)$$

$$q_y = q_y(\Theta_{Leg}^{pitch}), \quad (13)$$

$$\alpha = \alpha(\Theta_{Leg}^{yaw}). \quad (14)$$

Using the gait chain as an executable building block also allows us to incorporate different robot configurations just by exchanging them in the gait control chain. We expect better adaptability of the algorithm to different individuals of the same robot type.

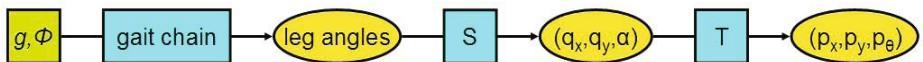


Fig. 7. A decomposition of function F . Given g and ϕ , the gait control chain calculates the leg angles at the moment of the next step. The physical step S maps the leg angles to the footstep q in the reference frame of the robot. The step transform T translates q to the footstep p in the reference frame of the support foot.

7 Experimental Results

Figure 8 presents data from Mocap and the approximated functions for q_x , q_y and α (eqs. 12-14). All three relationships show a strong linear character and can easily be fitted in the most simple manner with linear functions. Naturally,

one would expect the functions of leg angles to step sizes to be sinusoids. The most likely explanation for the seemingly linear coherence is that our robots are taking relatively small steps and sinusoid functions can be well approximated with linear ones, as long as their argument stays close to zero. The function mapping Θ_{Leg}^{yaw} to α is almost identity. The small deviation from identity must originate from the error the physical system makes when at the commanded leg angle. With this deviation accounted for we can expect an improvement in our footstep predictions.

We used the motion capture data to compare the performance of the forward kinematic approach and the machine learning approach. The error of a single prediction is measured by the Euclidean distance between the predicted footstep and the ground truth footprint taken from the motion capture data. We present figures of the mean error and standard deviation measured in three different experiments in Figure 8 lower right. In the first experiment we compared the performance of the forward kinematic model and the machine learning approach on the complete set of approximately 3000 footsteps. The performance of the machine learning approach was measured by 4-fold cross validation. In the second and third experiment we compared the two methods exclusively on the step set produced by only one of the robots, e.g. Conny and Ariane respectively. Additionally, we evaluated transferred models that were trained with 500 steps randomly sampled exclusively from the step set of the other robot.

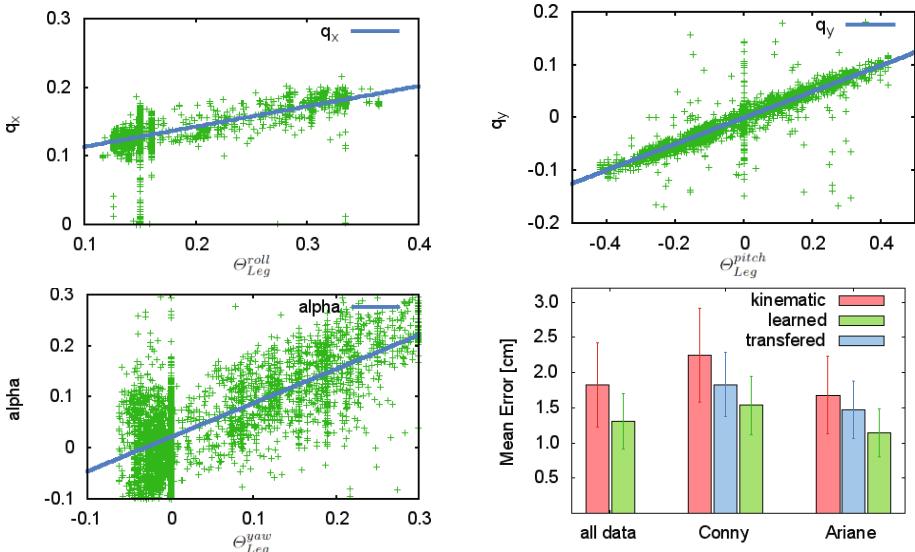


Fig. 8. Plots of the collected data and the approximated functions for q_x (upper left), q_y (upper right), and α (lower left). All three relationships show a linear character. The comparison of the prediction accuracies of the analytic model and the machine learning model on the full step set and the robot specific experiments are shown in the lower right.

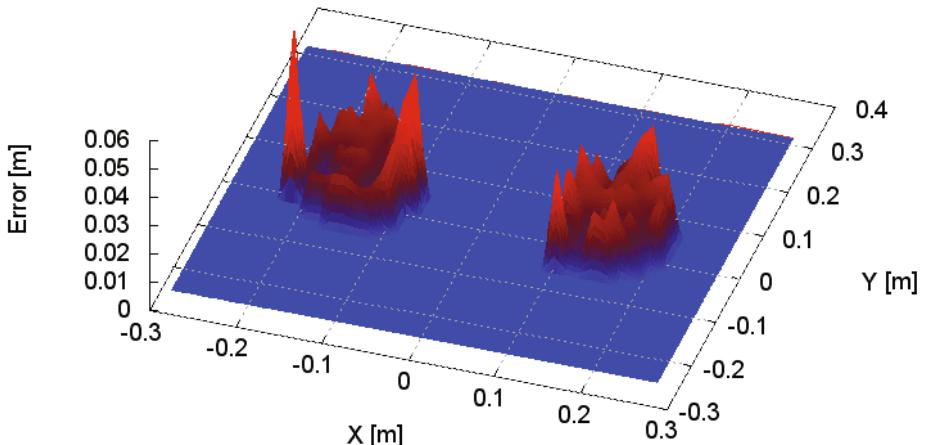


Fig. 9. The working spaces of the left foot and right foot. The height of the graph shows the measured error at each footstep location.

All trained models outperformed the forward kinematic approach. Since the forward kinematic is the same for both robots, the difference in accuracy between the Conny experiment and the Ariane experiment can be explained by a more unstable walking of Conny, which is more difficult to predict due to occasional stumbling. Even though the best results were achieved on Ariane (1.1 cm), we believe this model to be overly adapted to this robot and use the mode trained on the full step set as reference, which achieved an accuracy with a mean error of approximately 1.3 cm.

Figure 9 shows a representation of the working spaces of the feet. Every step was plotted by assuming the support foot location to be at the origin. The height describes the measured error. In the base stance of the robot, even if the gait control vector g is 0, the feet are 12 cm apart. These “zero” steps of the robot are roughly in the respective centers of the two areas. A not unexpected tendency of the prediction error growing with the deviation from the zero step can be observed. The diameters of the working spaces are approximately 20 cm and the average step size was determined to be 14.4 cm. Comparing the mean error of 1.3 cm to the average step size leads to the conclusion, that our predictions are in average 90.97% accurate. Future work will show if this accuracy is enough for a step planning algorithm to hit a small target such as a tennis ball.

Investigating further possibilities to improve the prediction accuracy, we considered discarding the independency assumption and using a non-linear regression method to learn the function

$$f(\Theta_{Leg}^{roll}, \Theta_{Leg}^{pitch}, \Theta_{Leg}^{yaw}) = (q_x, q_y, \alpha) \quad (15)$$

in one sweep. For this attempt to be fruitful, some dependency has to exist between the three parameters and each of the output dimensions. We examined the

Table 1. Correlation coefficients between the leg angles in roll, pitch and yaw directions and the deviation of the footstep q in the reference frame of the trunk

	Δq_x	Δq_y	$\Delta \alpha$
Θ_{Leg}^{roll}	0.022	0.043	0.037
Θ_{Leg}^{pitch}	0.070	0.028	0.020
Θ_{Leg}^{yaw}	0.241	0.013	0.001

correlation between the leg angles produced by the gait and the deviation Δq , which is the difference between the robot centered footstep q and the expected footstep predicted by our linear estimators. Table 1 contains the correlation coefficients. Apart from a small influence of Θ_{Leg}^{yaw} on q_x , no correlation between the parameters and the footstep deviations can be identified. Consequently, our linear approach has already exhausted the learning problem. A more complex approach cannot be expected to produce significantly better results on the same data.

8 Conclusions

We presented a footstep prediction algorithm for a central pattern generated omnidirectional walk. We were able to decompose the footstep prediction into three independent functions that could be approximated with linear regression performed on motion capture data. We achieved a predicton accuracy with a mean error of 1.3 cm. The learned model outperformed the analytic, forward kinematic based solution and showed some transferability to different individuals of the same robot model. We claim that our linear approach cannot be significantly improved with a more complex technique. In future work we plan to use the footstep predictions for a step planning gait control and test the algorithms in real robot soccer games.

Acknowledgments

Funding for the project is provided by Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under grants BE 2556/2-2,/4.

References

1. Behnke, S.: Online Trajectory Generation For Omnidirectional Biped Walking. In: ICRA (2006)
2. Zhao, M., Zhang, J., Dong, H., Liu, Y., Li, L., Su, X.: Humanoid Robot Gait Generation Based on Limit Cycle Stability. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 403–413. Springer, Heidelberg (2009)

3. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K.: Biped walking pattern generation by using preview control of zero-moment point. In: ICRA 2003, pp. 1620–1626 (2003)
4. Huang, Q., Yokoi, K., Kajita, S., Kaneko, K., Arai, H., Koyachi, N., Tanie, K.: Planning Walking Patterns for a Biped Robot. IEEE Transactions on Robotics and Automation 17, 280–289 (2001)
5. Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., Itozumi, T.: Humanoid Robot HRP-2. In: ICRA 2004, pp. 1083–1090 (2004)
6. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., Hirukawa, H.: The 3D linear inverted pendulum mode: a simple modeling for a bipedwalking pattern generation. In: IROS 2001, vol. 1, pp. 239–246.
7. Friedmann, M., Kiener, J., Petters, S., Sakamoto, H., Thomas, D., von Stryk, O.: Versatile, high-quality motions and behavior control of humanoid soccer robots. In: Workshop on Humanoid Soccer Robots of the 2006 Humanoids, pp. 9–16 (2006)
8. Chestnutt, J., Kuffner, J.: A Tiered Planning Strategy for Biped Navigation. In: Humanoids (2004)
9. Kuffner, J., Kagami, S., Nishiwaki, K., Inaba, M., Inoue, H.: Online Footstep Planning for Humanoid Robots. In: ICRA, pp. 932–937 (2003)
10. Chestnutt, J., Lau, M., Cheung, K.M., Kuffner, J., Hodgins, J.K., Kanade, T.: Footstep Planning for the Honda ASIMO Humanoid. In: ICRA (2005)
11. Gutmann, J.S., Fukuchi, M., Fujita, M.: Real-time path planning for humanoid robot navigation. In: Proc. of 19th Int. Conf. on Artificial Intelligence, pp. 1232–1237 (2005)
12. Kanoun, O., Yoshida, E., Laumond, J.P.: An Optimization Formulation for Footsteps Planning. In: Humanoids (2009)
13. Ayaz, Y., Owa, T., Tsujita, T., Konno, A., Munawar, K., Uchiyama, M.: Footstep Planning for Humanoid Robots Among Obstacles of Various Types. In: Humanoids (2009)

Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots

Judith Müller¹, Tim Laue², and Thomas Röfer²

¹ Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany

judy@informatik.uni-bremen.de

² Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de

Abstract. Complex motions like kicking a ball into the goal are becoming more important in RoboCup leagues such as the Standard Platform League. Thus, there is a need for motion sequences that can be parameterized and changed dynamically. This paper presents a motion engine that translates motions into joint angles by using trajectories. These motions are defined as a set of Bezier curves that can be changed online to allow adjusting, for example, a kicking motion precisely to the actual position of the ball. During the execution, motions are stabilized by the combination of center of mass balancing and a gyro feedback-based closed-loop PID controller.

1 Introduction

For successfully playing soccer with humanoid robots, much attention is paid to the development of not only fast but also robust gaits, i.e., to be flexible regarding sudden changes of the walking direction and to be able to compensate for disturbances such as collisions with other robots or the asperity of the ground floor. These two kinds of robustness are also a requirement for the second most important kind of motion in soccer: kicking. During the execution of a kicking motion, the position of the ball might change, either in reality after having been touched by a robot, or virtually in the kicking robot's world model after having perceived the ball at a slightly different place. In these cases, an online adaption of the kicking foot's trajectory becomes necessary to precisely hit the ball. In addition, even slightest disturbances during the kick phase might cause a loss of precision or even prevent a successful kick at all. Thus, balancing mechanisms for a robot standing on one foot only are a crucial factor.

In [5], a key-frame based approach that is one of the most common methods for motion designing is described. This method defines motions as a series of static joint angle sets. Each joint angle set is a key-frame. All joint positions between two key-frames are interpolated. The major disadvantage of this approach is its inflexibility. Once a motion execution has started, the robot cannot react on any new information. Similar approaches are currently used for kicking motions by

many RoboCup teams such as *Cerberus* [1], *Nao Team HTWK* [12], *Kouretes* [15], or *B-Human* [17].

In [7], the static key-frame based approach is extended by the possibility of balancing to compensate external disturbances. In this concept, key-frames are defined as Cartesian limb positions in between which it is interpolated; the actual joint angles are calculated by inverse kinematics. A similar approach is used by the *Nao Team Humboldt* [4], where kicking motions are also defined by Cartesian positions, and in addition, they are stabilized.

The contribution of this paper is the concept of a motion engine that extends the idea of trajectory-based motions of current walking approaches such as [17] and [14] to execute more complex motions such as kicking a ball. In order to stabilize the resulting motions, the engine contains a balancing module that is based on center of mass stabilization of [6] and a sensor feedback closed-loop PID controller based on [9] and [2]. The approach has been applied to a *Nao* humanoid robot as it is used in the RoboCup Standard Platform League.

This paper is organized in four main sections: section 2 introduces the main concept and the procedure of the motion engine. After that, Sect. 3 describes the definition of related motions and the concept of dynamic changes. The components providing stability are described in Sect. 4. Finally, in Sect. 5 the experiments conducted are presented and the results are discussed.

2 Motion Engine Design

The motion engine is developed as a module of the framework introduced in [17] and can be used in parallel to other modules such as a walking engine. The main idea behind the engine is to combine a set of simple motion curves to more complex curves. That way a whole motion is divided into *motion phases* such as lifting the foot or kicking the ball. Simple curves define the motion of a limb of the robot for a certain period of time. They are also referred to as trajectories. A set of simple motion curves or *motion phases* contains six different curves. Each foot and each arm has its own trajectory to control its position. Each foot has an additional trajectory to control the rotational movement.

Since simple trajectories are combined to more complex trajectories, the engine has to make sure that the combination of trajectories of two different phases is smooth to prevent unwanted twitching. The system guarantees smoothness by checking whether the connection point of two curves is continuously differentiable (cf. Sect. 3).

The motion engine receives a list of dynamic points as input from other modules to accomplish dynamic changes. Dynamic points contain information about the desired target position of certain curves and their motion direction (cf. Sect. 3.2). Since these points provide the system with information as, for example, the ball position, they are applied at times during motion execution.

As each leg of the robot contains six different joints, the unknown joint angles are calculated by inverse kinematics. The actual calculation is based on a geometric approach that was described by [11].

Algorithm 1 shows the general structure of the motion engine: first the motion definitions of the requested motion *id* are retrieved. Then, at the start of each phase, the dynamic points are applied to the current phase and the trajectories are initialized. Besides dynamically changing, in this step the smoothness of the curves of the current phase is adapted. Afterwards, the limb positions are retrieved from the current point in time and the trajectories. Since the position of the end effector is known, the joint angles are calculated using inverse kinematics. The final step is to apply the balance corrections.

Algorithm 1. The main procedure of the motion engine

```

1: if wasActive ≠ true then
2:   phase ← 0
3:   currentParameters ← getParameters(id)
4:   addDynValues(phase, currentParameters, dynValues)
5:   initCurrentTrajectories(phase, currentParameters)
6: end if
7: if phase ≤ maxPhase then
8:   time ← getTime(getCurrentTime(), getDeltaT(currentParameters))
9:   if time == 1 then
10:    phase ← phase + 1
11:    addDynValues(phase, currentParameters, dynValues)
12:    initCurrentTrajectories(phase, currentParameters)
13:    time ← 0
14: end if
15: positions ← getLimbPos(currentParameters, phase, time)
16: joints ← calcJoints(positions)
17: addBalance(joints, currentParameters, gyroData)
18: wasActive ← true
19: return joints
20: else
21:   wasActive = false
22:   return stand
23: end if

```

3 Motion Design

In many applications, trajectories are mostly combined equations that are selected by a case differentiation over time. These equations are continuous but not continuously differentiable as the derivative is not defined at the connection points. This means that a real servo motor is supposed to change its speed in no measurable time, which causes a twitch.

A better suiting equation for motion controlling is the Bezier curve. Such a curve is defined through $n+1$ control points. Each of these control points affects the whole curve. Bezier curves can be combined easily and even the connection points between two curves are continuously differentiable under certain conditions. The motion engine presented in this paper uses combined cubic Bezier curves ($n = 3$) as shown in equation 1.

$$b(t) = \sum_{i=0}^3 \binom{3}{i} t^i (1-t)^{3-i} P_i \quad (1)$$

3.1 Combined Trajectories

A motion is a set of phases and a phase is a set of six trajectories. Each trajectory controls a limb, e.g. the left foot. That means that if a motion has two phases, the motion of the left foot is described by two trajectories. Such a combination of two trajectories has to satisfy some conditions for continuous differentiability in the connection point.

The connection point of two Bezier curves is the fourth control point of one curve and the first control point of the next curve. The forth control point of a curve $b_1(t_1)$ is reached if $t_1 = 1$ for $0 \leq t_1 \leq 1$ and the first control point of a curve $b_2(t_2)$ if $t_2 = 0$ for $0 \leq t_2 \leq 1$. For that reason, two Bezier curves $b_1(t_1)$ and $b_2(t_2)$ with the control points P_0, P_1, P_2, P_3 and Q_0, Q_1, Q_2, Q_3 have a connection point if $b_1(1) = b_2(0)$.

If Δt_1 and Δt_2 are equal, i.e., the timings of both phases are equal, continuously differentiability is given if P_2, P_3 , and Q_1 of $b_1(t_1)$ and $b_2(t_2)$ are collinear and equidistant [10]. This means that those three points are on a straight line and the distance between P_2 and P_3 equals the distance of P_3 and Q_1 with $P_3 = Q_0$.

If $P_3 = Q_0$, equation 1 is valid:

$$b_1(1) = b_2(0) \quad (2)$$

The tangents in the points P_3 and Q_0 are calculated using the first derivative of the Bezier curve:

$$\dot{b}_1(1) = 3 \cdot (P_3 - P_2) \quad (3)$$

$$\dot{b}_2(0) = 3 \cdot (Q_1 - Q_0) \quad (4)$$

Equation 3 and 4 applied to 2 provides the condition to guarantee continuous differentiability in the connection point between two Bezier curves with equal timings [10]:

$$P_3 - P_2 = Q_1 - P_3 \quad (5)$$

If the timings of two Bezier curves are not equal, i.e., when Δt_1 and Δt_2 are not equal, then Eq. 5 is insufficient. According to [10], the connection point of two Bezier curves with different timings is continuously differentiable if the points P_2, P_3 and Q_1 are collinear and the ratio of the distance between P_2 and P_3 and the distance of P_3 and Q_1 equals the ratio between Δt_1 and Δt_2 . Equation 6 integrates that condition into equation 5.

$$\frac{P_3 - P_2}{\Delta t_1} = \frac{Q_1 - P_3}{\Delta t_2} \quad (6)$$

Since the phases have an order, the system only has to calculate point Q_1 using Eq. 6 for each phase except for the first one. In the first phase, all control points can be set without conditions as long as the control points in the following phases are calculated correctly.

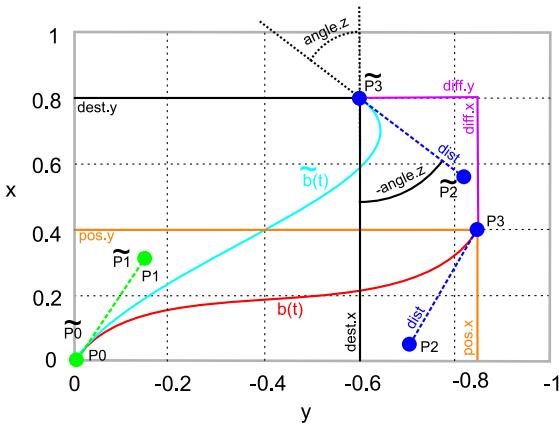


Fig. 1. Applying a dynamic point to curve $b(t)$

3.2 Dynamic Motions

The engine allows changing motions online. Thus, it receives parameters from other components such as the robot's behavior control that can affect a motion specified offline. These parameters are a target position, a motion direction, the phase number, and the limb name. The phase number and the limb name address the curve that should be changed and the target position modifies the fourth control point of that curve. The motion direction rotates the third control point of a curve around the fourth control point to change a forward kick for example into a sideways kick.

The dynamic points are added to the curves at the start of the execution of the phase, not at the start of the execution of the whole motion. This implies that a motion can be changed during its execution, e. g., to follow a ball with the foot.

The actual addition of a dynamic point to a curve $b(t)$ is moving the forth control point P_3 to the desired position and the rotation of the third control point P_2 to the desired angle or motion direction. In Fig. 1, two curves $b(t)$ and $\tilde{b}(t)$ are shown. Curve $b(t)$ is the curve created offline and $\tilde{b}(t)$ is the corresponding curve changed dynamically. When P_3 is moved to the target position, P_2 has to be translated by the same amount to keep the continuous differentiability of the connection point. After the translation of P_3 , P_2 is rotated around P_3 .

4 Stability

Center of mass (COM) based approaches are already in use for the creation of walking gaits, e. g. by [14] who use a trajectory to set the body motion and thus stabilize the gait. The approach presented in [13] introduces a concept that – as *foresighted car driving* – not only considers the actual location of the COM but also the future movements. However, it is not possible to absorb external disturbances using the COM only.

For that reason, a second balancing module was added to the system to compensate for external disturbances. The approach for the secondary module is inspired by [9] and [2]. Disturbances become compensated by the use of a closed loop PID controller that uses gyroscope measurements as input.

4.1 Center of Mass Balancing

Since the robot used is a biped, there are three different stand cases: standing on two feet, standing on the left foot, and standing on the right foot. The desired COM depends on these stand cases. The COM position projected to the ground should always be inside the support polygon to maintain static stability [6]. This support polygon equals the size of the standing foot. In this approach, a sufficient stability is assumed if the COM projected to the ground is at the center of the standing foot.

The calculation of the desired COM projected to the ground is realized by computing the harmonic mean of the n future positions of the stand foot. This lets the desired COM approach the next stand case early, and the COM adaption can take place before the feet actually move.

To realize stabilization, the robot's tilt angle is controlled by a PID controller. The input for this controller is the error between the measured COM projected to the ground and the desired COM. The calculation of the measured COM is made by equation 7 that is obtained from [16].

$$\overrightarrow{com} = \frac{\sum_{i=0}^n \overrightarrow{part_i m_i}}{m_{total}} \quad (7)$$

The manipulated variables obtained by the PID controller using equations 8a and 8b cannot directly be used as tilt angles for the body, because the height of the COM is not included.

$$balance_{y_t} = k_{py} * error_{y_t} + k_i \cdot \int_0^t error_\tau d\tau + k_{dy} \cdot \frac{derror_{y_t}}{dt} \quad (8a)$$

$$balance_{x_t} = k_{px} * error_{x_t} + k_i \cdot \int_0^t error_\tau d\tau + k_{dy} \cdot \frac{derror_{y_t}}{dt} \quad (8b)$$

Equations 9a and 9b include the COM height using the Pythagorean theorem. The angles obtained state the rotation of the foot around the origin of the leg, which is considered by the inverse kinematics.

$$\theta_{x_t} = \text{atan2}(balance_{x_t}, com_{z_t}) \quad (9a)$$

$$\theta_{y_t} = \text{atan2}(balance_{y_t}, com_{z_t}) \quad (9b)$$

4.2 Gyroscope Feedback-Based Balancing

Our approach is based on [9] who use a P-controller with the measured data of a gyroscope as input to compensate for external disturbances. The manipulated variables of the approach are directly used to control the servo motors used for balancing. Since gyroscopes are measuring the angular velocity independently from body tilt and roll, they can only be used to keep an angular velocity.

As extension to the method introduced in [9], our approach uses a PID controller instead of a P controller. The PID controller is distinct from the P controller as it is able to eliminate the entire control deviation and it is consequently more reliable in this context. Furthermore, the approach of [9] suggests that stabilization is given when the rotational velocity of the body is kept by $\frac{0^\circ}{s}$. This is not applicable to our approach as the body has to move during the COM balance stabilizing. It has to be distinguished between desired angular velocity and undesired angular velocity to allow the two balancing components to work together.

The desired angular velocities are taken from the commanded angles of the balancing joints of the standing leg with Eq. 10a and Eq. 10b before the gyro feedback balance is added. The joints responsible to compensate for disturbances are the hip pitch joints and the hip roll joints. It is important to use the target angles without any gyro feedback balancing since to obtain the desired angular velocity, there should not be any velocity included, which is introduced by the disturbance compensation. Otherwise, the desired velocity becomes the velocity that is used to compensate for disturbances in the next frame and the system tries to keep the new velocity that is added to the desired velocity in the next frame and so on.

$$ref_{x_t} = \frac{\theta_{joint_{hipRoll}_t} - \theta_{joint_{hipRoll}_{t-1}}}{\Delta time_t} \quad (10a)$$

$$ref_{y_t} = \frac{\theta_{joint_{hipPitch}_t} - \theta_{joint_{hipPitch}_{t-1}}}{\Delta time_t} \quad (10b)$$

Since the gyroscopes of the robot used are inert, our approach uses exponential smoothing to forecast the next possible angular velocity from previous measurements. Eq. 11 calculates a forecast that is factorized by α . It uses the factorized forecast of the last frame and adds it to the current factorized measurement. This equation is defined recursively. Newly made measurements are weighed more than measurements taken a while ago.

$$\tilde{g}_{x_t} = g_{x_t} \cdot \alpha + \tilde{g}_{x_{t-1}} \cdot (1 - \alpha) \quad (11)$$

The desired angular velocity is also known as the reference velocity that should be kept. Thus, the velocity error is obtained by the subtraction of the desired velocities from the measurements of the gyroscopes. This is done by Eq. 12.

$$error_{x_t} = \tilde{g}_{x_t} - ref_{x_t} \quad (12)$$

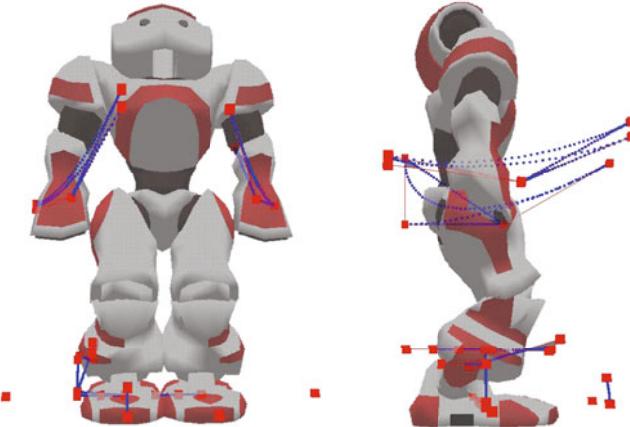


Fig. 2. Visualization of a basic kick motion

The velocity error serves as input for the PID controller in Eq. 13a and Eq. 13b and the manipulated variables serve as offset angles – that become added directly – for the balancing joints (hip pitch and hip roll).

$$\theta_{offset_{hipRoll_t}} = K_p \cdot error_{x_t} + K_i \cdot \int_0^t error_{x_\tau} d\tau + K_d \cdot \frac{derror_{x_t}}{dt} \quad (13a)$$

$$\theta_{offset_{hipPitch_t}} = K_p \cdot error_{y_t} + K_i \cdot \int_0^t error_{y_\tau} d\tau + K_d \cdot \frac{derror_{y_t}}{dt} \quad (13b)$$

5 Experiments

Figure 2 visualizes a kick motion created for the motion engine introduced in this paper. The motion is divided into seven phases: shift the masses, lift the foot, strike out, kick the ball, take the foot back, lower the foot, and shift the masses back.

The phases *strike out* and *kick the ball* of this motion can be changed online. In Fig. 3 the motion is changed dynamically to hit the ball at the center to get a straight kick. The dynamic parameters for that motion depend on the ball position and the target angle. In Fig. 3 top the desired target angle was 0° . In Fig. 3 bottom, the same motion that was used for the straight kick was changed online at the phases *strike out* and *kick the ball* to kick the ball from the side. The target angle was 90° .

5.1 Angular Deviation Experiment

An experiment with 25 trials of kicking a ball to a target angle of 0° was made to evaluate how effective the straight kick can be. In each trial, the robot was

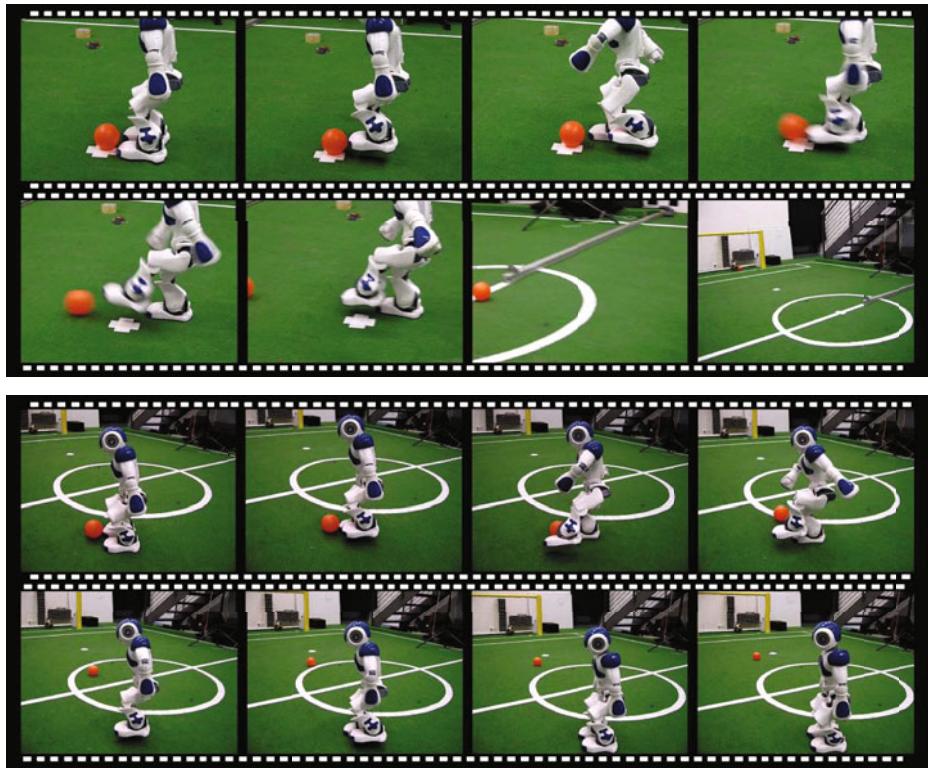


Fig. 3. The dynamic forward kick (top) and the dynamic side kick (bottom) are sharing the same basic motion

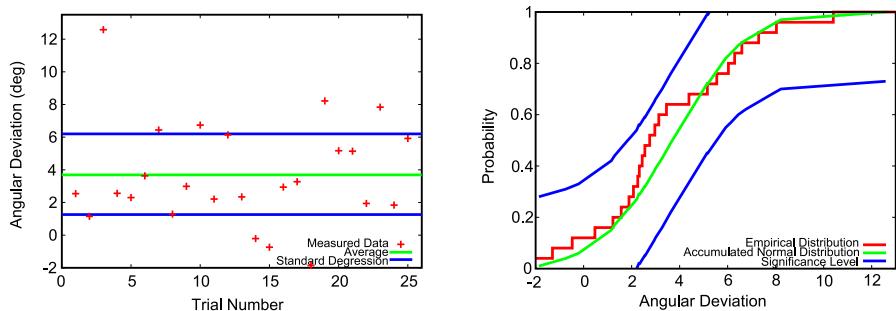


Fig. 4. The measured angular deviations with average (3.32°) and standard deviation (2.11°) (left); Check whether current distribution is normally distributed (right)

set about 30 cm away from the ball but with the face pointing to it. The robot had to go to the ball on its own for reaching a random kick position without human interaction. The ball used was a standard orange tennis ball.

The results of the angular deviations of each of the 25 trials are presented in Fig. 4 left. The overall average (in green) is 3.32° and the standard deviation (in blue) is 2.11° . Furthermore, 16 of 25 samples are within the standard deviation. Since a normal distribution is very reasonable if 68% of the samples are within the standard deviation, a Kolmogorow-Smirnow test for normal distribution was made. The test was made with a significance level of 0.05 and the result is presented in Fig. 4 right. The test is positive, when none of the samples are outside of the tolerance. Since the test proves that the experiment result is normally distributed, it is very likely that there were no unusual disturbances. Furthermore, the test implies that 68% of all kicked balls have an angular deviation within the range of 1.21° to 5.43° . Since normally distributed samples have the characteristic to differ to 95% at most 2σ from the average, the kicked balls have an angular deviation within the range of -0.9° to 7.54° by a chance of 95%.

Besides the angular deviation experiment the maximum distance the kicked balls reached was also measured. The overall average of 25 samples is 5435.65 mm and the standard deviation is 251.03 mm. Since the samples of this test are normally distributed, the kicked balls reach a maximum distance within the range of 4933.59 mm and 5937.65 mm by a chance of 95%.

5.2 Pendulum Experiment

Besides tests about the effectiveness of motions, a pendulum experiment was conducted to evaluate the stability. In this experiment, a pendulum with the weight of 500g was attached to a rope. The rope with the length of 45 cm was attached to a rack with a height of 80 cm. The pendulum is held in a 90° angle relative to the robot. After releasing the pendulum, it hits the robot at a height of 35 cm from the ground. This experiment includes ten trials with balancing

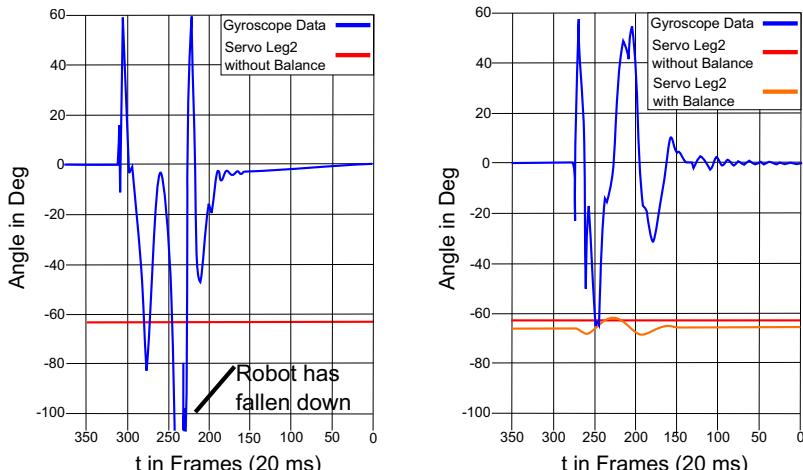


Fig. 5. The plotted data without balance (left) and with balance (right)

and ten trials without. The robot is standing on one leg and the pendulum hits it from the back.

Figure 5 left shows a plot of the test without balancing. The blue curve is the measured error retrieved from the gyroscope and the orange line is the movement of the balancing joint. In this case, the joint is not moving because the balance module is turned off. The reference angular velocity from this test is $\frac{0^\circ}{s}$. Furthermore, the figure shows that the robot could not compensate the disturbance and fell down after 75 frames. In each of the ten trials, the robot fell down.

Figure 5 right shows a plot of the trials with balancing. The robot was able to compensate for the disturbance after 125 frames. Since one frame lasts 20 ms, the whole compensation was completed after only 2.5 seconds. The robot did not fall down in all ten trials, and it was always able to compensate for the disturbance in a time similar to 2.5 seconds.

6 Conclusions and Future Work

The experiments imply that it is possible to create very effective motions with the motion engine introduced. The motions are stable despite the dynamical changing of the kick direction. Since the stabilization is divided into two approaches, it would be an enhancement to replace these by a single method such as the preview control introduced in [8]. The preview control takes the system dynamics into account and uses the principle of the zero moment point to stabilize the motions.

Since Bezier curves are well known, motions can be created very easily even by less experienced users. Nevertheless, an alternative approach would be to record motions by kinesthetic manipulation as in [3].

References

1. Akin, H.L., Mericli, T., Özkucur, K.C., Gökce, B.: Cerberus 2010 team description paper (2009), <http://www.tzi.de/spl/pub/Website/Teams2009/Cerberus10TDP.pdf> (as of April 26, 2010)
2. Bartsch, S.: Steuerung der Fortbewegung eines humanoiden Roboters, robust gegen externe Störungen und geeignet für unebenes Terrain, basierend auf biologisch inspirierter Architektur. Diplomarbeit, Universität Bremen (January 2007)
3. Berger, E., Amor, H.B., Vogt, D., Jung, B.: Towards a simulator for imitation learning with kinesthetic bootstrapping. In: Menegatti, E. (ed.) Workshop Proceedings of Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR), pp. 167–173 (November 2008)
4. Borisov, A., Ferdowsizadeh, A., Mohr, C., Mellmann, H., Martius, M., Krause, T., Hermann, T., Welter, O., Xu, Y.: NAO-Team Humboldt 2009 (2009), <http://www.naoteamhumboldt.de/papers/NaoTH09Report.pdf> (as of April 26, 2010)
5. Brunn, R., Düffert, U., Jüngel, M., Laue, T., Lötzsch, M., Petters, S., Risler, M., Röfer, T., Spiess, K., Sztybryc, A.: GermanTeam 2001. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 705–708. Springer, Heidelberg (2002)

6. Bräunl, T.: *Embedded Robotics - Mobile Robot Design and Applications with Embedded Systems*. Springer, Heidelberg (2003)
7. Czarnetzki, S., Kerner, S., Klagges, D.: Combining key frame based motion design with controlled movement execution (2010)
8. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems* 57(8), 839–845 (2009)
9. Faber, F., Behnke, S.: Stochastic optimization of bipedal walking using gyro feedback and phase resetting. In: *Proceedings of the International Conference on Humanoid Robots (Humanoids)* (2007)
10. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: *Computer Graphics - Principles and Practice*, 2nd edn. Addison-Wesley Publishing Company, Reading (1990), xxIII, 1174 S : Ill
11. Graf, C., Härtl, A., Röfer, T., Laue, T.: A robust closed-loop gait for the standard platform league humanoid. In: Zhou, C., Pagello, E., Menegatti, E., Behnke, S., Röfer, T. (eds.) *Proceedings of the Fourth Workshop on Humanoid Soccer Robots in Conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots*, Paris, France, pp. 30–37 (2009)
12. Jahn, P.D.K.U., Borkmann, D., Reinhardt, T., Tilgner, R., Rexin, N., Seering, S.: Nao Team HTWK Leipzig team research report 2009 (2009), <http://naoteam.imn.htwk-leipzig.de/documents/techReportHTWK.pdf> (as of April 26, 2010)
13. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Jirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: *Proceedings of the 2003 IEEE, International Conference on Robotics and Automation* (2003)
14. Niehaus, C., Röfer, T., Laue, T.: Gait optimization on a humanoid robot using particle swarm optimization. In: Pagello, E., Zhou, C., Menegatti, E., Behnke, S. (eds.) *Proceedings of the Second Workshop on Humanoid Soccer Robots in Conjunction with the 2007 IEEE-RAS International Conference on Humanoid Robots*, Pittsburgh, PA, USA (2007)
15. Panakos, A., Paraschos, A., Pierris, G., Chroni, D., Vafeias, E., Chatzilaris, E., Vazaios, E., Lagoudakis, M.G., Vlassis, N.: Kouretes 2008 - Nao team report (2009), <http://www.intelligence.tuc.gr/kouretes/docs/2008-kouretes-nao-report.pdf> (as of April 26, 2010)
16. Pratab, R., Ruina, A.: *Introduction to Statics and Dynamics*. Oxford University Press, Oxford (2009), <http://ruina.tam.cornell.edu/Book/> (Preprint)
17. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-Human team report and code release 2009 (2009), http://www.b-human.de/download.php?file=coderelease09_doc

Towards Semantic Scene Analysis with Time-of-Flight Cameras

Dirk Holz¹, Ruwen Schnabel², David Droeßel¹,
Jörg Stueckler¹, and Sven Behnke¹

¹ University of Bonn, Institute of Computer Science VI

² University of Bonn, Institute of Computer Science II

{holz,droeszel,stueckler}@ais.uni-bonn.de,
{schnabel,behnke}@cs.uni-bonn.de

Abstract. For planning grasps and other object manipulation actions in complex environments, 3D semantic information becomes crucial. This paper focuses on the application of recent 3D Time-of-Flight (ToF) cameras in the context of semantic scene analysis. For being able to acquire semantic information from ToF camera data, we a) pre-process the data including outlier removal, filtering and phase unwrapping for correcting erroneous distance measurements, and b) apply a randomized algorithm for detecting shapes such as planes, spheres, and cylinders. We present experimental results that show that the robustness against noise and outliers of the underlying RANSAC paradigm allows for segmenting and classifying objects in 3D ToF camera data captured in natural mobile manipulation setups.

1 Introduction

Autonomous mobile robots need environment models in order to plan actions and navigate effectively. Two-dimensional metric maps, built from 2D laser range scans, became the de-facto standard to tackle navigation problems such as path planning and localization of the robot platform. For planning arm motions and grasps, however, 3D semantic information becomes crucial since:

1. Objects need to be detected in the presence of other objects (e.g., on a cluttered table).
2. The robot needs to determine whether or not an object is graspable (e.g., with respect to its size).
3. The robot needs to determine the 3D pose of the object as a goal for its end-effector.
4. The robot needs to determine the 3D pose (and boundaries) of neighboring objects in order to plan an obstacle-free path.

The context of the work presented here is the RoboCup@Home league. This league addresses service robot applications and focuses on navigation (and SLAM) in dynamic environments, mobile manipulation and human-robot-interaction. A typical task for a mobile service robot is to fetch and deliver objects such as beverages.

This involves detecting and recognizing objects as well as planning arm motions and grasps.

In previous work, we used a 2D laser scanner for detecting possible object locations on tables. The 2D positions of the object candidates were then projected into the image plane of a color camera for feature-based object recognition. In case of a successful recognition (i.e., the object to deliver was among the candidates), grasping of the object was initiated. The drawback of this approach (as illustrated in Figure 1) is that objects can occlude each other in the two-dimensional measurement plane of the laser range scanner.

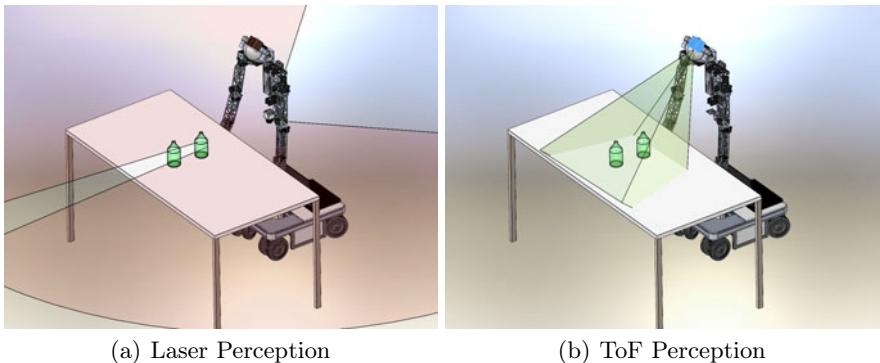


Fig. 1. Perceiving objects on the table. With the trunk laser scanner (a), the second object is occluded and not perceived. Using the ToF camera (b), mounted in the robot's head, would allow for perceiving both objects.

In this work, we present our efforts in using Time-of-Flight (ToF) cameras for perceiving semantic information in the robot's workspace and, in particular, detect tables and objects as well as their shapes.

One of the first applications in robotics considering ToF cameras as an alternative to laser scanning has been presented in 2004 by Weingarten, Grüner, and Siegwart who evaluated a SwissRanger SR-2 camera in terms of basic obstacle avoidance and local path-planning capabilities [14]. In 2005, Sheh et al. used a ToF camera for human-assisted 3D mapping in the context of the RoboCup Rescue league [12]. Ohno et al. used a SwissRanger SR-2 camera for estimating a robot's trajectory and reconstructing the surface of the environment in 2006 [9]. Recently, May et al. presented and evaluated different approaches for registering multiple range images of a SwissRanger SR-3000 camera in the context of fully autonomous 3D mapping [6]. All the aforementioned approaches have shown that ToF cameras require to take care of their complex error model (see [6] for an overview).

The extraction of semantic information from 3D laser scan data has seen a lot of progress in the last decade, of which we want to mention two approaches. Nüchter et al. extract environmental structures such as walls, ceilings and drivable surfaces from 3D laser range scans and use trained classifiers to detect

objects, like for instance humans and other robots [8]. Rusu et al. extract hybrid representations of objects consisting of detected shapes, as will be done here, as well as surface reconstructions where no shapes have been detected [10]. Both approaches show good results when processing accurate 3D laser range data.

The remainder of this paper is organized as follows (referring to Figure 2): Section 2 covers pre-processing steps being necessary to cope with the complex error model of ToF cameras. Section 3 deals with the detection of table tops, the estimation of the corresponding planar models and the segmentation of individual objects. Section 4 finally covers the detection of primitive shapes in 3D point clouds and the classification of the segmented objects.

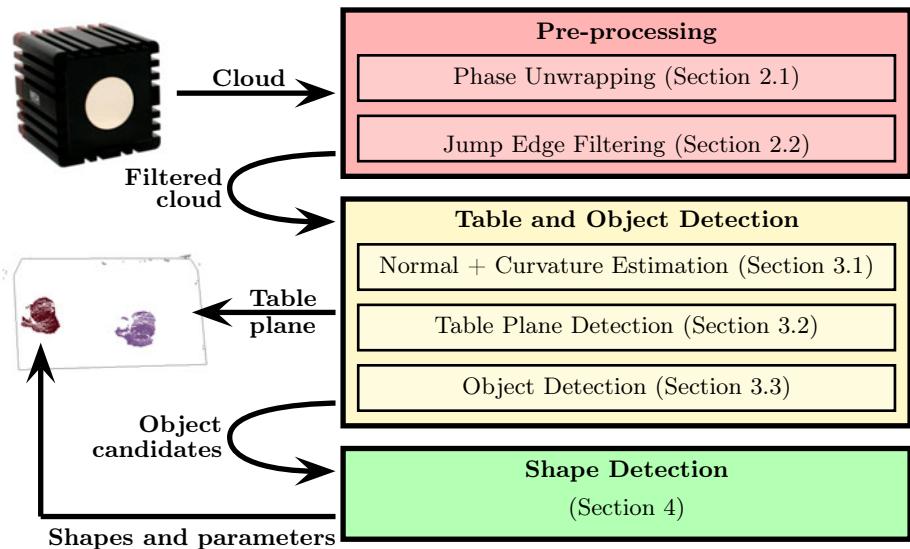


Fig. 2. System overview

2 Pre-processing 3D Point Clouds

Besides a large variety of systematic and non-systematic errors (see [6]), ToF cameras show two problems that are characteristic for their measurement principle. The first problem is the ambiguity of distance measurements. The second problem is that the acquired point clouds contain phantom measurements occurring at distance discontinuities, i.e., at the boundaries of surfaces partially occluding each other. Both problems cause spurious measurements that do not correspond to any object in the real physical environment. By means of phase unwrapping and jump edge filtering both problems are addressed when pre-processing the acquired point clouds.

2.1 Probabilistic Phase Unwrapping

ToF cameras illuminate the environment by means of an array of LEDs that emit amplitude-modulated near-infrared light. The reflected light is received by a CCD/CMOS chip. Depth information is gained for all pixels in parallel by measuring the phase shift between the emitted and the reflected light. This phase shift is proportional to the object's distance to the sensor modulo the wavelength of the modulation frequency. This characteristic results in a distance ambiguity. That is, objects farther away from the sensor than the maximum measurable distance d_{\max} are, respectively, *wrapped* and projected into the interval $[0, d_{\max}]$.

A common way to handle these distance ambiguities is to neglect measurements based on the ratio of measured distance and intensity. The amplitude of the reflected signal decreases quadratically with the measured distance. Sorting out points not following this scheme, e.g., points with a low intensity at a short distance, removes the majority of wrapped measurements but also valid measurements on less reflective surfaces.

In contrast to these approaches, we correct the wrapped measurements instead of neglecting them. We apply phase unwrapping techniques to reconstruct depth measurements behind the sensor's non-ambiguity range. The goal of phase unwrapping is to infer a number of phase jumps from the wrapped signal. Under the assumption that neighboring measurements are more likely close to each other than farther apart, relative phase jumps between neighboring pixels can be extracted. The signal can be unwrapped by integrating these phase jumps into the wrapped signal. We use a probabilistic approach based on [3] that relies on discontinuities in the image to infer these phase jumps. In addition to depth discontinuities, we incorporate the intensity of the reflected signal, since it depends on the object's distance and can indicate inconsistencies between a measured and the corresponding real distance.

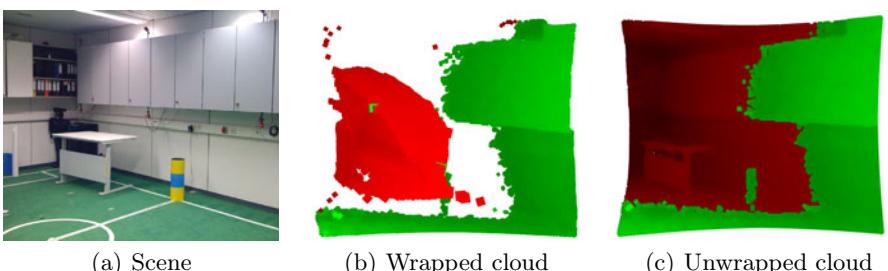


Fig. 3. Phase unwrapping. By correcting the depth image at detected phase jumps, we can obtain valid measurements from objects being farther away from the sensor than the maximum measurable distance d_{\max} . Here the red measurements need to be corrected, the green points naturally lie in the interval $[0, d_{\max}]$.

2.2 Jump Edge Filtering

Jump edges are known to cause spurious measurements that should either be corrected or neglected when processing ToF depth information. For simply neglecting these measurements, sufficient results are achieved by examining local neighborhood relations. From a set of 3D points $P = \{\mathbf{p}_i \in \mathbb{R}^3 | i = 1, \dots, N_p\}$, jump edges J can be determined by comparing the opposing angles $\theta_{i,n}$ of the triangle spanned by the focal point $\mathbf{f} = \mathbf{0}$, point \mathbf{p}_i and its eight neighbors $P_n = \{\mathbf{p}_{i,n} | i = 1, \dots, N_p : n = 1, \dots, 8\}$ with a threshold θ_{th} :

$$\theta_i = \max \arcsin \left(\frac{\|\mathbf{p}_{i,n}\|}{\|\mathbf{p}_{i,n} - \mathbf{p}_i\|} \sin \varphi \right), \quad (1)$$

$$J = \{\mathbf{p}_i | \theta_i > \theta_{th}\}, \quad (2)$$

where φ is the apex angle between two neighboring pixels. That is, neighboring points that lie on a common line-of-sight to the focal point \mathbf{f} are, respectively, removed from the point cloud and marked as being invalid. A typical result of applying this filter is shown in Figure 4.

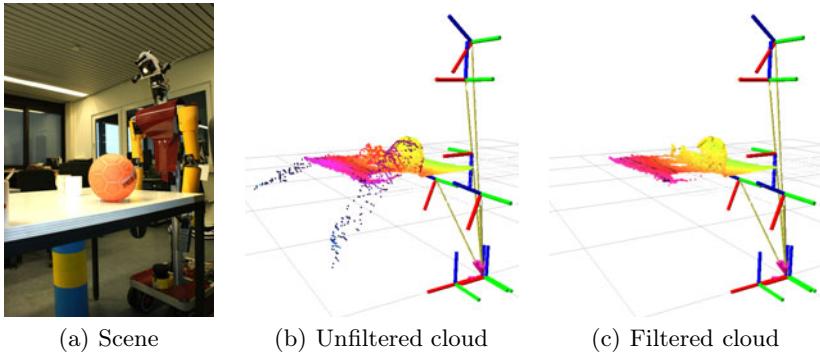


Fig. 4. Sorting out jump edges. Shown are a photo of an example scene (a), the captured unfiltered point cloud (b) and the filtered cloud (c). It can be seen that the majority of erroneous measurements caused by jump edges, e.g., between table and floor in (b), are sorted out in the filtered cloud (c).

3 Table and Object Detection

The detection of tables and objects in the filtered point clouds is conducted in three steps: We first compute local surface normals and variations for all points. This information is then used to detect larger horizontal planes and fitting corresponding planar models into the data. Points above these planes but inside their boundaries are then clustered in order to form the object candidates for later shape detection and feature-based recognition.

3.1 Computing Local Surface Normals and Curvature Changes

A common way for determining the normal to a point p_i on a surface is to approximate the problem by fitting a plane to the point's local neighborhood \mathcal{P}_i in a least squares error sense. This neighborhood is formed either by the k nearest neighbors or by all points within a radius r from p_i .

Searching for nearest neighbors is computationally expensive. Even specialized algorithms like approximate search in kd -trees [7] can cause longer runtimes when building the search structure for a larger point set. Instead of really searching for nearest neighbors, we approximate the problem and exploit the order of the measurements in the point cloud (176×144 distance measurements). We build a lookup table storing, for every point index, the ring-neighborhood being formed by the k closest indices in index space. That is, starting from p_i we circle around the image index ($x = i/176, y = i \bmod 176$) in anti-clockwise order and store the point index for the traversed pixels in the lookup table. When processing a new point cloud we only update the squared distances from every point p_i to its k neighbors as provided by the lookup table.

The approximated nearest neighbors do not resemble the true nearest neighbors in the vicinity of transitions between different surfaces or partial occlusions, and if the point cloud is highly affected by noise and erroneous measurements. To take this into account, we check the computed squared distances and mark those being larger than some threshold r^2 as being invalid. That is, our local neighborhood \mathcal{P}_i is bounded by both a maximum number of neighbors k and a maximum distance r .

Given the local neighborhood \mathcal{P}_i , the local surface normal \mathbf{n}_i can be estimated by analyzing the eigenvectors of the covariance matrix $C_i \in \mathbb{R}^{3 \times 3}$ of \mathcal{P}_i . An estimate of \mathbf{n}_i can be obtained from the eigenvector $\mathbf{v}_{i,0}$ corresponding to the smallest eigenvalue $\lambda_{i,0}$. The ratio between the smallest eigenvalue and the sum of eigenvalues provides an estimate of the local curvature Δc_i .

The aforementioned neighborhood approximation drastically decreases the computational complexity of the surface normal and curvature estimation. However, more important is that the involved inaccuracies did not considerably

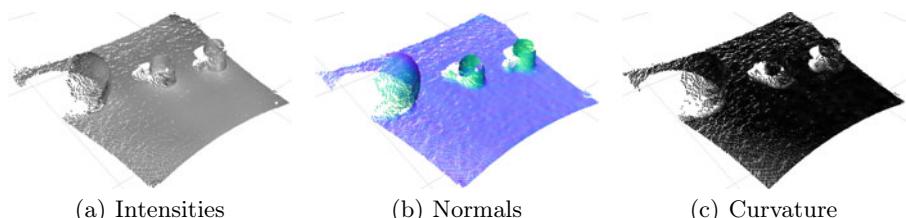


Fig. 5. Computing local surface normals and curvature changes. Shown are the input point cloud with intensity information (a) as well as the computed surface normals (b) and local curvature changes (c). The used parameters are $k = 40$ and $r = 20$ cm.

degraded the results in our experiments. Figure 5 shows a typical example of local surface normals and curvature changes as computed for a pre-processed point cloud.

3.2 Detecting the Table Plane

For detecting tables in the vicinity of the robot, we extract those points \mathbf{p}_i from the point cloud that satisfy the following constraints: 1.) the surface normal \mathbf{n}_i is nearly parallel to the z -axis (i.e., $\mathbf{n}_i \parallel \hat{\mathbf{Z}}$) and 2.) the surface around \mathbf{p}_i is smooth (i.e., $\Delta c \approx 0$). Points satisfying these constraints have likely been measured on the surface of a table and form an initial set of table points T . In order to distinguish multiple tables, we examine the distribution in the measured heights $\mathbf{p}_i^z, \mathbf{p}_i \in T$ and split T into multiple sets T_1, \dots, T_n in case of larger fluctuations. The same is done for larger variations in the position $(\mathbf{p}_i^x, \mathbf{p}_i^y)^T$ of the points.

In order to obtain an efficient representation of tables and to segment individual objects, we fit a planar model into each table point set T_i using the *M-Estimator Sample Consensus* (MSAC) framework – an extension to the well-known RANSAC paradigm where inliers receive a certain score depending on how well they fit the data [13]. This M-Estimator is particularly robust against noise and outliers. For the point cloud from Figure 5, all points in T belong to the same table. The result of fitting a planar model to T is shown in Figure 6.a. The planar model that best fits the data is almost parallel to the xy -plane and is supported by 20 731 inliers. It can already be seen that, despite some points on the table’s boundaries, the outliers correspond to the objects on the table.

Once the planar model has been found, we project all inliers onto the detected plane and compute the 2D convex hull by means of Graham’s Scan Algorithm [4]. The convex hull for the 20 731 points from Figure 6.a is shown in Figure 6.b. It consists of 9 points and accurately represents the table top.

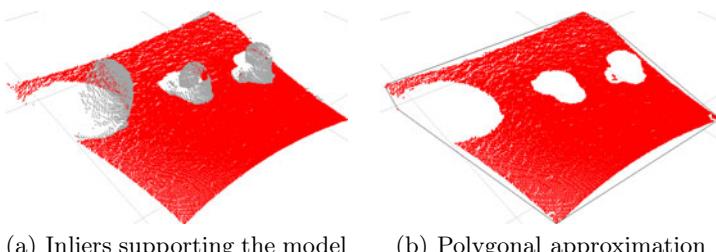


Fig. 6. Detecting the table plane. Shown are the inliers (red) supporting the planar model (a) as well as the 3D polygonal approximation (b) formed by the two-dimensional convex hull of the inliers (projected onto the table plane).

3.3 Clustering Outliers and Detecting Objects

All outliers from fitting the planar model as well as the points that have not been considered for the table point set T are potential object points. That is, they could have been measured on the surface of an object. Since we are only interested in objects on top of the table, we first sort out all points lying below the table plane as well as those points that do not lie within the bounding polygon. In order to obtain point sets that represent a common object, we apply a simple clustering based on the Euclidean distance between neighboring points. Neighboring points whose point-to-point distance is below a threshold d_{\max} are recursively merged into clusters. Clusters whose cardinality exceed a minimum number n_{\min} of support points are considered as object candidates.

The resulting segmentation of the ongoing examples from Figure 5 and Figure 6 is shown in Figure 7. In order to use the segmented object clusters for motion planning, we compute the centroid as well as the oriented bounding box for all points in each cluster. For planning grasps, however, we need to determine the shape of the objects.

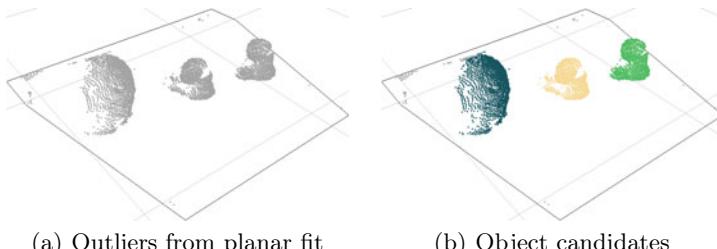


Fig. 7. Detecting object candidates. Shown are the unclustered outliers (a) and the object candidates (b) obtained from Euclidean clustering. Object candidates are colored. The remaining points are gray. Here, the parameters are $d_{\max} = 2.5$ cm and $n_{\min} = 250$.

4 Randomized Shape Detection

In order to robustly detect different kinds of geometric primitives, we employ an efficient RANSAC algorithm that directly operates on the point clouds and the associated surface normal information. Indeed, in our setting we can closely follow a simplified version of the approach proposed by Schnabel et al. [11]. While the original method focuses on achieving efficiency even on huge point clouds, the point clouds in the considered application are comparatively small and thus not all optimizations worthwhile.

Given a point cloud $P = \{\mathbf{p}_i \in \mathbb{R}^3 | i = 1, \dots, N_p\}$ with associated normals $\{\mathbf{n}_i, \dots, \mathbf{n}_{N_p}\}$, the output of the algorithm is a set of primitive shapes $\Psi = \{\psi_1, \dots, \psi_n\}$ with corresponding disjoint sets of points $P_\psi = \{P_{\psi_1} \subset P, \dots, P_{\psi_n} \subset P\}$ and a set of remaining points $R = P \setminus \bigcup_\psi P_\psi$.

The shape extraction problem is framed as an optimization problem defined by a score function σ_P . In each iteration of the algorithm, the primitive with maximal score is searched using the RANSAC paradigm. New shape candidates are generated by randomly sampling minimal subsets of P . Candidates of *all* considered shape types are generated for *every* minimal set and all candidates are collected in the set C . Thus, no special ordering has to be imposed on the detection of different types of shapes. After new candidates have been generated, the candidate m with the highest score is computed employing the efficient lazy score evaluation scheme presented in Sec. 4.3. The best candidate is only accepted if, given the number of inliers $|m|$ of the candidate and the number of drawn candidates $|C|$, the probability that no better candidate was overlooked during sampling is high enough (see [2]). If a candidate is accepted, the corresponding points P_m are removed from P and the candidates C_m generated with points in P_m are deleted from C . The algorithm terminates as soon as the probability of detection for a shape with a user defined minimal size τ is large enough.

4.1 Shape Estimation

The shapes we consider in this work are planes, spheres, cylinders, cones and tori which have between three and seven parameters. Every 3D-point p_i fixes only one parameter of the shape. In order to reduce the number of points in a minimal set, we also use the unoriented approximate surface normal n_i for each point, so that the direction gives us two more parameters per sample. That way it is possible to estimate each of the considered basic shapes from at most three point samples. However, always using one additional sample is advantageous because the surplus parameters can be used to immediately verify a candidate and thus eliminate the need of evaluating many relatively low scored shapes [5].

4.2 Score

The score function σ_P is responsible for measuring the quality of a given shape candidate. We use the following aspects in our scoring function: 1.) To measure the support of a candidate, we use the number of points that fall within an ϵ -band around the shape. 2.) To ensure that the points inside the band roughly follow the curvature pattern of the given primitive, we only count those points inside the band whose normals do not deviate from the normal of the shape more than a given angle α . 3.) Additionally we incorporate a connectivity measure: Among the points that fulfill the previous two conditions, only those are considered that constitute the largest connected component on the shape.

4.3 Score Evaluation

Obviously the cost of evaluation would be prohibitive without any optimizations because in a naïve implementation, the distance to all points in P would have to

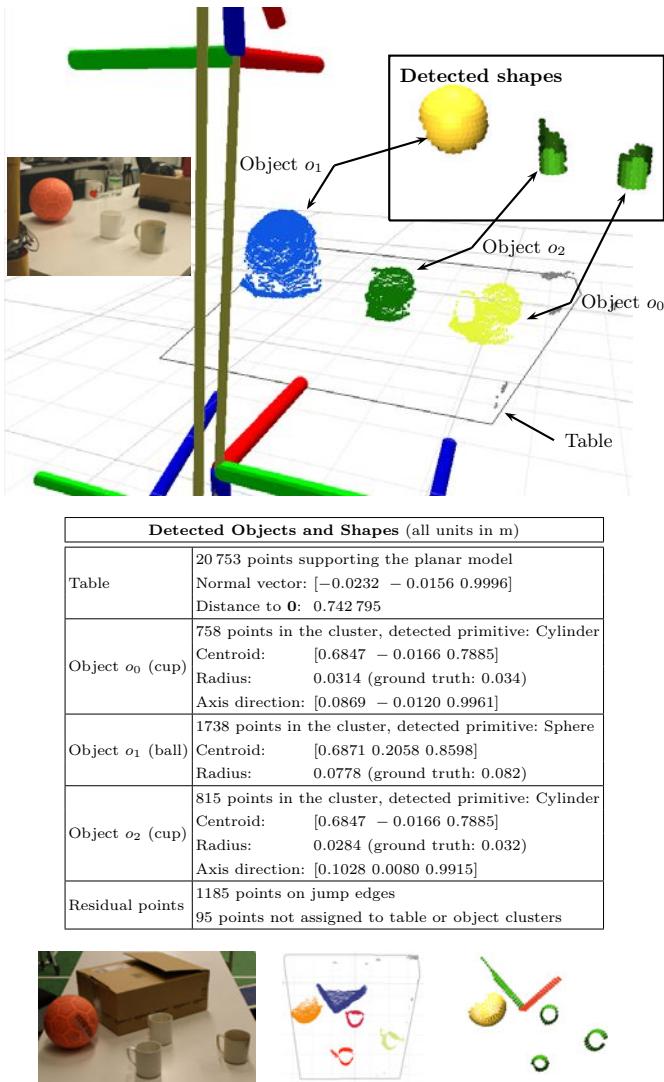


Fig. 8. Typical result of table, object, and shape detection. TOP: After fitting the planar model, the 20 753 inliers are removed and the table is represented solely by the model parameters and the convex hull of the inliers. The remaining 4591 points are segmented into 3 clusters (random colors) and successfully classified as being a sphere and two cylinders (see table below). The summarized results incl. ground truth information can be found in the table below. BOTTOM: Example of a more complex scene with one false detection (the left side of the box is detected as a cylinder due to the highly inaccurate data in this region). Colors are red (plane), green (cylinder) and yellow (sphere).

be computed together with a normal at a corresponding position on the shape for each candidate. But since in each run we are only interested in the candidate that achieves the highest score, using the entire point cloud P when computing $\sigma_P(\psi)$ is not necessary for every shape candidate.

We significantly reduce the number of points that have to be considered in the evaluation of $\sigma_P(\psi)$ by splitting the point cloud P into a set of disjoint random subsets: $P = S_1 \cup \dots \cup S_r$.

After a shape candidate was generated and successfully verified, the candidate is only scored against the first subset S_1 and no connected component is extracted yet. From the score $\sigma_S(\psi)$ on a subset $S \subset P$ an estimate $\hat{\sigma}_P(\psi)$ for the score $\sigma_P(\psi)$ on all points can be extrapolated using the well known induction from inferential statistics:

$$\hat{\sigma}_P(\psi, S) = -1 - f(-2 - |S|, -2 - |P|, -1 - |S_\psi|), \quad (3)$$

$$\text{where } f(N, x, n) = \frac{xn \pm \sqrt{\frac{xn(N-x)(N-n)}{N-1}}}{N} \quad (4)$$

is the mean plus/minus the standard deviation of the hypergeometric distribution. $\hat{\sigma}_P(\psi)$ is a confidence interval $[l_\psi, u_\psi]$ that describes a range of likely values for the true score $\sigma_P(\psi)$. The expected value $E(\sigma_P(\psi))$ is given by $\frac{l_\psi + u_\psi}{2}$. With this extrapolation the potentially best candidate ψ_m can be quickly identified by choosing the one with the highest expected value. Since the uncertainty of the estimation is captured in the confidence intervals, the truly maximal candidate can be found by comparing the confidence intervals of the candidates.

If the confidence intervals of ψ_m and another candidate ψ_i overlap, the score on an additional subset is evaluated for both candidates and new extrapolations are computed, now taking into account the scores on all subsets that have already been computed. The more subsets have been considered, the smaller becomes the range of the confidence intervals, since the uncertainty in the estimation decreases. Further subsets are included until the confidence intervals of ψ_i and ψ_m no longer overlap and it can be decided if either ψ_i or ψ_m is better.

The advantage of this priority-based candidate evaluation is that it is less dependent on the random order in which candidates are generated. Compared to the related but order-dependent approach of David Capel [1], we achieve a 20% speedup on average on a single core machine.

Results of applying the shape detection algorithm to the object clusters from Section 3 are shown in Figure 8.

5 Conclusion

We have presented a simple, yet efficient and robust, tool chain for extracting semantic information from ToF camera data. It comprises techniques for correcting erroneous measurements caused by the ambiguity in distance measurements as well as for filtering points on jump edges. By means of a MSAC-based approach and the information about the surface in a point's local neighborhood,

we are able to detect table tops and objects thereon. Furthermore, we presented an approach for detecting primitive shapes in the detected objects that allow, e.g., for planning grasping motions.

In its current state, the presented approach can process complete point clouds of 25 344 points (i.e., without working on a sub-sampled copy of the cloud) with 2.5 Hz to 5 Hz. In future, we plan to further speed up the processing tool chain for being able to track objects with much higher frame rates. Furthermore, it is planned to use the extracted semantic information for registration and to construct 3D semantic maps of the robot's workspace.

References

1. Capel, D.P.: An effective bail-out test for RANSAC consensus scoring. In: Proc. British Machine Vision Conf., pp. 629–638 (2005)
2. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24(6), 381–395 (1981)
3. Frey, B.J., Koetter, R., Petrovic, N.: Very Loopy Belief Propagation for Unwrapping Phase Images. In: Neural Information Processing Systems Conference (NIPS), Algorithms & Architectures (2001)
4. Graham, R.L.: An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. *Information Processing Letters* 1(4), 132–133 (1972)
5. Matas, J., Chum, O.: Randomized RANSAC with $t(d, d)$ test. In: Proc. of the British Machine Vision Conference 2002, BMVC (2002)
6. May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., Hertzberg, J.: Three-dimensional mapping with time-of-flight cameras. *Journal of Field Robotics, Special Issue on Three-Dimensional Mapping, Part 2* 26(11-12), 934–965 (2009)
7. Mount, D., Arya, S.: ANN: A library for approximate nearest neighbor searching. In: Proc. of the 2nd Annual Fall Workshop on Computational Geometry (1997)
8. Nüchter, A., Hertzberg, J.: Towards semantic maps for mobile robots. *Robotics and Autonomous Systems* 56(11), 915–926 (2008)
9. Ohno, K., Nomura, T., Tadokoro, S.: Real-time robot trajectory estimation and 3d map construction using 3d camera. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2006)
10. Rusu, R.B., Blodow, N., Marton, Z.C., Beetz, M.: Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Human Environments. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2009)
11. Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum* 26(2), 214–226 (2007)
12. Sheh, R., Kadous, M.W., Sammut, C.: On building 3d maps using a range camera: Applications to rescue robotics. Technical report, UNSW, Sydney, Australia (2006)
13. Torr, P.H., Zisserman, A.: MLESAC: a new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding* 78(1), 138–156 (2000)
14. Weingarten, J.W., Grüner, G., Siegwart, R.: A State-of-the-Art 3D Sensor for Robot Navigation. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2004)

Providing Ground-Truth Data for the Nao Robot Platform

Tim Niemüller¹, Alexander Ferrein², Gerhard Eckel³, David Pirro³,
Patrick Podbregar⁴, Tobias Kellner⁴, Christof Rath⁴, and Gerald Steinbauer⁴

¹ Knowledge-based Systems Group,
RWTH Aachen University, Aachen, Germany

niemueller@kbsg.rwth-aachen.de
² Robotics and Agents Research Lab

University of Cape Town, Cape Town, South Africa
alexander.ferrein@uct.ac.za

³ Institute for Electronic Music and Acoustics
University of Music and Performing Arts, Graz, Austria

eckel@iem.at, pirro@iem.at

⁴ Institute for Software Technology
Graz University of Technology, Graz, Austria
{patrick.podbregar,t.kellner.c.rath}@student.tugraz.at,
steinbauer@ist.tugraz.at

Abstract. Collecting ground truth-data for real-world applications is a non-trivial but very important task. In order to evaluate new algorithmic approaches or to benchmark system performance, they are inevitable. This is particularly true for robotics applications. In this paper we present our data collection for the biped humanoid robot Nao. Reflective markers were attached to Nao's body, and the positions and orientation of its body and head were tracked in 6D with an accurate professional vision-based body motion tracking system. While doing so, the data of Nao's internal state, i.e., the readings of all its servos, the inertial measurement unit, the force receptors plus a camera stream of the robot's camera were stored for different, typical robotic soccer scenarios in the context of the RoboCup Standard Platform League. These data will be combined in order to compile an accurate ground-truth data set. We describe how the data were recorded, in which format they are stored, and show the usability of the logged data in some first experiments on the recorded data sets. The data sets will be made publicly available for the RoboCup's Standard Platform League community.

1 Introduction

Collecting ground-truth data for real-world applications is a nontrivial but very important task. In order to evaluate new algorithmic approaches or to benchmark system performance, they are inevitable. This is particularly true for robotics applications. For instance, the European Robotics Research Network (EURON) states in a *Survey and Inventory of Current Efforts in Comparative Robotics*

Research: “It was a well-known fact that the current practice of publishing research results in robotics made it extremely difficult not only to compare results of different approaches, but also to assess the quality of the research presented by the authors.” [1]. Therefore, EURON started an initiative on how results in robotics could be better compared, leading to a number of publications and workshops at international robotics conferences. One option for comparing results more easily is to pose a common and non-trivial real-world problem, which is to be solved by a number of research groups. Among them are the DARPA Grand Challenge, ELROB, or RoboCup. There is also a number of specialized tasks for establishing benchmarking objectives such as robot manipulating, motion planning, or for mobile robots. As for the latter, the Rawseed project [2] is concentrating on multi-sensor data sets for fundamental problems such as localization and navigation. With such data sets and associated ground-truth data, new localization algorithms can be benchmarked and be compared with existing approaches. Another available robotics data set is, for example, the Radish Data Set [3] or the KTH collection for simultaneous localization and mapping (SLAM) [4]¹. Standardized data sets are also common and available for machine learning approaches and computer vision (e.g. [5,6,7]).

The purpose of ground-truth data is to provide an absolute reference to a particular set of properties like the position of objects relative to the robot or the robot itself relative to the environment. The ground-truth data has to have a certain accuracy but at least some magnitudes higher than the precision of the algorithm to be evaluated in order to allow for a significant evaluation. Such data sets usually contain a set of sensor data (e.g. laser range scans, inertial measurements, camera images) and a reference position and orientation of the robot. In indoor environments usually a three-dimensional reference (a two-dimensional position plus an orientation) is sufficient. For outdoor environments usually a full six-dimensional reference (a three-dimensional position plus three rotational angles) is necessary.

In the case of a humanoid robot like the Aldebaran Nao [8], which we address in this paper, even more reference data are interesting. In order to perform a task such as playing soccer, the robot has to solve sub-tasks such as object tracking or self-localization. For these sub-tasks, the robot needs an estimation of the position and orientation of its limbs, torso, and head. This information is estimated by processing sensor data of the robot, e.g. inertial measurements, force receptors, joint angles, and camera images. Therefore, it is desirable to have ground truth for the position and orientation of the head and the torso in the reference data set. In order to provide such an extended data set, we conducted experiments in a body motion capturing system with the biped robot platform Nao on a RoboCup Standard Platform League (SPL) football field.

In this paper, we describe the gathering of ground-truth data for the robot. Unlike e.g. [9], where the ground-truth data of the robot were estimated in 3D (x, y, θ) for a localization experiment, we here aim at providing full data sets of the robot’s sensory information while performing typical tasks. In particular,

¹ More SLAM related data sets are listed at <http://www.openslam.org>



Fig. 1. The experimental setup for the data recording. The left image shows the RoboCup Standard Platform League field within motion capturing system. The right image shows one of the 15 cameras of the body motion tracking system. The LEDs at the front of the camera provide pulsed infra-red light allowing capturing under day light conditions.

the global 6D coordinates of the robot's head and torso together with the servo values and the camera images offer data sets for also benchmarking algorithms on the Nao platform or conducting off-line experiments, e.g. machine learning.

In the remainder of this paper we describe our experimental setup in Sect. 2. We particularly show the setup of the body motion capturing system and the markers that were attached to the robot during the experiments. In Sect. 3, we briefly go over how the data were logged with our robot control software FAWKES [10], before we show some first experiments on the usefulness of the collected data for benchmarking in Sect. 4. We conclude with Sect. 5.

2 Body Motion Capturing with the Robot Nao

For collecting the extended ground-truth data, we used body motion capturing technology. This technology allows to continuously track the position and orientation of parts of a human body in a three-dimensional space. Moreover, using a model of the human skeleton also the orientation and position of joints can be recorded. The technology is commonly used in movie production, animation, development of computer games, and artistic performances, where the recordings of the natural movements of a human is required.

We used the CUBE laboratory of the Institute for Electronic Music and Acoustics of the University of Music and Performing Arts in Graz. The laboratory is equipped with a high performance motion capturing system and is usually used for research on innovative forms of arts and music [11]. The body motion capturing system is the high resolution fully-3D tracking system V624 by the company Vicon. It consists of 15 infra-red cameras which records the position of a number of reflective markers attached to the body. The system is able to record the positions with a frame-rate of 120 Hz and a high spacial accuracy of less than

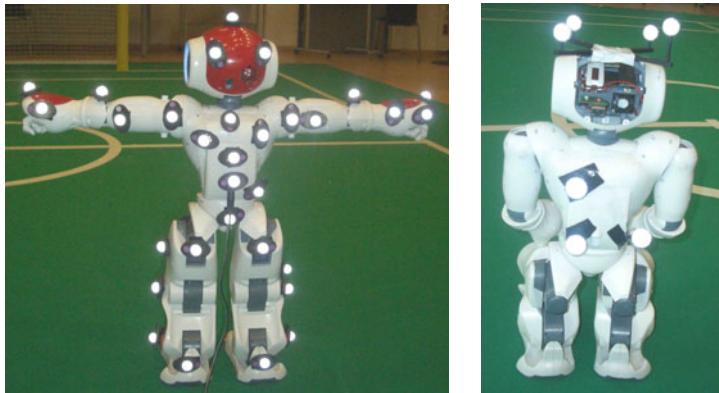


Fig. 2. The experimental setup for the humanoid robot Nao. The left image shows the robot Nao with the 42 tracking markers for a full body motion capturing. The right image shows the robot with four tracking markers each for the head and the torso (the fourth marker on the chest is not visible).

one millimeter. The system is able to track a volume of about 60 m^3 . The left image of Figure 1 shows the experimental setup of the tracking system. In order to be able to record full-featured camera images with the robot, we set up a fully rule-compliant RoboCup Standard Platform League (SPL)² field within the body motion capturing system. The right image shows one of the cameras of the tracking system. The camera uses pulsed infra-red light which allows for tracking under day light conditions. This in turn allowed us to record a video stream of the soccer field with the robot's internal camera.

We started with attaching 42 reflective markers to the robot's head, torso, and limbs, just as it would be done for human motion capturing. The left image of Fig. 2 shows the robot with the 42 markers. Unfortunately, it turned out that the robot is too small for a full body motion capturing. Because of the size of the robot, some markers especially on the back and on the feet are too close to each other, which makes it impossible for the system to reliably track these markers. The system fails to distinguish close markers and therefore regularly fails to provide tracking data. Therefore, we decided to track only the position and orientation of the head and torso of the robot. The right image of Fig. 2 shows the robot with markers only attached to the head and torso. We hoped for whole-body data, however, the ground-truth position and orientation of head and torso still provide a reasonable means for benchmarking. The upper body serves as global reference for the robot on the field, while the head provides a global reference for the position and orientation of the cameras which are the main sensors for most applications. The body motion capturing system provides the position and orientation in 6D of two separate rigid bodies modeled as cubes. The torso has its rotational point in the center, while the head has its rotational point in the head joint. Fig. 3 shows the coordinate system for the ground-truth data.

² SPL website is at <http://www.tzi.de/spl/>

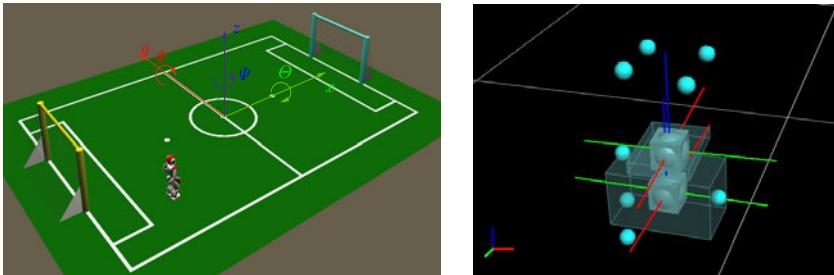


Fig. 3. The coordinate system of the tracking data. The left image shows the global coordinate system in respect to the SPL soccer field. The right image shows the coordinates for the cubes representing the position and orientation of head and torso of the robot with respect to the global coordinate system. The image depicts the alignment of the axis and markers for the robot shown on the left.

The left image of Figure 3 shows the 6D coordinate system $(x, y, z, \Theta, \Phi, \Psi)$. The center of this global coordinate system is the center of the field. The x axis runs parallel to the side lines of the field from the yellow goal towards the blue one. The positive y axis runs along the middle line to the left. The z axis points upwards. The right image shows the position and orientation of the two tracking cubes for the head (top) and the torso (bottom). The markers (spheres in the image) are shown in the true alignment with the cubes (refer to Figure 2). The rotation points of the cubes are in the torso's centers and in the head joint, resp. For the torso cube the rotation point is in the center of the upper body behind the chest button. The angles Θ, Φ, Ψ are global and are always related to the global axis. The robot on the left image of Figure 3 has its torso and head in an upright position and is aligned with the global axis. The robot is oriented along the x axis. The connecting line of the shoulder joints is parallel to the y axis. The direction of the rotation of the angles are depicted by the corresponding circular arrows.

3 Data Logging

For acquiring useful data sets, it would be desirable to log all the data the robot can provide. However, the throughput of CPU and memory is restricted on the Nao platform, and we therefore had to assess first, how much data we could log. We describe our assessment in Sect. 3.1. In Sect. 3.2, we describe how we logged the data through our robot control software FAWKES.

3.1 CPU and Memory Assessment

To determine the computational constraints for software on the Nao and the specific problems of logging, we had to investigate the basic capabilities of the robot platforms in terms of CPU speed and memory throughput.

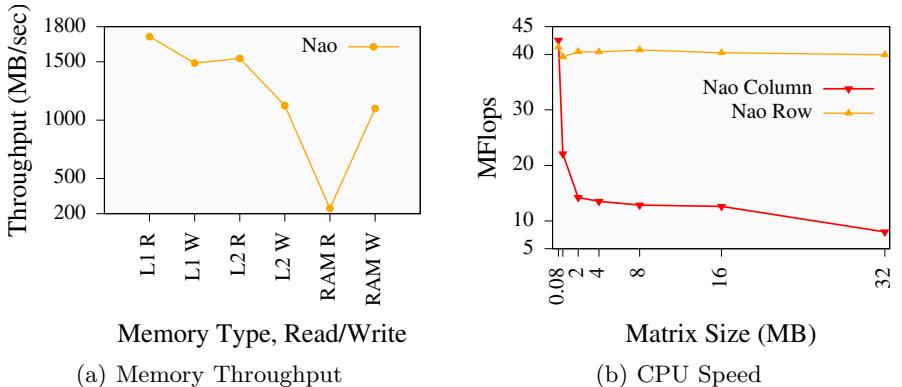


Fig. 4. CPU and Memory Benchmarks for the Nao Robot Platform

The *memory* of the computing system is divided into several stages, the main memory (RAM) and several levels of cache within the CPU. The AMD Geode LX-800 used in the Nao has two levels of cache, with the L2 cache of 128 KB having more influence on the performance. We have applied a benchmark³ to test the throughput of the different memory types. The averaged results of 10 runs are shown in Fig. 4(a). The L1 and L2 caches show the expected behavior, with a peak transfer rate of 1.7 GB/sec and 1.5 GB/sec respectively. But the performance of the RAM is much worse, dropping to only 240 MB/sec.

To measure the *CPU speed* an artificial CPU benchmark that implements a matrix-vector multiplication has been used. It implements the calculation of $A = B \cdot C$, where B is a matrix and A and C are vectors. We vary the amount of memory required to store a matrix ranging from 0.08 MB to 32 MB. We have implemented two variants of the multiplication. The first multiplies each row of the matrix B with C , allowing for a sequential access to the matrix memory. The second multiplies each value of a column of B with the appropriate element in C . This requires access to values spread throughout the whole chunk of memory.

Fig. 4(b) shows the results given in million floating point operations per second (MFlops) with double precision. The row-based implementation has an almost constant speed of about 40 MFlops for any matrix size. This indicates that the memory throughput is high enough to keep the CPU busy all of the time as long as the memory can be read optimally. This changes drastically for the column-based implementation. For the smallest matrix the speed is comparable, as the problem of 0.08 MB still fits into the L2 cache of 128 KB. But for larger matrices the computation speeds drops to 10 MFlops and below. Here we see that the bad throughput from the CPU to the RAM causes the computation units of the CPU to wait for new data most of the time.

³ Bandwidth benchmark: <http://home.comcast.net/~fbui/bandwidth.html>

3.2 Logging with Fawkes

FAWKES⁴ is a component-based robot software framework [10] developed for real-time applications in the field of cognitive mobile robotics. Functionality to operate the robot is encapsulated in plugins which can be combined and configured at run-time.

FAWKES follows a component-based approach for defining different functional blocks. We deploy a blackboard as communication infrastructure. Information is stored as semantic groups of values accessed by a unified interface and identified by a type and a unique ID, but the blackboard also supports message channels between components. The robot data as provided by NaoQi are stored in blackboard interfaces. Data is read from NaoQi at the DCM update rate of 50 Hz⁵ and transferred to FAWKES via a remote blackboard connection.

The blackboard is one of the corner stones for the data recording during the experiments. Since all the data produced by the system is stored at a central place it can be logged very efficiently. FAWKES features a generic blackboard logging component. Because all value groups follow a common interface for data access, there is no need for a specific logging mechanism, rather the plugin is only configured for the interfaces which should be logged.

We took the data logs as depicted in Fig. 5. Each log file has a preface describing the data fields contained in the log, which is given as a comma-separated value list. Each line of the ground-truth data consists of a time stamp and the 6D coordinates of head and torso. Each sensor data frame consists of a relative time offset since the start of logging, and the joint angles for all joints. A global time stamp is attached at the end of each frame. Similarly, the electric current values are stored. These data were recorded with a frequency of 50 Hz.

The camera frame were recorded separately. Each frame comes with a global time stamp followed by the camera image itself. The camera image has a resolution of 320×240 in the format YUV 422. The frequency had to be restricted to 15 Hz. With the global time stamps in data and camera frame, both logs can be synchronized.

4 Data Sets

Overall, 20 different scenarios were logged with different setups. Sometimes the robot was running towards an opponent, or the ball, sometimes the robot was just walking around recording camera data. Two example scenarios are shown in Fig. 6. The trajectory presented in these figures are drawn from the tracked (x, y) coordinates of the torso. Note that each trajectory 'wiggles' along the robot's path. The explanation for this is that the torso is tilting to the left and the right, respectively, when the robot walks. As the tracker accuracy is below 1 mm, one can find such movement in the tracked data. The spatial resolution

⁴ FAWKES is free software and available at <http://www.fawkesrobotics.org>

⁵ Experiments were conducted with NaoQi 1.3.17 providing a minimum loop time of 20 ms.

Ground Truth Log

# 6 lines header, describing the text fields													
a	b	c	d	e	f	g	h	i	k	l	m	n	\n

Hardware Log

# 8 lines header, describing the text fields																							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	\n

Hardware Current Log

# 8 lines header, describing the text fields																							
1	2	3*	4*	5*	6*	7*	8*	9*	10*	11*	12*	13*	14*	15*	16*	17*	18*	19*	20*	21*	22*	23*	24*
25*	26*	27*	28*	29*	30*	31*	32*	33*	34*	35*	36*	37*	38*	39*	40*	41*	42*	43*	44*	45*	46*	47*	\n

Camera Frame

timestamp	camera frame YUV 422, 320 × 240@15 Hz
...	
end camera frame	

Legend: ^atime stamp, ^bhead roll, ^chead pitch, ^dhead yaw, ^ehead x, ^fhead y, ^ghead z, ^htorso roll, ⁱtorso pitch, ^ktorso yaw, ^ltorso x, ^mtorso y, ⁿtorso z, ¹time offset, ²servo enabled ∈ {true, false}, ³head yaw, ⁴head pitch, ⁵l shoulder pitch, ⁶l shoulder roll, ⁷l elbow yaw, ⁸l elbow roll, ⁹l hip yaw pitch, ¹⁰l hip roll, ¹¹l hip pitch, ¹²l knee pitch, ¹³l ankle pitch, ¹⁴l ankle roll, ¹⁵r hip yaw pitch, ¹⁶r hip roll, ¹⁷r hip pitch, ¹⁸r knee pitch, ¹⁹r ankle pitch, ²⁰r ankle roll, ²¹r shoulder pitch, ²²r shoulder roll, ²³r elbow yaw, ²⁴r elbow roll, ²⁵acc x, ²⁶acc y, ²⁷acc z, ²⁸gyro x, ²⁹gyro y, ³⁰gyro ref, ³¹l FSR front l, ³²l FSR front r, ³³l FSR rear l, ³⁴l FSR rear r, ³⁵r FSR front l, ³⁶r FSR front r, ³⁷r FSR rear l, ³⁸r FSR rear r, ³⁹us reading, ⁴⁰us direction, ⁴¹l foot bumper l side, ⁴²l foot bumper r side, ⁴³r foot bumper l side, ⁴⁴r foot bumper r side, ⁴⁵chest button state, ⁴⁶battery charge, ⁴⁷DCM time stamp, note: the starred data yield the current values of the resp. sensor

Fig. 5. Data Frames

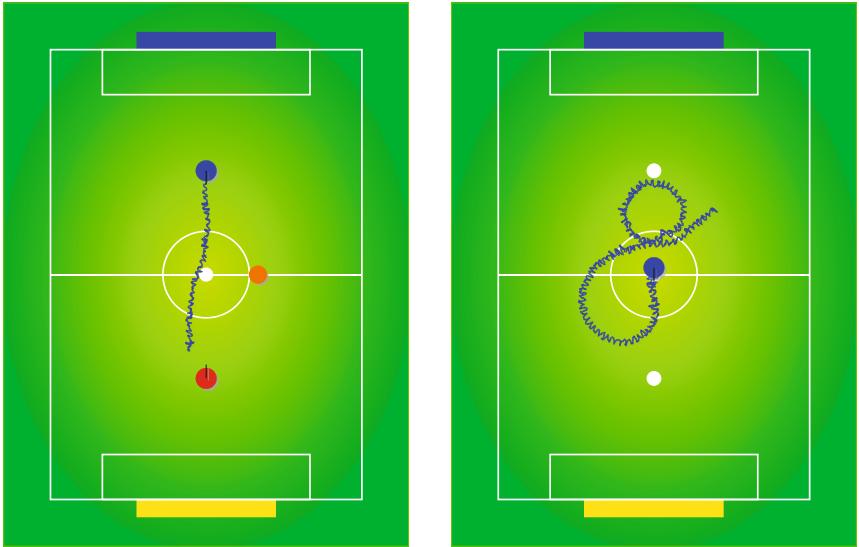


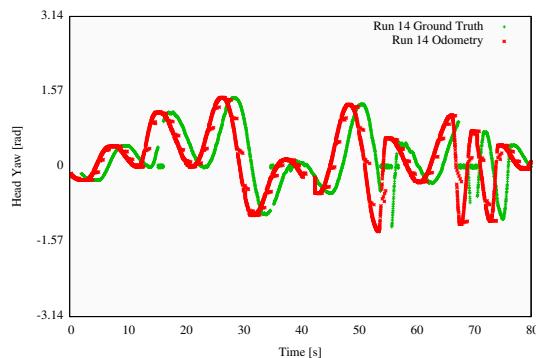
Fig. 6. Example scenarios

and accuracy of the ground truth data are determined by the motion capturing system. The time synchronization between the data of the tracking system and the data recorded by the robot was done manually. We used easy-to-find motions like stand-up as synchronization points.

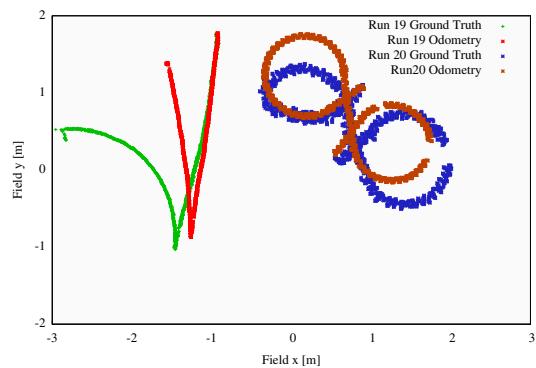
Further, we analyzed the time accuracy of our logging session. In the 20 runs of our current experiments, we could observe some variances in the aspired frame rate depending on the complexity of the task. The data were recorded at frequency of 50 Hz, i.e. 20,000 μ sec. The average frame time we could achieve with our robots for the logged data over 92,661 data frames lies at 21,245.82 μ sec, with a standard deviation of 89 μ sec.

Fig. 7 shows first experimental results on using the recorded data sets for benchmarking the robot Nao and our software. In particular, we compare the data of the head yaw that the robot logged with the globally tracked data (Fig. 7(a)), the calculated odometry of the robot's position with the tracked relative positions (Fig. 7(b)), as well as the global position, which our localization system yields, with the globally tracked torso position (Fig. 7(c)).

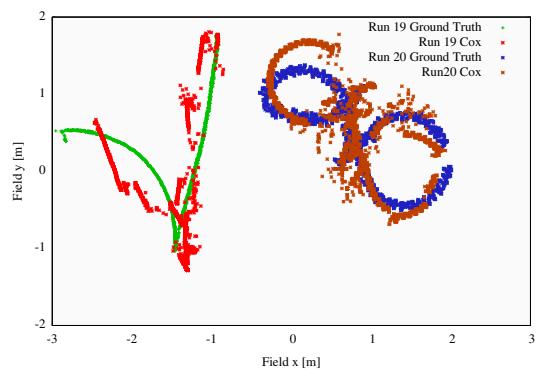
The head yaw data as depicted in Fig. 7(a) were achieved the following way. Several times, the robot was turning around itself (i.e. no translational movement), while his head was constantly moving from the left to the right. This resembles our *search-ball* behavior. The relative head yaw was computed from the tracked data as $head_yaw_{rel} = \text{normalize}(\Theta_{head} - \Theta_{torso})$ with Θ the global yaw angle (see Fig. 3 for the definition of the global coordinate system). The local data were just taken from Nao's head joint reading. Note that the global reference data in Fig. 7 are drawn with a time offset of 1 sec for readability reasons as, otherwise, the graphs would overlap with each other. Also note that



(a) Head yaw odometry vs. ground truth; note that ground truth data were shifted by 1 sec for visibility reasons.



(b) Odometry vs. ground truth



(c) Cox localization vs. ground truth

Fig. 7. Comparison between derived data and tracked data

the tracker could not always yield data for the torso and the head (gaps in the graph). Moreover, note that the yaw odometry comes from an absolute angle sensor and therefore there the errors do not accumulate like the odometry shown in Fig. 7(b)).

The other graphs show our inverse kinematic calculations of the robot's torso position compared with the globally tracked data, and the results of our localization algorithm (cf. [12] for a concise description of the implementation) for the two data sets *Run-19* and *Run-20*. These two graphs show the usefulness of the recorded data sets, in particular. We now can establish good error measures and compare future enhancements of the kinematic and the localization on these data.

5 Conclusion

In this paper we described how we recorded a ground-truth data collection for the biped humanoid robot Nao. During our experiments we logged all relevant sensor data that the robot Nao can provide, from all sensor joint angles over the joint currents up to the camera stream. An assessment of the throughput of the robot's hardware revealed that the camera stream could not be recorded with a desired 30 Hz at full VGA resolution, but only with 15 Hz at 320×240 pixels. The sensor data were logged with a frequency of 50 Hz, and data analysis showed that this frequency could be satisfactorily achieved in over 90,000 frames. The respective ground truth to our experiments were established by means of a professional motion capturing system that can track human body motion in 6D with a sub-millimeter accuracy. 15 pulsed infra-red cameras were used to track a the head and the torso of the Nao. This technology allowed also for logging the camera images of the robot. In this environment, we set up a football field fully compliant with the RoboCup 2009 SPL rules, and recorded 20 different runs.

As we conducted the experiments very recently, we are still processing the data for compiling the single logs into a coherent data collection. The data sets are available at http://www.ist.tugraz.at/nao_gt. Moreover, we are going to develop tool support to provide the data in different formats for allowing real-time playback. By now, we provide the data as comma-separated value text files and in a binary format as it was logged by our robot control software FAWKES. A tool for converting these binary data into comma-separated value files exists already.

In first experiments we showed the plausibility of the data, and that these data sets are meaningful and can be beneficially used for benchmarking both, the robot's hardware as well as the own control software. We compared the robot's head yaw with the globally tracked data and our odometry and pose estimates calculations with the global reference data.

With this paper we hope to start an initiative for benchmarking the platform Nao similar to project like Rawseed by providing reference data sets. This should help for comparing the approaches of the different research groups that work with the Nao.

Acknowledgments

Alexander Ferrein is currently a Feodor Lynen fellow supported by a grant of the Alexander von Humboldt Foundation. This research is also partly supported by the International Bureau of the German Ministry of Education and Research with grant no. SUA 07/031. We would like to thank the anonymous reviewers for their helpful comments.

References

1. Euron - European Robotics Search Network: Survey and inventory of current efforts in comparative robotics research,
<http://www.robot.uji.es/EURON/en/index.htm>
2. Fontana, G., Matteucci, M., Sorrenti, D.: The RAWSEEDS Proposal for Representation-Independent Benchmarking of SLAM. In: Workshop on Good Experimental Methodologies in Robotics, co-located with Robotics, Science and Systems (2008)
3. Howard, A., Roy, N.: The Robotics Data Set Repository (Radish) (2003)
4. Folkesson, J.: Kth slam data sets,
<http://www.nada.kth.se/~johnf/kthdata/dataset.html>
5. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
6. Ponce, J.: Datasets for computer vision research,
http://www-cvr.ai.uiuc.edu/ponce_grp/data/
7. Lawrence Berkeley National Laboratory: Berkeley image library,
<http://www.lbl.gov/image-gallery/image-library.html>
8. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: The nao humanoid: a combination of performance and affordability. CoRR abs/0807.3223 (2008)
9. Laue, T., de Haas, T.J., Burchardt, A., Graf, C., Röfer, T., Härtl, A., Rieskamp, A.: Efficient and reliable sensor models for humanoid soccer robot self-localization. In: Proceedings of the 4th Workshop on Humanoid Soccer Robots (Humanoid 2009) (2009)
10. Niemueler, T.: Developing A Behavior Engine for the FAWKES Robot-Control Software and its Adaptation to the Humanoid Platform Nao. Master's thesis, Knowledge-Based Systems Group, RWTH Aachen University (2009)
11. Eckel, G., Pirro, D., Sharma, G.K.: Motion-Enabled Live Electronics. In: Proceedings of the 6th Sound and Music Computing Conference, Porto, Portugal (2009)
12. Rath, C.: Self-localization of a biped robot in the RoboCup domain. Master's thesis, Institute for Software Technology, Graz University of Technology (2010)

Optimizing Particle Filter Parameters for Self-localization

Armin Burchardt¹, Tim Laue², and Thomas Röfer²

¹ Universität Bremen, Fachbereich 3 – Mathematik und Informatik,
Postfach 330 440, 28334 Bremen, Germany
armin@informatik.uni-bremen.de

² Deutsches Forschungszentrum für Künstliche Intelligenz,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
{Tim.Laue,Thomas.Roefer}@dfki.de

Abstract. Particle filter-based approaches have proven to be capable of efficiently solving the self-localization problem in RoboCup scenarios and are therefore applied by many participating teams. Nevertheless, they require a proper parametrization – for sensor models and dynamic models as well as for the configuration of the algorithm – to operate reliably. In this paper, we present an approach for optimizing all relevant parameters by using the Particle Swarm Optimization algorithm. The approach has been applied to the self-localization component of a Standard Platform League team and shown to be capable of finding a parameter set that leads to more precise position estimates than the previously used hand-tuned parametrization.

1 Introduction

In recent years, the application of particle filters to robot localization, the so-called Monte-Carlo Localization (MCL) approach [4], gained immense popularity and is also a commonly used approach in different RoboCup scenarios since it is able to efficiently cope with non-linear dynamics, multi-modal probability distributions, and the *kidnapped robot problem*. Especially in soccer leagues with local directed vision systems such as the Standard Platform League (SPL) and the Humanoid KidSize League it can be considered as being the de facto standard approach for self-localization. At RoboCup 2009, it was used by all top three teams of the KidSize League – *Darmstadt Dribblers* [5], *FUmanoid* [13], and *CIT Brains* [7] – as well as by two of the top three teams of the SPL – *B-Human* [15] and *Nao Devils Dortmund* [3]. Only the runner-up *Northern Bites* [8] used an EKF-based approach [16]. In other RoboCup Soccer leagues, self-localization is not a topic of research – the Small Size League provides a global view on the whole relevant environment [17] – or can be solved in a different way – the Middle Size League allows omni-directional vision systems that make iterative matching approaches as [11] an efficient solution.

Nevertheless, successfully applying particle filters demands a proper configuration of a variety of different parameters. This affects the sensor models as well

as different parameters that control the sensor resetting [12] that is needed to recover from kidnapping situations. The major contribution of this paper is an optimization approach that is able to optimize all relevant parameters in order to obtain more precise position estimates. In general, adequate sensor models can be learned as described in [16] but in this case, they have to be optimized together with the algorithm's parameters due to the strong interdependencies between sensor models and sensor resetting. Our work is based on the Particle Swarm Optimization (PSO) approach by [9] that is able to provide results faster than evolutionary algorithms [1]. PSO has already been applied successfully to gait optimization in the RoboCup context, e.g. by [14]. Since the proposed optimization procedure is computationally intensive, a minor contribution of this work is a modification of the original PSO algorithm that leads to a faster convergence near an optimum, in case of noisy problems. Please note that due to the use of stochastic factors in the algorithm and the noisy evaluation, the particles may converge to a local optimum and a better parameter set may remain undiscovered.

This paper is organized as follows: Section 2 describes the particle filter-based approach and all parameters that need to be optimized. The optimization process as well as the applied Particle Swarm Optimization algorithm are presented in Sect. 3. The conducted experiments and their results are described in Sect. 4.

2 Particle Filter-Based Self-localization

The implementation to become optimized is using an adapted version of the Augmented MCL approach by [6] that extends the standard MCL approach by an adaptive sensor resetting component. This section provides a brief overview of the algorithm and describes in detail all parameters that need to become optimized for obtaining more accurate position estimates.

2.1 Basic Localization Algorithm

Using the Monte-Carlo Localization approach, a set of samples representing the belief of the robot is sequentially updated with the estimate of the current action and weighted by applying likelihood functions of the observed field features. Resampling the samples according to their weightings is used to reduce the number of required samples to approximate arbitrarily probability densities.

In scenarios that include the possibility of robot kidnappings, such as most RoboCup robot soccer leagues, a mechanism for position recovery is needed. In general, the sensor resetting approach by [12] is applied; it generates new samples based on recent sensor measurements and can easily be integrated into the MCL resampling step (cf. line 13 of Algorithm 1). The Augmented MCL approach includes a mechanism for controlling the number of new samples by keeping track of the sample set's overall weighting over time. Recent changes can be interpreted as a delocalization that requires a resetting. In detail, the probability for inserting new samples is computed by using two averaged weightings w_{slow} and w_{fast}

Algorithm 1. Augmented MCL, derived from [16, p.258]

```

function Augmented_MCL( $X_{t-1}, t_y, z_t$ ):
1: static  $w_{\text{slow}}, w_{\text{fast}}$ 
2:  $\bar{X}_t = \emptyset, X_t = \emptyset, w_{\text{avg}} = 0$ 
3: for  $m = 1$  to  $M$  do
4:    $x_t^{[m]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$ 
5:    $w_t^{[m]} = \text{measurement\_model}(z_t, x_t^{[m]})$ 
6:    $\bar{X}_t = \bar{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$ 
7:    $w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M} w_t^{[m]}$ 
8: end for
9:  $w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$ 
10:  $w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$ 
11: for  $m = 1$  to  $M$  do
12:   with probability  $\max\{0, 1 - w_{\text{fast}}/w_{\text{slow}}\}$ 
13:     add random pose to  $X_t$ 
14:   else
15:     draw  $i$  with probability  $\propto w_t^{[i]} + \text{resampling\_threshold} \cdot w_{\text{avg}}$ 
16:     add  $x_t^{[i]}$  to  $X_t$ 
17:   endwith
18: end for
19: return  $X_t$ 

```

of the mean sample weighting w_{avg} . The ratio is calculated using two scalars α_{fast} and α_{slow} that control the reactivity of this mechanism. These two parameters are subject to the optimization process. One additional parameter is *resampling_threshold* which prevents the effect of particle depletion which might occur easily in case of small sample sets.

The complete algorithm is shown in Algorithm 1.

2.2 Motion Model

The pose of each sample is updated in every cycle using odometry data. Since the data is error-prone a random offset is added (Algorithm 2) depending on the motion speed. In addition to a minimum white noise, factors $(\lambda_x, \lambda_y, \lambda_\theta)$ are used to scale the level of the noise in respect to the motion speed $(\Delta_x, \Delta_y, \Delta_\theta)$:

$$\lambda_x = \max\{\Delta_x \text{Noise}_{\text{TranslationMajorDirWeight}}, \Delta_y \text{Noise}_{\text{TranslationMinorDirWeight}}, \text{Noise}_{\text{Translation}}\} \quad (1)$$

$$\lambda_y = \max\{\Delta_x \text{Noise}_{\text{TranslationMinorDirWeight}}, \Delta_y \text{Noise}_{\text{TranslationMajorDirWeight}}, \text{Noise}_{\text{Translation}}\} \quad (2)$$

$$\lambda_\theta = \max\{|\Delta_x + \Delta_y| \text{Noise}_{\text{RotationMovedDistWeight}}, \Delta_\theta \text{Noise}_{\text{RotationMovedAngleWeight}}, \text{Noise}_{\text{Rotation}}\} \quad (3)$$

All six Noise_* parameters are subject to the optimization process.

Algorithm 2. Noisy update of the state hypotheses (x_t, y_t, θ_t) with action Δ . The noise is scaled by the λ_* argument to the random function *sample*.

```

function sample-motion-model( $\Delta, \langle x_{t-1}, y_{t-1}, \theta_{t-1} \rangle$ ) :
  1:  $\begin{pmatrix} x_t^{[m]} \\ y_t^{[m]} \end{pmatrix} \leftarrow \begin{pmatrix} x_{t-1}^{[m]} \\ y_{t-1}^{[m]} \end{pmatrix} + R_{t-1}^{[m]} \begin{pmatrix} \Delta_x + \text{sample}(\lambda_x) \\ \Delta_y + \text{sample}(\lambda_y) \end{pmatrix}$ 
  2:  $\theta_t^{[m]} \leftarrow \theta_{t-1}^{[m]} + \Delta_\theta + \text{sample}(\lambda_\theta)$ 
  3: return  $\langle x_t, y_t, \theta_t \rangle$ 

```

2.3 Measurement Models

To calculate a weighting for each sample used in the particle filter, a set of measurement models is applied to the features extracted from the camera images. The current implementation of the vision system is able to perceive goal posts, lines, line crossings, and the center circle of a standard SPL field. A detailed description of the system is given in [10].

The implemented sensor models are configured via standard deviations for the likelihood of errors in measured sensor data. An example of the perception of a goal post is depicted in Figure 1: The angle α between the upper and lower post points (p_{upper}) and (p_{lower}) is used to calculate the distance d . Estimation errors in respect to α are rated by the sensor model using the density function of a normal distribution with the standard deviation $\sigma_{\text{GoalpostSizeDistance}}$. In Fig. 1b, the observed angle β to a goal post is compared to the assumed measurement. The difference is used to rate the likelihood of sample s_i based on the standard deviation $\sigma_{\text{GoalpostSizeAngle}}$.

An overview of all standard deviations which are subject to the optimization process used by the measurement models is provided in Tab. 1.

2.4 Sensor Resetting

For the generation of new samples, only perceptions of goal posts are used since this is the only kind of feature providing a unique global direction. To avoid the placement of all new samples on the same spot within one execution cycle, a measurement used for resetting becomes distorted by an uncorrelated random error scaled by $\sigma_{\text{GoalpostSampleBearingDistance}}$ or $\sigma_{\text{GoalpostSampleSizeDistance}}$, depending on the kind of measurement provided by the vision system.

3 Particle Swarm Optimization

To optimize the parameters in Table 1, the Particle Swarm Optimization [9] approach is used as it is known to quickly find parameters in the neighborhood around the minimum of a function (cf. [1]). This section briefly explains the general approach, an extension to PSO that had to be added for this scenario, the applied benchmark function, and the setup for efficient distributed optimization.

3.1 General Approach

Optimization in this context means to find a parameter set x with the lowest localization error $f(x)$:

$$x = \operatorname{argmin}_{x_i} f(x_i)$$

In order to optimize parameters for a self-localization scenario, every parameter set needs to be tested against the same sequence of observations. To assure this condition, the whole optimization process is carried out offline using recorded log data. To compute $f(x)$, the data is augmented with reference results obtained from a global tracking system as described in Sect. 4.2.

The algorithm uses a set of particles P to find results in the given search space (Algorithm 3). Each particle p has a memory $p_{bestpos}$ for the best parameter set found in the previous iterations. The first iteration is randomly initialized within given limits (Table 2). Neighborhoods are used to exchange information between the particles. The velocity p_{vel} is updated with a momentum to direct a particle's position towards a randomized combination of the best position found by itself and the best position known in its neighborhood.

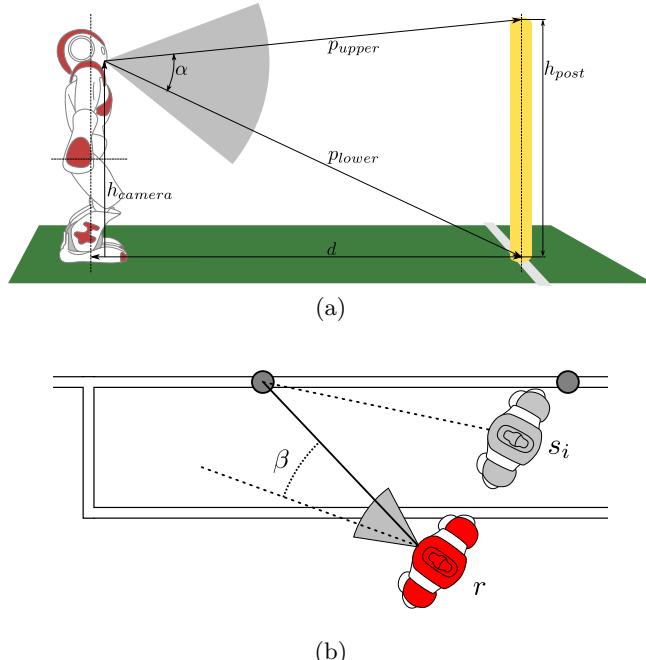


Fig. 1. a) Sensor model for distance estimation by observed goal post height α b) Measurement error β for hypothetical field position s_i , the real robot position is r

Table 1. Parameters for *Augmented MCL* algorithm (upper) and implemented sensor models (lower part). All parameters have been determined empirically by human experts.

parameter	value	parameter	value
α_{slow}	0.0059	α_{fast}	0.0060
<i>resampling_threshold</i>	4.0		
$\sigma_{\text{FieldLines}}$	512 rad/1024	σ_{Corners}	512 rad/1024
$\sigma_{\text{GoalpostAngle}}$	0.02 rad	$\sigma_{\text{GoalpostBearingDistance}}$	0.4 rad
$\sigma_{\text{GoalpostSizeDistance}}$	0.2 rad	$\sigma_{\text{GoalpostSampleBearingDistance}}$	150.0 mm
$\sigma_{\text{GoalpostSampleSizeDistance}}$	150.0 mm	$\sigma_{\text{CenterCircleAngle}}$	0.2 rad
$\sigma_{\text{CenterCircleDistance}}$	0.4 rad	$\text{Noise}_{\text{Translation}}$	25.0 mm
$\text{Noise}_{\text{TranslationMajorDirWeight}}$	2.0	$\text{Noise}_{\text{TranslationMinorDirWeight}}$	1.0
$\text{Noise}_{\text{Rotation}}$	0.1 rad	$\text{Noise}_{\text{RotationMovedAngleWeight}}$	1.0
$\text{Noise}_{\text{RotationMovedDistWeight}}$	0.002		

3.2 Modification for Noisy Data

Noisy results are apparent in an environment where a simulation or real-world experiments are used to evaluate the performance of a specific parameter set. Since PSO has a memory for the best result found so far, it becomes irritated by outliers. A modification (cf. Algorithm 4) has been made to the PSO algorithm (cf. Algorithm 3) in line 4 to lessen this effect. After the evaluation of a parameter set, the result is compared with the particle's best result as in the original implementation. If the memory is not updated with a better position, the previous rating is degraded by a configurable factor κ . This allows the algorithm to escape from a local minimum over time.

Algorithm 3. *Particle Swarm Optimizer* (cf. [2, Page 80ff.]).

```

function PSO():
1: while continueSearch do
2:   for  $p \in P$  do
3:      $p_{\text{result}} \leftarrow f(p_{\text{pos}})$ 
4:     if  $p_{\text{bestresult}} > p_{\text{result}}$  then
5:        $p_{\text{result}} \leftarrow p_{\text{bestresult}}$ ;       $p_{\text{bestpos}} \leftarrow p_{\text{pos}}$ 
6:     end if
7:   end for
8:   for  $p \in P$  do
9:      $p_{\text{neighbestpos}} \leftarrow \text{FindBestNeighbour}(p)$ 
10:     $p_{\text{vel}} \leftarrow c_1 r_1 p_{\text{vel}} + c_{\max} r_2 (p_{\text{bestpos}} - p_{\text{pos}}) + c_{\max} r_3 (p_{\text{neighbestpos}} - p_{\text{pos}})$ 
11:     $p_{\text{pos}} \leftarrow p_{\text{pos}} + p_{\text{vel}}$ 
12:  end for
13: end while

```

Algorithm 4. Modification to PSO.

```

1: if  $p_{bestresult} > p_{result}$  then
2:    $p_{bestresult} \leftarrow p_{result}; \quad p_{bestpos} \leftarrow p_{pos}$ 
3: else
4:    $p_{bestresult} \leftarrow p_{bestresult}(1 + \kappa)$ 
5: end if

```

3.3 Benchmark Function

To rate a parameter set x , a benchmark function is required to estimate an error $f(x)$. In this scenario, the Euclidean distance between the estimated field position $p(t, x)$ using parameter set x and the reference position $p'(t)$ is used for every data set corresponding to a captured camera image t (frame). The camera is able to capture images at a rate of 30 Hz, some frames are missing ground truth information due to a fallen robot or obscured marker and are therefore ignored for parameter benchmarking. The self-localization is conducted using the complete input data stored in a log file for one parameter set under test. The image processing is only conducted once while recording the log file and storing the extracted perceptions. We used the distance as the error per frame but it is also possible to use a linear combination of the square or logarithmic translational and rotational distances. The mean of the per frame errors is used as parameter rating for a single log file evaluation:

$$f(x) = \frac{\sum_{t=1}^{\text{frames}} \|p(t, x) - p'(t)\|}{\text{frames}}.$$

3.4 Distributed Computing

Since the evaluation of a previously recorded log file consisting of about 17000 frames is computationally expensive, the implementation of the optimizer supports distribution by using multiple processes controlled over TCP/IP network connections. The number of particles determines the maximum number of processes. In the current implementation, all parameters have to be evaluated before the velocity and position can be updated to generate a new set of particles.

4 Experiments and Results

The proposed approach has been applied to recorded game scenes with two Nao robots in each team. During the experiments, all perceived data of one robot became recorded and merged with reference data from a global tracking system.

4.1 Evaluation of PSO Modification

Before using the proposed PSO modification for optimization, it was successfully tested with *Griewank30D*, a function commonly used for optimization

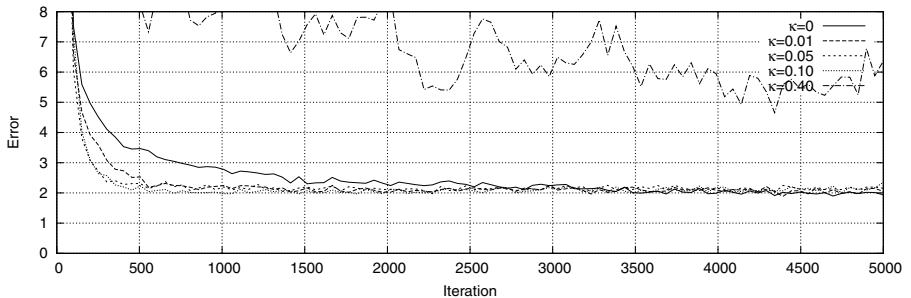


Fig. 2. Benchmark function $f_{\text{Griewank30D}}$ with added noise $\mathcal{N}_{0,1}$, search space is $[-300; 300]^{30}$

benchmarks (Figure 2). The slow degrading of old ratings ($\kappa = 0.01, \kappa = 0.05$) accelerates the optimization within the first 3000 frames in comparison to the non-modified PSO ($\kappa = 0$). The performance is significantly worse for high levels ($\kappa = 0.4$) of degradation.

4.2 Global Tracking System

To rate the performance of a parametrization the SSL-Vision application (cf. [17]) is used as a ground truth system. A camera is mounted above the field (Figure 3a) to keep track of a marker fixed on the head of the Nao. The SSL-Vision is calibrated to provide the projection of the marker position to the ground in a global coordinate system. This information is transmitted to the robot via WLAN, the offset caused by tilt of the robot body and different head joint angles is compensated using filtered sensor readings of the accelerometer, gyroscope, and measured joint angles. The result is used as ground truth position (Figure 3b).

4.3 Optimization

The Particle Swarm Optimizer is configured as follows: number of particles: 40, full neighborhood, $c_1 = 0.69$, $c_{\max} = 1.43$, $\kappa = 0.1$. The values for c_1 and c_{\max} are suggested in [2], a large neighborhood, the amount of particles and the value used for degradation of results (κ) were found to produce good results in previously conducted experiments. Modifications of these parameters may improve the performance of the optimizer but they are computationally expensive to evaluate. Two log files containing recognized field features like goal posts and field lines out of ten minutes of game play are used as training data. Each evaluation is repeated three times, the mean of these measurements is used as parameter rating. The optimization progress was canceled after 41 iterations due to a lack of progress. The best parameter set was selected by re-evaluating all particles of the last iteration for 16 times. The position of the particles in the search space are displayed in Figure 4a using the example of α_{slow} and α_{fast} , the rating of the particles in each iteration is shown in Figure 4b.

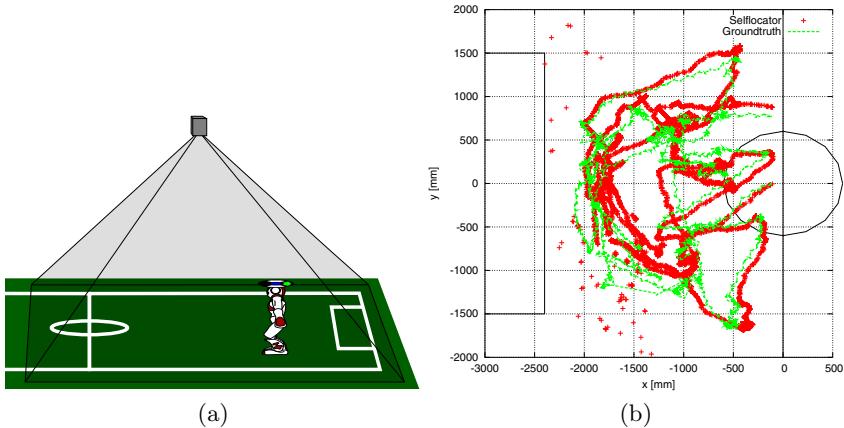


Fig. 3. a)Schematic view of the ground truth setup used b)Comparison of ground truth data with self-localization

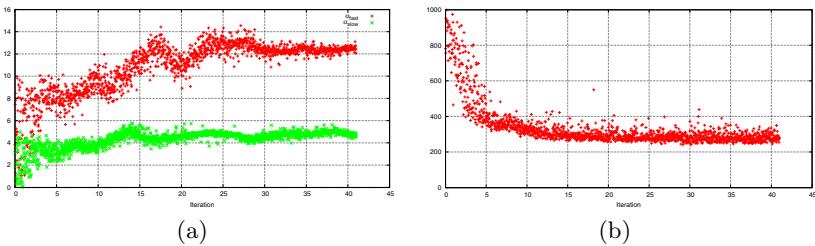


Fig. 4. a) Particle position for α_{slow} and α_{fast} during optimization b) Result of evaluation of particles per iteration step

4.4 Interpretation of Results

By comparing the optimized parameter values in Table 2 to the original values in Table 1, big differences are apparent. The large values for α_{slow} and α_{fast} suggest that insertion of new samples by sensor resetting or randomized field positions is used to a greater extend. The small values were originated from a scenario with more landmarks used for sensor resetting and thus introducing more ambiguous state hypotheses.

The *resampling_threshold* was reduced to 0, indicating that the modification of the Augmented MCL in line 15 of Algorithm 1 in comparison to the Augmented MCL in [16] to support particles with low weightings had no significant benefit.

The σ_* values were mostly adjusted to larger values with the exception of $\sigma_{GoalpostBearingDistance}$ and $\sigma_{CenterCircleDistance}$. These higher standard

Table 2. Parameter set of best particle after 41 iterations

parameter	initial range	value
α_{slow}	0.0001 – 5.0	4.807
α_{fast}	0.0001 – 10.0	12.43
<i>resampling_threshold</i>	0.0 – 100.0	0.0
$\sigma_{\text{FieldLines}}$	16 – 4000 rad/1024	1883.0 rad/1024
σ_{Corners}	16 – 4000 rad/1024	747.8 rad/1024
$\sigma_{\text{GoalpostAngle}}$	0.01 – 2.0 rad	0.96 rad
$\sigma_{\text{GoalpostBearingDistance}}$	0.01 – 2.0 rad	0.32 rad
$\sigma_{\text{GoalpostSizeDistance}}$	0.01 – 2.0 rad	0.436 rad
$\sigma_{\text{GoalpostSampleBearingDistance}}$	0.01 – 2000.0 mm	1878.0 mm
$\sigma_{\text{GoalpostSampleSizeDistance}}$	0.01 – 15000.0 mm	8418.0 mm
$\sigma_{\text{CenterCircleAngle}}$	0.01 – 2.0 rad	0.402 rad
$\sigma_{\text{CenterCircleDistance}}$	0.01 – 2.0 rad	0.08 rad
Noise _{Translation}	0.0 – 100.0 mm	115.2 mm
Noise _{TranslationMajorDirWeight}	0.0 – 20.0	5.11
Noise _{TranslationMinorDirWeight}	0.0 – 20.0	1.13
Noise _{Rotation}	0.0 – 1.0 rad	0.072 rad
Noise _{RotationMovedAngleWeight}	0.0 – 10.0	2.7
Noise _{RotationMovedDistWeight}	0.0 – 0.1	0.0

deviations can be interpreted as a certain compensation for the elimination of *resampling_threshold* as they lead to more homogeneous weightings within a sample set.

The uncertainty introduced in the motion update is slightly reduced for the minimum rotational noise and nearly unchanged concerning the factor for rotational noise by moved distance. The ratios for random translational errors by moved distance (Noise_{TranslationMajorDirWeight}, Noise_{TranslationMinorDirWeight}) and the velocity independent minimum error Noise_{Translation} are increased.

After the optimization, the overall precision of the self-localization has been increased. Comparisons are shown in Table 3 and Figure 5.

Table 3. The performance of the self-localization is compared by reprocessing data from a log file not used for optimization 100 times

log file	min	max	mean error
log file A (original)	242.7 mm	332.2 mm	292.3 mm ($\pm 6.1\%$)
log file A (optimized)	160.0 mm	222.2 mm	188.9 mm ($\pm 8.4\%$)
log file B (original)	283.5 mm	364.7 mm	307.7 mm ($\pm 6.0\%$)
log file B (optimized)	154.2 mm	231.0 mm	198.5 mm ($\pm 8.9\%$)

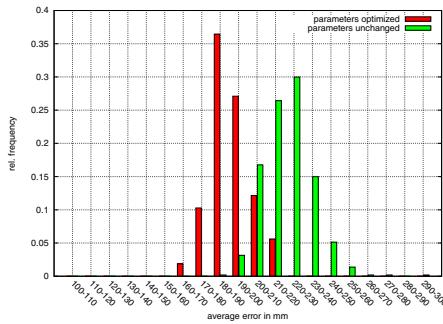


Fig. 5. Histogram of mean errors used for the evaluation of a parameter set (log file was used in training data set)

5 Conclusions and Future Work

In this paper, we presented an approach for optimizing the parameters of a robot's self-localization component. The approach has been successfully applied to a Standard Platform League game scenario. The results did not only lead to a more precise position estimate but also provided insights into the necessity of certain parameters of the applied localization algorithm.

Due to computational constraints set by the robot hardware, the number of samples (which is currently set to 100) used in the particle filter has not been subject to the conducted optimization. As the runtime of the self-localization scales linearly with the number of samples, including this parameter would also demand a more complex evaluation function for the optimizer, trading off computing time versus precision.

In the future, more experiments have to be conducted to study the effects of different error functions (i. e. weighted mean of translational and rotational error). Another interesting extension might also be the inclusion of more variables, e. g. those controlling the image processing modules.

References

1. Angeline, P.J.: Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: Porto, V.W., Saravanan, N., Waagen, D., Eiben, A.E. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)
2. Clerc, M.: Particle Swarm Optimization. ISTE, London (2006)
3. Czarnetzki, S., Kerner, S.: Nao Devils Dortmund - team description for robocup 2009. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 58–68. Springer, Heidelberg (2010)
4. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte-Carlo localization: Efficient position estimation for mobile robots. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, Orlando, FL, USA, pp. 343–349 (1999)

5. Friedmann, M., Petersen, K., Petters, S., Radkhah, K., Scholz, D., Thomas, D., von Stryk, O.: Darmstadt Dribblers - team description for humanoid kidsize league of robocup 2009. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949. Springer, Heidelberg (2010)
6. Gutmann, J.S., Fox, D.: An experimental comparison of localization methods continued. In: Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002), Lausanne, Switzerland, vol. 1, pp. 454–459 (2002)
7. Hayashibara, Y., Minakata, H., Seike, Y., Ogura, S., Ichizawa, K., Machi, K., Takamatsu, K., Yoshizawa, S., Yamada, Y., Horiuchi, T., Fukuta, M., Fujita, S., Irie, K., Kaminaga, H., Kawakami, T., Nishizaki, M.S.Y., Sakamoto, H.: CIT BRains (kid size league). In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949. Springer, Heidelberg (2010)
8. Hermans, T., Strom, J., Slavov, G., Morrison, J., Lawrence, A., Krob, E., Chown, E.: Northern Bites 2009 team report (2009)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, Piscataway, NJ, USA, vol. 4, pp. 1942–1948 (1995)
10. Laue, T., de Haas, T.J., Burchardt, A., Graf, C., Röfer, T., Härtl, A., Rieskamp, A.: Efficient and reliable sensor models for humanoid soccer robot self-localization. In: Zhou, C., Pagello, E., Menegatti, E., Behnke, S., Röfer, T. (eds.) Proceedings of the Fourth Workshop on Humanoid Soccer Robots in Conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots, Paris, France, pp. 22–29 (2009)
11. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: An efficient and accurate approach for robot self-localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 142–153. Springer, Heidelberg (2006)
12. Lenser, S., Veloso, M.: Sensor resetting localization for poorly modeled mobile robots. In: Proc. of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000), San Francisco, CA, USA, vol. 2, pp. 1225–1232 (2000)
13. Moballegh, H.R., Hohl, G., Landgraf, T., Fischer, B., Langner, T., Fassbender, T., Otte, S., Stoll, K., Tuchscherer, A., Steig, D., Heinrich, S., Mielke, S., Seifert, D., Kukulski, M., Kahlert, B., Rojas, R.: FUMANOID team description 2009. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949. Springer, Heidelberg (2010)
14. Niehaus, C., Röfer, T., Laue, T.: Gait optimization on a humanoid robot using particle swarm optimization. In: Pagello, E., Zhou, C., Menegatti, E., Behnke, S. (eds.) Proceedings of the Second Workshop on Humanoid Soccer Robots in Conjunction with the 2007 IEEE-RAS International Conference on Humanoid Robots, Pittsburgh, PA, USA (2007)
15. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.H.: B-Human team report and code release 2009 (2009), http://www.b-human.de/download.php?file=coderelease09_doc
16. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
17. Zickler, S., Laue, T., Birbach, O., Wongphati, M., Veloso, M.: SSL-Vision: The Shared Vision System for the RoboCup Small Size League. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 425–436. Springer, Heidelberg (2010)

Improving People Awareness of Service Robots by Semantic Scene Knowledge

Jörg Stückler and Sven Behnke

University of Bonn

Computer Science Institute VI, Autonomous Intelligent Systems

Roemerstr. 164, 53117 Bonn, Germany

stueckler@ais.uni-bonn.de, behnke@cs.uni-bonn.de

Abstract. Many mobile service robots operate in close interaction with humans. Being constantly aware of the people in the surrounding of the robot thus poses an important challenge to perception and behavior design.

In this paper, we present an approach to people awareness for mobile service robots that utilizes knowledge about the semantics of the environment. The known semantics, e.g., about walkable floor, chairs, and shelves, provides the robot with prior information. We utilize information about the a-priori likelihood that people are present at semantically distinct places. Together with reasonable face heights inferred from scene semantics, this information supports robust detection and awareness of people in the robot's environment. For efficient exploration of the environment for people, we propose a strategy which chooses search locations that maximize the expected detection rate of new persons.

We evaluate our approach with our domestic service robot that competes in the RoboCup@Home league.

1 Introduction

Today's industrial mass production would not be possible without the invention of robots that efficiently and precisely carry out repetitive manufacturing tasks. Just as manufacturing tasks, many of our everyday tasks are monotone, cumbersome or even dangerous. The development of autonomous service robots that one day might relieve humans from these kinds of tasks will thus attain significant importance in the future.

The requirements for service robots differ vastly from those of industrial applications. Manufacturing robots work in an isolated static environment where they fulfill specific tasks. Service robots, on the other hand, need to work in dynamic environments in close interaction with humans. Being constantly aware of the people in the surrounding of the robot thus poses an important challenge to perception and behavior design.

In this paper, we present an approach to people awareness for mobile service robots that utilizes knowledge about the semantics of the environment. The known semantics, e.g., knowledge about the location and properties of furniture

like chairs, provides important hints on the presence likelihood and the appearance of persons. In our approach, we exploit such kind of information in various ways. We estimate the presence probability of persons and incorporate semantic knowledge as a prior to the estimation. The presence belief and the knowledge about the appearance of persons enables us to discover false positive detections of persons which is a frequent problem of person detection methods. When the robot needs to explore its environment, we utilize the estimated presence belief to select promising search locations that maximize the expected detection rate of new persons. We evaluate our approach with our domestic service robot that competes in the RoboCup@Home league.

The remainder of this paper is organized as follows: After a brief review of related work in Sec. 2, we outline our method for detecting and tracking multiple persons in the robot’s vicinity in Sec. 3. We detail our approach to estimating the presence of persons and how to improve people awareness by semantic scene knowledge in Sec. 4. In Sec. 5 we introduce an efficient method to explore the environment for new people. We evaluate our approach in experiments in Sec. 6.

2 Related Work

Tracking people with laser range finders is a well studied topic in mobile robotics (e.g., [1,2]). Many approaches detect and track legs of people and fuse this information in a multi-hypothesis tracker [3].

The computer vision community developed a variety of methods for tracking multiple persons with camera systems. For statically mounted cameras (e.g., [4,5]), background subtraction can be applied to improve the robustness of tracking. When the camera moves (as in [6,7,8,9]), subtracting background is no longer possible. Instead, robust person detectors are required that provide stable information for tracking.

Some approaches use spatial and semantic information to improve tracking robustness. In [7], information about tracked objects is fed back as semantic information into visual odometry to improve the robustness of the overall multi-person tracking system. Luber et al. [10] learn Poisson process models of the occurrence of persons throughout the environment. They demonstrate improved tracking accuracy with their model compared to standard multi-hypothesis tracking approaches. We propose to use semantic scene knowledge derived from the environment to increase the robustness of people awareness.

3 Continuous People Awareness

For detection and tracking of multiple persons we combine measurements from two laser range finders (LRFs) and a camera. With the LRFs we continuously detect and keep track of possible persons. One LRF is mounted shortly above the ground at a height of 24 cm and detects legs of people. We additionally detect torsos of people with a second LRF at a height of approx. 80 cm.

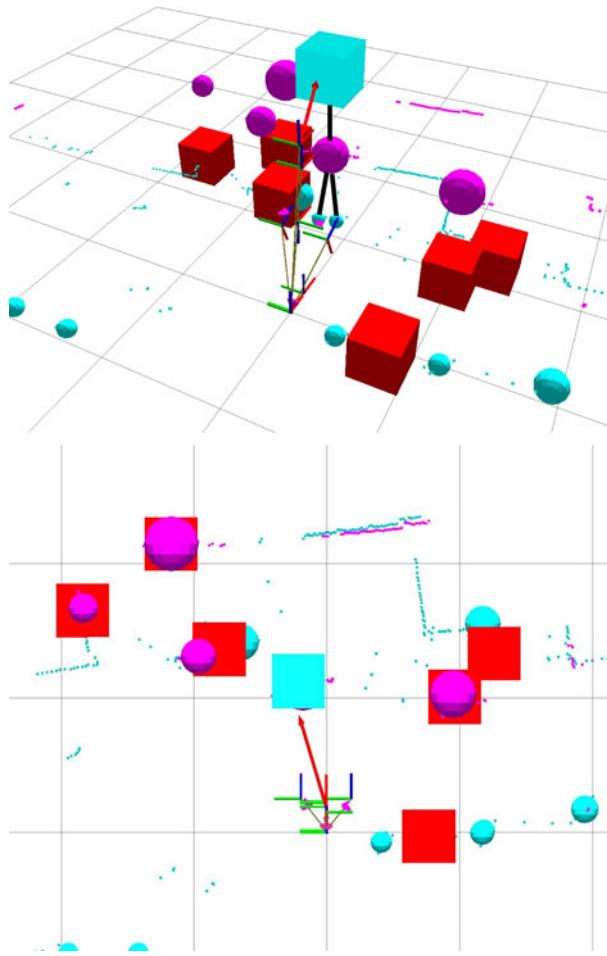


Fig. 1. Persons are detected as legs (cyan spheres) and torsos (magenta spheres) in two laser range scans (cyan and magenta dots). The detections are fused in a multi-hypothesis tracker (red and cyan boxes). Faces are detected with a camera mounted on a pan-tilt unit. We validate tracks as persons (cyan box) when they are closest to the robot and match the line-of-sight towards the face (red arrow). From the projection of the track position onto the face direction we also determine the face height.

We fuse person detections from both LRFs in a multi-hypothesis tracker. We estimate position and velocity of each hypothesis by Kalman filters (KFs). In the KF prediction step, we use odometry information to compensate for the motion of the robot. The tracks are corrected with the observations of legs and torsos.

We use the Hungarian method [11] to associate each torso detection in a scan uniquely with existing hypotheses. In contrast, as both legs of a person may be

detected in a scan, multiple leg detections may be assigned to a person hypothesis. Only unassociated torso detections are used to initialize new hypotheses. Spurious tracks with low detection rates are removed.

Of course, the shape of legs and torsos in laser range scans are not discriminative enough, such that parts of the environment cause false positive detections. For this reason, we verify that tracks correspond to persons through a second sensor modality that provides complementary information: We detect frontal and profile views of faces in camera images using the Viola and Jones [12] algorithm. Additionally, we detect upper bodies with a method based on Histograms of Oriented Gradients [13].

The camera is mounted on a pan-tilt unit in a height of approx. 1.5 m. Since the LRFs have a larger field-of-view than the cameras, we implemented an active gaze strategy. We find the correspondence of detected faces and upper bodies with possible person tracks by determining the matching track on the line-of-sight of the detected face that is closest to the robot. We also measure the height of the detected face by projecting the track position onto the face direction. We test that the face has a reasonable height in a later processing stage. Fig. 1 illustrates our approach to continuous people awareness (CPA) with an example.

4 Utilizing Semantic Scene Knowledge for Estimating People Presence

Approaches to person tracking are prone to false detections. Low false positive rates in person detection can hardly be achieved, even with more powerful vision-based approaches to person recognition. Thus, we propose to use prior semantic scene knowledge to further increase the robustness of people awareness.

In our approach, we estimate the probability of the presence of people. Similar to the well-known occupancy grid mapping approach to 2D environment mapping, we estimate person presence in a two-dimensional discretized representation of the environment. From semantic annotations of the environment, e.g., walkable floor, chairs, and shelves, we infer prior information on the presence of people. The semantics also provides us with the valid height range of faces. The prior knowledge and the estimated presence likelihood helps us to decide when tracks are falsely detected as persons.

4.1 Estimating People Presence

Our CPA provides the position of persons in the robot's vicinity. Where no person is detected, LRFs measure distance and bearing to the environment structure and thus provide negative information on the presence of people along the measurement. From these measurements and the pose of the robot, we estimate the presence likelihood of people in the environment.

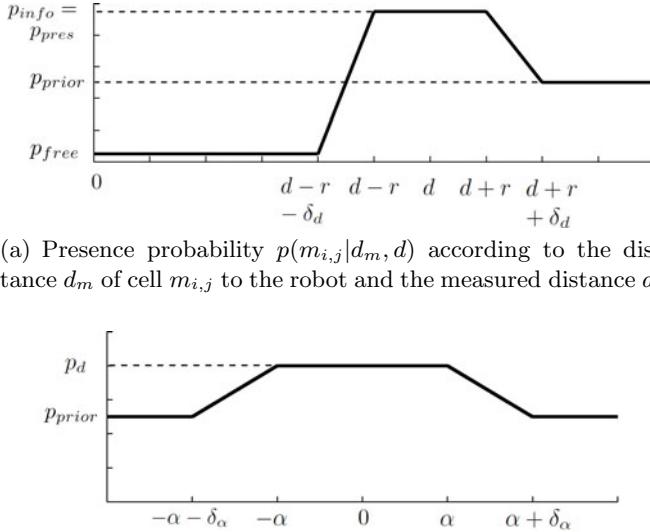


Fig. 2. Components of the inverse measurement model

We discretize the environment into a 2D grid. For each cell $(i, j) \in \mathbb{N} \times \mathbb{N}$ in the grid, we estimate the belief of the presence of a person at time t

$$bel_t(m_{i,j}) := p(m_{i,j}|s_{1:t}, z_{1:t})$$

from the poses $s_{1:t}$ of the robot and the positive and negative measurements $z_{1:t}$ up to time t . We assume independence between individual cells, which enables us to estimate the posterior $p(m|s_{1:t}, z_{1:t})$ over the complete map by estimating presence for each cell individually.

Following basically the same derivation as for occupancy grid mapping [14], we arrive at the recursive update scheme in log-odds form:

$$l_t(m_{i,j}) = l_{t-1}(m_{i,j}) + \log \left(\frac{p(m_{i,j}|s_t, z_t)}{1 - p(m_{i,j}|s_t, z_t)} \right) - \log \left(\frac{p(m_{i,j})}{1 - p(m_{i,j})} \right),$$

where we define

$$l_t(m_{i,j}) := \log \left(\frac{p(m_{i,j}|s_{1:t}, z_{1:t})}{1 - p(m_{i,j}|s_{1:t}, z_{1:t})} \right).$$

Here, $p(m_{i,j}|s_t, z_t)$ is the inverse sensor model and $p(m_{i,j})$ is the prior presence probability of cell (i, j) .

The inverse measurement model is composed of two parts: The first component $p(m_{i,j}|d_m, d)$ determines the presence probability depending on the distance d_m of cell $m_{i,j}$ to the robot and the measured distance d (cf. Fig. 2(a)). We model

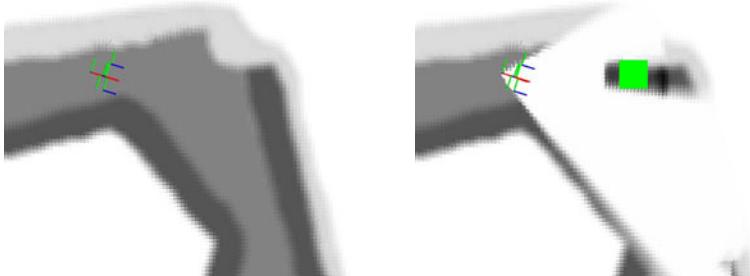


Fig. 3. A person is detected and included into the knowledge base (green square). We update the presence belief (black for probability 1) with positive measurements of the detected person and negative measurements of the environment structure.

high likelihood $p_{info} = p_{present}$, when the cell lies within the influence region of a person, which we denote size radius r . For measurements of the environment structure, the presence probability $p_{info} = p_{free}$ is small within an influence radius r . For cells behind the distance $d_m \geq d + r + \delta_d$ we cannot measure, and thus we model prior probability p_{prior} . Cells in front of the robot and with a distance $0 \leq d_m \leq d - r - \delta_d$ are not occupied by persons. Distance δ_d is a small interpolation range to model uncertainty in the measured distance.

The second part $p(m_{i,j}|\phi_m)$ models influence width and uncertainty along the orthogonal direction to the line of sight. We consider the angular deviation ϕ_m between the cell and the measured position (cf. Fig. 2(b)). In this model, the angular range α is determined from the width of the person or the influence range of a beam. Within this range, the presence likelihood is determined by the first component $p_d := p(m_{i,j}|d_m, d)$. Beyond the angular deviation $|\phi_m| \geq \alpha + \delta_\alpha$ the measurement bears no information for the cell and thus we model prior probability p_{prior} . Again, the interpolation range δ_α models uncertainty in the measurement.

Finally, we consider dynamic changes in the environment by unlearning the acquired information about the presence of people. To implement this exactly, one would require to store a temporal history of measurement updates to each grid cell. As this is not feasible, we approximate unlearning with a small temporal decay towards the prior cell probability.

4.2 Maintaining Person Knowledge

When new persons are found, we represent their size, location, and face height in a knowledge base. Then, the presence map is updated from positive and negative measurements. Fig. 3 illustrates this process in an exemplary situation.

While persons are tracked, their information is continuously corrected in the person knowledge base. Simultaneously, the presence map is updated with positive measurements of the tracked persons. We remove persons from the knowledge base, if the presence belief at their location drops below some threshold.

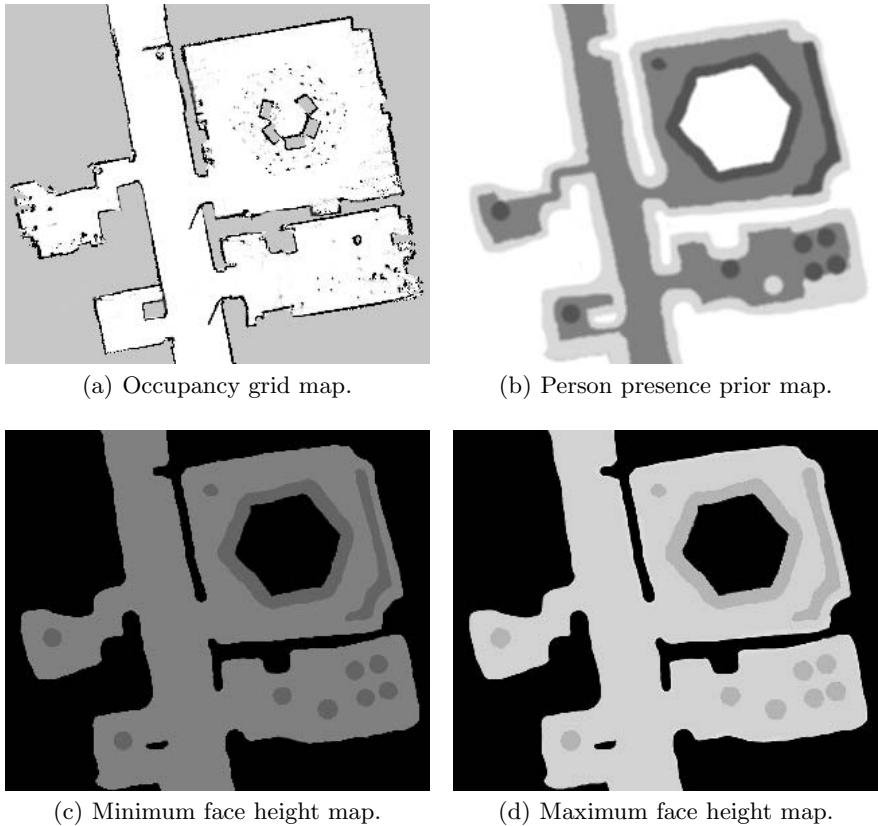


Fig. 4. Semantic prior maps (b)-(c) for the environment in (a) Darker is lower / more likely

4.3 Prior Information from Semantic Scene Knowledge

For high-level control of autonomous robots, the semantics of the environment is an important source of information. In the context of people awareness, we propose to exploit the semantics to support the interpretation and disambiguation of sensory data for robust person detection and representation. For example, a chair gives us a hint that persons are more likely to be present at this location than on walkable floor. There, a person typically occupies space for only short durations. In addition, we can use information on the possible postures of people to constrain the detectable heights of faces. Furthermore, we can surely exclude that people reside in walls or furniture like shelves.

We incorporate semantic scene knowledge into our approach in the following way: From semantics, we infer the prior probability of person presence. Additionally, we deduce minimum and maximum face heights throughout the map. We represent this information in 2D grid maps, which enables us to efficiently use the information during presence belief and person knowledge base updates.

Fig. 4 shows examples for grid maps of occupancy used for navigation, person presence prior, and minimum and maximum face height.

4.4 Improving People Awareness by Semantic Priors

With semantic priors we can improve the robustness of our people awareness approach. The prior probability on person presence can be directly incorporated into our method for person presence estimation. We use the person presence prior to initialize the posterior estimate. Furthermore, applying Bayes rule to the inverse measurement model, we see that

$$p(m_{i,j}|s_t, z_t) = \eta p(z_t|s_t, m_{i,j}) p(m_{i,j}).$$

We assume an uninformed prior probability of $p(m_{i,j}) = \frac{1}{2}$ in our previous inverse measurement model and seek to obtain the semantic inverse measurement model by replacing it with the semantic prior $p_{sem}(m_{i,j})$. Unfortunately, the prior is also contained in the normalization factor $\eta = \sum_m p(z_t|s_t, m) p(m)$. However, by substituting $p(z_t|s_t, m_{i,j})$ with $\frac{2}{\eta} p(m_{i,j}|s_t, z_t)$ in

$$\begin{aligned} p_{sem}(m_{i,j}|s_t, z_t) &= \eta' p(z_t|s_t, m_{i,j}) p_{sem}(m_{i,j}), \text{ and} \\ p_{sem}(\neg m_{i,j}|s_t, z_t) &= \eta' p(z_t|s_t, \neg m_{i,j}) p_{sem}(\neg m_{i,j}), \end{aligned}$$

and exploiting the fact that $p_{sem}(m_{i,j}|s_t, z_t)$ is a probability measure over a binary variable, we find

$$p_{sem}(m_{i,j}|s_t, z_t) = \frac{p(m_{i,j}|s_t, z_t) p_{sem}(m_{i,j})}{p(m_{i,j}|s_t, z_t) p_{sem}(m_{i,j}) + (1 - p(m_{i,j}|s_t, z_t)) (1 - p_{sem}(m_{i,j}))}.$$

The prior strongly biases belief estimation. At locations with very low prior probability, spurious person detections increase the person presence belief insignificantly. On the other hand, detections in a-priori highly probable areas will faster converge to high presence probability.

We further exploit semantic priors to reject falsely detected persons in regions with low presence probability. Low prior probability in such regions supports the robustness against false detections, as the posterior belief is hardly increased from spurious person detections. Finally, we validate the face height of detections using semantic priors.

5 Exploiting Semantics for Efficient People Search

So far, our approach robustly tracks multiple persons in the robot's surrounding. It estimates a belief of person presence in the viewed environment and also maintains knowledge of found people. Typically, a mobile service robot operates in an environment which it cannot survey from a single location. When the robot

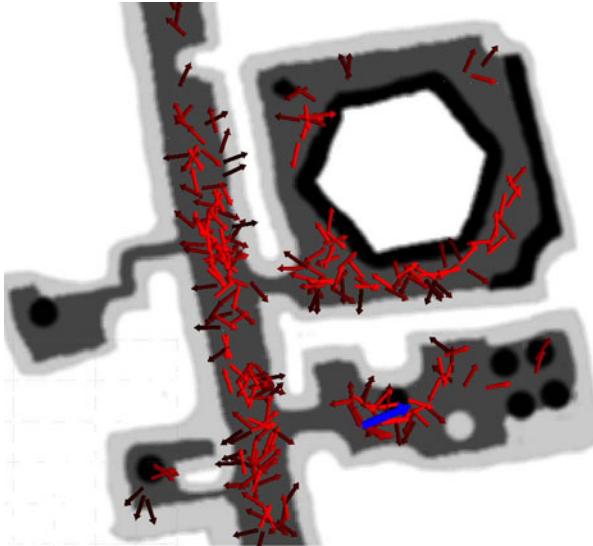


Fig. 5. For efficient exploration, we randomly draw $N = 400$ poses from a normal distribution around the current pose. Among these poses (small red arrows) we select a new exploration pose (big blue arrow) that maximizes the expected detection rate (increasing from dark to bright).

further requires awareness of people in the complete environment, an efficient exploration strategy needs to be devised.

For this purpose, we propose an exploration strategy that utilizes semantic prior knowledge. Based on the presence belief of people, the robot selects promising exploration views in which the rate of detecting new persons is maximized.

We sample N random poses from a normal distribution around the current pose of the robot. From these poses we select the one that achieves best expected detection rate in the viewed area. We define the expected detection rate $\mathbf{E}_{\mathcal{M}}(D)$ in a map region \mathcal{M} as

$$\mathbf{E}_{\mathcal{M}}(D) = \sum_{m_{i,j} \in \mathcal{M}} (p(D|m_{i,j}) \text{ } bel(m_{i,j}) + p(D|\neg m_{i,j}) \text{ } bel(\neg m_{i,j})) ,$$

where $bel(m_{i,j})$ is the current belief estimate for the presence of a person in cell (i, j) and $p(D|m_{i,j})$ is the detection probability given the presence of a person in a cell.

We determine the viewed area \mathcal{M} by ray-casting in the environment map. To prevent search at places where persons have already been found, we exclude regions that are occupied by known persons in the knowledge base. In this way, we measure the expected detection rate of new persons. Combined with our semantic prior, the robot prefers to explore regions with high expected detection rate of new persons.

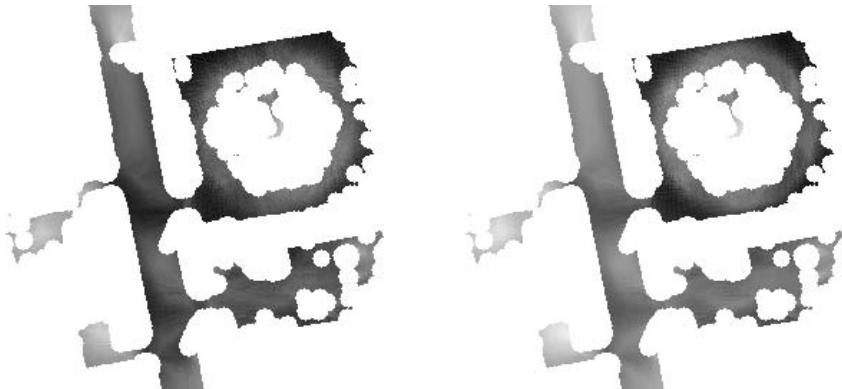


Fig. 6. Expected detection rate in discretely samples poses without (left) and with (right) semantic prior. For each position the best orientation is shown. With semantic prior the expected detection rate is high for poses that view regions with predominantly high presence likelihood. Without prior, the number of cells under the pose’s view solely determines the detection rate measure.

6 Experiments

We first present exemplary results for our exploration strategy. Fig. 6 visualizes the effect of the semantic prior on the expected detection rate. We sample poses at equidistant positions with a spacing of 5 cm and in $\frac{\pi}{8}$ orientation steps. At each location we determine the maximum value over orientations and normalize all values to the interval [0, 1]. Without prior, the number of cells under the pose’s view solely determines the detection rate measure. When we add the semantic prior, the expected detection rate is high for poses that view regions with predominantly high presence likelihood. By utilizing semantic information, our exploration approach concentrates on those regions with high detection rate of new persons.

We also evaluate our exploration approach with our domestic service robot that competes in the RoboCup@Home league. We conduct an experiment in our test environment (cf. Fig. 7). Three persons are placed at the locations depicted in the upper left of Fig. 7. Person presence prior and face height maps have been manually designed to represent the semantics of the environment. A timeline of the events in the experiment is given on the bottom of Fig. 7. The robot drives to 16 exploration views. It finds all three persons in ca. 909 seconds. Three false positive detections are rejected: the first one due to an invalid face height of approx. 0.55 m, the others as they are located at places with low presence estimate.

We successfully applied continuous people awareness during the RoboCup GermanOpen 2010 in Magdeburg. In the Who-Is-Who test, the robot detected and identified all 5 persons that either sat or stood in an apartment-like environment.

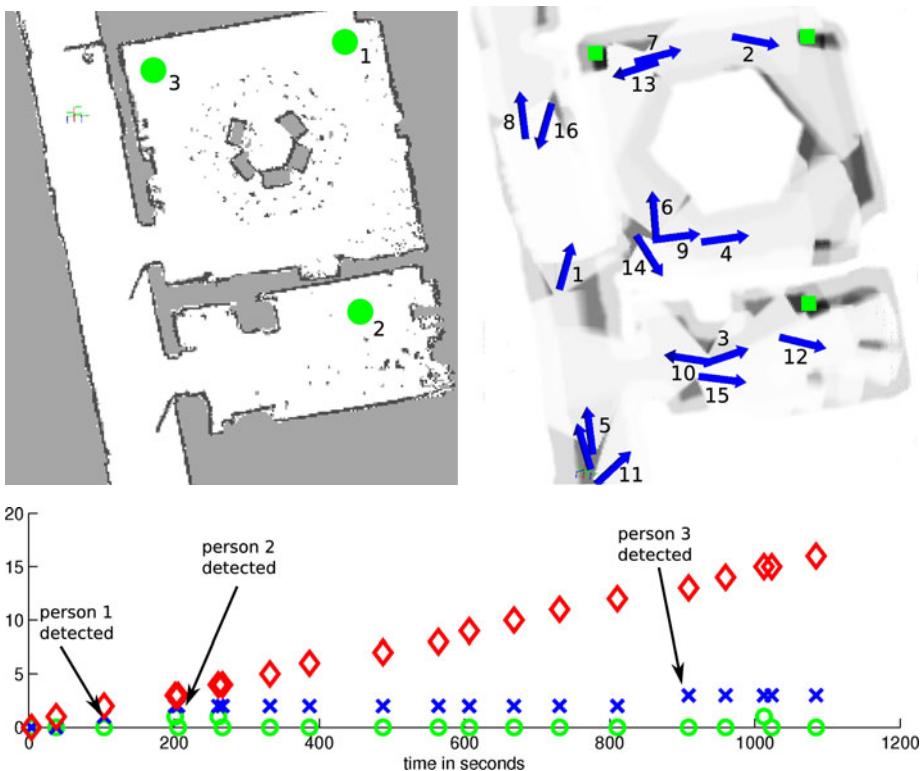


Fig. 7. Top left: Environment of the exploration experiment with three persons (green circles). Top right: Final person presence belief and locations in the knowledge base (green boxes). The blue arrows indicate exploration poses. Bottom: Event timeline of the experiment showing visit time of exploration poses (red diamonds), number of detected persons (blue crosses), and number of represented false detections in the person knowledge base (green circles).

7 Conclusions

We propose an approach to people awareness for mobile service robots that exploits knowledge about the semantics of the environment. From known semantics, we model the prior probability of the presence of persons and represent it in a 2D grid. We estimate the presence probability of persons recursively and incorporate the semantic prior into the estimation. Furthermore, we extract valid face height ranges from semantics.

During person search, we reject false detections in unlikely regions and with invalid face heights. We also propose a search strategy for people that maximizes the expected detection rate in the robot's view. We demonstrate our approach with our domestic service robot that competes in the RoboCup@Home league.

Currently, our approach is limited by the person detection methods we employ. The use of more generic person detection schemes could further improve robustness and detection rate.

In the current system the user provides semantic annotations and interpretations in the form of priors and face height ranges. It is an interesting line of research to recognize objects and places, and to learn semantic prior models by observation of people at such spatial entities.

References

1. Schulz, D., Burgard, W., Fox, D., Cremers, A.B.: People tracking with a mobile robot using sample-based joint probabilistic data association filters. *International Journal of Robotics Research* 22 (2003)
2. Arras, K., Grzonka, S., Luber, M., Burgard, W.: Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, USA (2008)
3. Reid, D.B.: An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control* 24(6) (1979)
4. Berclaz, J., Fleuret, F., Fua, P.: Robust People Tracking with Global Trajectory Optimization. In: *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 744–750 (2006)
5. Lanz, O.: Approximate bayesian multibody tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* 28(9), 1436–1449 (2006)
6. Gavrila, D.M., Munder, S.: Multi-cue pedestrian detection and tracking from a moving vehicle. *Int. Journal of Computer Vision* 73(1), 41–59 (2007)
7. Ess, A., Leibe, B., Schindler, K., van Gool, L.: A mobile vision system for robust multi-person tracking. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2008)
8. Okuma, K., Taleghani, A., de Freitas, N., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: Pajdla, T., Matas, J(G.) (eds.) *ECCV 2004. LNCS*, vol. 3021, pp. 28–39. Springer, Heidelberg (2004)
9. Wu, B., Nevatia, R.: Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision* 75(2), 247–266 (2007)
10. Luber, M., Tipaldi, G.D., Arras, K.O.: Spatially grounded multi-hypothesis tracking of people. In: *Proceedings of the ICRA Workshop on People Detection and Tracking* (2009)
11. Kuhn, H.W.: The hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2(1), 83–97 (1955)
12. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2001)
13. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *International Conference on Computer Vision & Pattern Recognition*, vol. 2, pp. 886–893 (June 2005)
14. Thrun, S., Burgard, W., Fox, D.: *Probabilistic Robotics*. MIT Press, Cambridge (2005)

An Evaluation of Open Source SURF Implementations

David Gossow, Peter Decker, and Dietrich Paulus

Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
dgossow@uni-koblenz.de
<http://robots.uni-koblenz.de>

Abstract. SURF (Speeded Up Robust Features) is a detector and descriptor of local scale- and rotation-invariant image features. By using integral images for image convolutions it is faster to compute than other state-of-the-art algorithms, yet produces comparable or even better results by means of repeatability, distinctiveness and robustness. A library implementing SURF is provided by the authors. However, it is closed-source and thus not suited as a basis for further research.

Several open source implementations of the algorithm exist, yet it is unclear how well they realize the original algorithm. We have evaluated different SURF implementations written in C++ and compared the results to the original implementation.

We have found that some implementations produce up to 33% lower repeatability and up to 44% lower maximum recall than the original implementation, while the implementation provided with the software Pan-o-matic produced almost identical results.

We have extended the Pan-o-matic implementation to use multi-threading, resulting in an up to 5.1 times faster computation on an 8-core machine. We describe our comparison criteria and our ideas that lead to the speed-up. Our software is put into the public domain.

1 Introduction

Many problems of computer vision can be solved by finding point correspondences between images using local features. Examples are object recognition, depth reconstruction and self localization. The first step in computing local features consists of *detecting* salient locations such as corners, blobs and t-junctions. From the neighbourhood regions of these *interest points*, image features are then calculated, yielding a *descriptor* for each one. Corresponding points between two images can then be found by comparing these descriptors.

SURF (Speeded Up Robust Features) [BTVG06, BETvG08] aims at being invariant to weakly affine transformations (scaling and in-plane rotation) and homogeneous changes in intensity, robust to other typical geometric and photometric transformations and fast to compute. It detects blob structures of arbitrary size in grayscale images by searching maxima of the determinant of the

Hessian matrix over the spacial dimensions as well as a wide range of scales. The dominant orientation of gradients in the blob's neighbourhood, together with its scale, are used to construct a normalized descriptor window. From this window, a feature vector containing gradient information is calculated.

SURF makes use of box filters instead of Gaussian kernels to construct the scale space representation. These are a combination of filters calculating the mean value of a rectangular image region in constant time. This is achieved by using integral images as data structure.

Its invariance and run time properties make SURF particularly suited for applications in highly uncontrolled environments where computation time is a critical issue, such as in the field of autonomous robotic systems.

Several evaluations of different local feature detectors and descriptors have been made [MS05, MTS⁺05, MP07]. However, in practice the actual implementation of an algorithm can also have a large influence on its performance.

In this paper, we evaluate the performance of different implementations of the SURF algorithm. We focus on open source implementations of SURF, as they are best suited as a basis for further research and improvements. A list of the implementations can be found in section 2.

To assure the comparability of our results, we rely on an established evaluation framework already used in several other publications [BTVG06, BETvG08, MS05, MTS⁺05]. It consists of image sequences with different transformations and tools for calculating performance values, as described in section 3. The results show significant differences between the implementations, as described in section 4.

In section 4.3, we take a closer look on the implementations which produced the best results in our evaluation. We show that the use of linear interpolation between the bins of the descriptor window significantly increases its performance. In section 5 we describe how to parallelize parts of the algorithm, speeding up its computation on multi-core machines.

Section 6 gives hints to related publications. A resume of the contributions made by this paper is given in section 7.

2 The Contestants

All tested implementations are available as open source software, except for the library released by the authors of SURF¹.

- *OpenSURF*² is a dedicated library implementing the SURF algorithm. We have compared two different releases of it. The first one, released on 22 March 2009, implements the algorithm as described in the original SURF publication [BETvG08, Low04]. The second release of 31 August 2009 contains a modified algorithm for calculating the descriptor as described in [AKB08].

¹ <http://www.vision.ee.ethz.ch/~surf/>

² <http://code.google.com/p/opensurf1/>

- *OpenCV*³ and *dlib* 17.20⁴ are image processing libraries which contain implementations of SURF. We used the OpenCV repository version of 22 April 2010 for our evaluations. It supports multithreading using OpenMP⁵.
- *Pan-o-matic* 0.9.4⁶ is a tool for computing point correspondences between images for use in panorama creation tools, which includes an implementation of SURF.
- The library *libmv*⁷ contains an implementation of SURF. In the current version (April 2010), it does not support rotation invariance and was not included in the evaluation.
- *P-SURF*⁸ is a modified version of OpenSURF using multi-threading as described in [Zha10]. According to the author, it is based on the March 2009 release of OpenSURF or an earlier one and contains only minor modifications besides the multi-threading support. Due to time reasons, it was not included in the evaluation.

3 Evaluation Framework

Our evaluation is based on Mikolajczyk’s software framework and image sequences⁹. It was also used in other evaluations [MS05, MTS⁺05] and in the original SURF publications [BTVG06, BETvG08].

The parameters of all implementations were set to the same values. The initial sampling step was set to 1 pixel and the number of searched octaves to 4. No image doubling was used. The descriptor was configured to yield a rotation-invariant 64-dimensional feature vector.

3.1 Image Sequences

The sequences used in this evaluation each contain 6 images of natural textured scenes with increasing geometric and photometric transformations. They are either planar or the camera remained at a fixed position during the acquisition, so the images of one sequence are related by a homography (plane projective transformation). The homography matrices are provided with the image data. They were calculated by manually defining point correspondences between the images and refining the resulting homography using a robust small-baseline homography estimation algorithm.

- The *Bark* and *Boat* sequences (Figures 4a and 4b) contain images of two scenes with increasing zoom and different rotations of the image.

³ <http://sourceforge.net/projects/opencvlibrary/>

⁴ <http://dclib.sourceforge.net/>

⁵ <http://openmp.org>

⁶ <http://aorlinsk2.free.fr/panomatic/>

⁷ <http://code.google.com/p/libmv/>

⁸ <http://www.xjtlu.edu.cn/depts/csse/csse1511>

⁹ <http://www.robots.ox.ac.uk/~vgg/research/affine/>

- The *Graffiti* and *Bricks* sequences (Figures 4c and 4d) contain images of planar scenes. The viewpoint changes up to approximately 60° .
- In the *Bikes* and *Trees* sequences (Figures 4e and 4f), the focus of the camera was altered to create an increasing effect of image blur.
- In the *Cars* sequence (Figure 4g), the aperture of the camera was altered, varying from under- to over-exposition.
- The *UBC* sequence (Figure 4h) contains several versions of the same image with an increasing degree of JPEG compression artefacts.

3.2 Performance Criteria

The performance criteria are *repeatability* for the detection step and *precision* and *recall* for the descriptor.

The *repeatability* of a detector is defined by the percentage of interest points detected again under a given transformation of the image. To calculate it, the regions representing the interest points are mapped from one image to the other using the homography. It is then checked how many of them overlap with a region from the other image by a given minimal percentage. Only interest points are taken into account that are visible in both images (see Figure 1).

In the case of SURF, each interest point with scale σ is assigned a circular region with radius 10σ . If the overlap ratio of two regions A and B is above a threshold ϵ_0 , then the interest points represented by them are considered as corresponding. The overlap ratio of two regions is determined by the ratio of their intersection and union (see Figure 2):

$$\frac{A \cap B}{A \cup B} \geq \epsilon_0$$

If the images contain n_1 and n_2 relevant interest points, then the repeatability is defined as

$$\text{repeatability} = \frac{\#\text{correspondences}}{\min(n_1, n_2)}$$

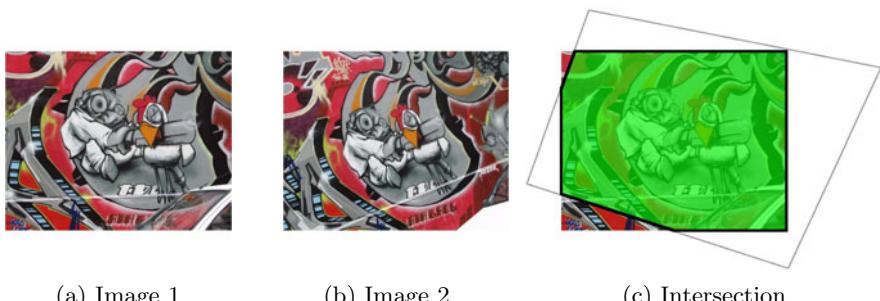


Fig. 1. (a) and (b) show the first two images of the *Graffiti* sequence. (c) shows the region visible in both images (green) determined by the homography (gray).

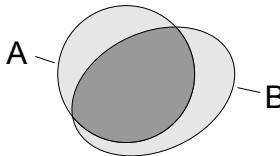


Fig. 2. Two overlapping regions. One region is circular, while the other one was mapped from another image by a homography and is thus distorted.

SURF uses a threshold on the determinant of the Hessian matrix to filter out weakly localized interest points. This threshold was set to zero for all implementations, thus effectively disabling it. The 500 interest points having the strongest determinant of Hessian response were then selected for each image and implementation and used for the evaluation.

The measures we use for descriptor performance are *precision* and *recall*. The precision is represented by the number of correct matches relative to the total number of found matches:

$$\text{precision} = \frac{\#\text{correct matches}}{\#\text{matches}}$$

Recall is the number of correct matches relative to the total number of corresponding interest regions:

$$\text{recall} = \frac{\#\text{correct matches}}{\#\text{correspondences}}$$

To calculate precision and recall, each interest point in one image is matched to the closest neighbour in the other image based on the euclidean distance of their feature vectors. Matches are rejected if their *distance ratio* [Low04] is above a threshold t_ϕ . It is defined as the ratio of the distances to the closest neighbour and to the second-closest neighbour. By varying t_ϕ , a sequence of precision-recall pairs is obtained which allows the comparison of different descriptors.

4 Results

4.1 Detector Comparison

Figures 5 and 6 show the repeatability of all implementations under various image transformations.

The implementation included in Pan-o-matic produces nearly identical results to the original SURF library. In the *Bark*, *Graffiti*, *Bricks*, *Trees* and *UBC* sequences, they produce the highest repeatability, in the others OpenCV shows a higher performance. In the *Bark* sequence, OpenCV does not detect any interest points in the region of interest of the third and all following images.

4.2 Descriptor Comparison

Figure 7a shows the evaluation results of the different descriptor implementations. The Pan-o-matic implementation produces the same results as the original SURF library, followed by the August release of OpenSURF, OpenCV and the March release of OpenSURF. The Dlib implementation performs significantly worse in comparison to the others.

4.3 Effect of Linear Interpolation

In contrast to the description given in the original SURF publication [BTVG06, BETvG08], Pan-o-matic as well as the second release of OpenSURF interpolate between the bins of the descriptor window. Pan-o-matic uses linear interpolation, while OpenSURF uses Gaussian functions centred at each bin as described in [AKB08].

To measure the effect of interpolation, we removed the interpolation step from Pan-o-matic. Figure 7b shows that the recall drops by approximately 7 percentage points if no interpolation is used.

By analysing the source code of Pan-o-matic, we have found that several aspects of it differ from the description given in [BTVG06, BETvG08]. However, as the source code is publicly available, further study of these details will be left to the reader.

5 A Multi-threaded Implementation of SURF

We extended the implementation included in Pan-o-matic so it performs parts of the calculations in parallel by employing the thread pool pattern. A thread pool contains a fixed number of threads. An arbitrary number of tasks can be scheduled for execution. Each thread consecutively removes tasks from the queue and executes them. We used the library *threadpool*¹⁰, which relies on the *Boost threads*¹¹ library.

In the detection step, the scales of each octave are built and searched in a single task, resulting in a maximal parallelism of 5 threads of execution. In the description process, the orientation and feature vector of each interest point is computed in a single task. Thus, the maximal parallelism in this step is limited by the number of interest points.

The source code is made publicly available¹² under the GPL license.

Figure 3 shows the mean computation time of the multi-threaded implementation for the images used in the previous evaluation. The red bars show the hypothetical upper limit of the speed increase, which is a linear scaling depending on the number of cores used. The experiment was performed on an 8-core 3.0 GHz Intel Xeon machine with 16 GB RAM. The program was compiled using *gcc* with compiler optimizations turned on. The parameters for the SURF

¹⁰ <http://threadpool.sourceforge.net/>

¹¹ <http://www.boost.org/>

¹² <http://parallelsurf.sourceforge.net/>

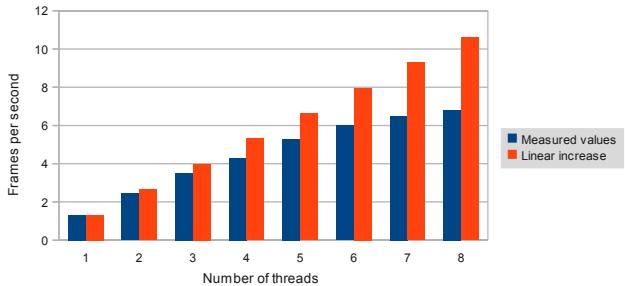


Fig. 3. Mean calculation speed when using different numbers of threads

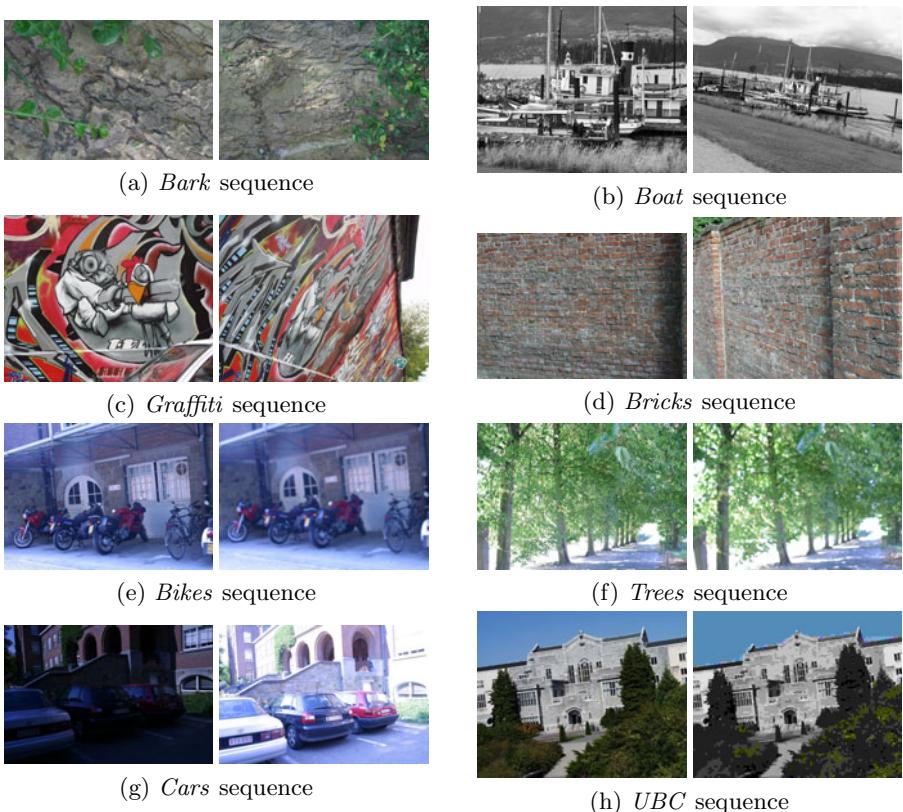


Fig. 4. Examples from the image data set. Shown are the first and last image of each sequence.

algorithm were set to the default values of the original implementation, except for the initial sampling step, which was set to 1. 6625 interest points were found in the images in average.

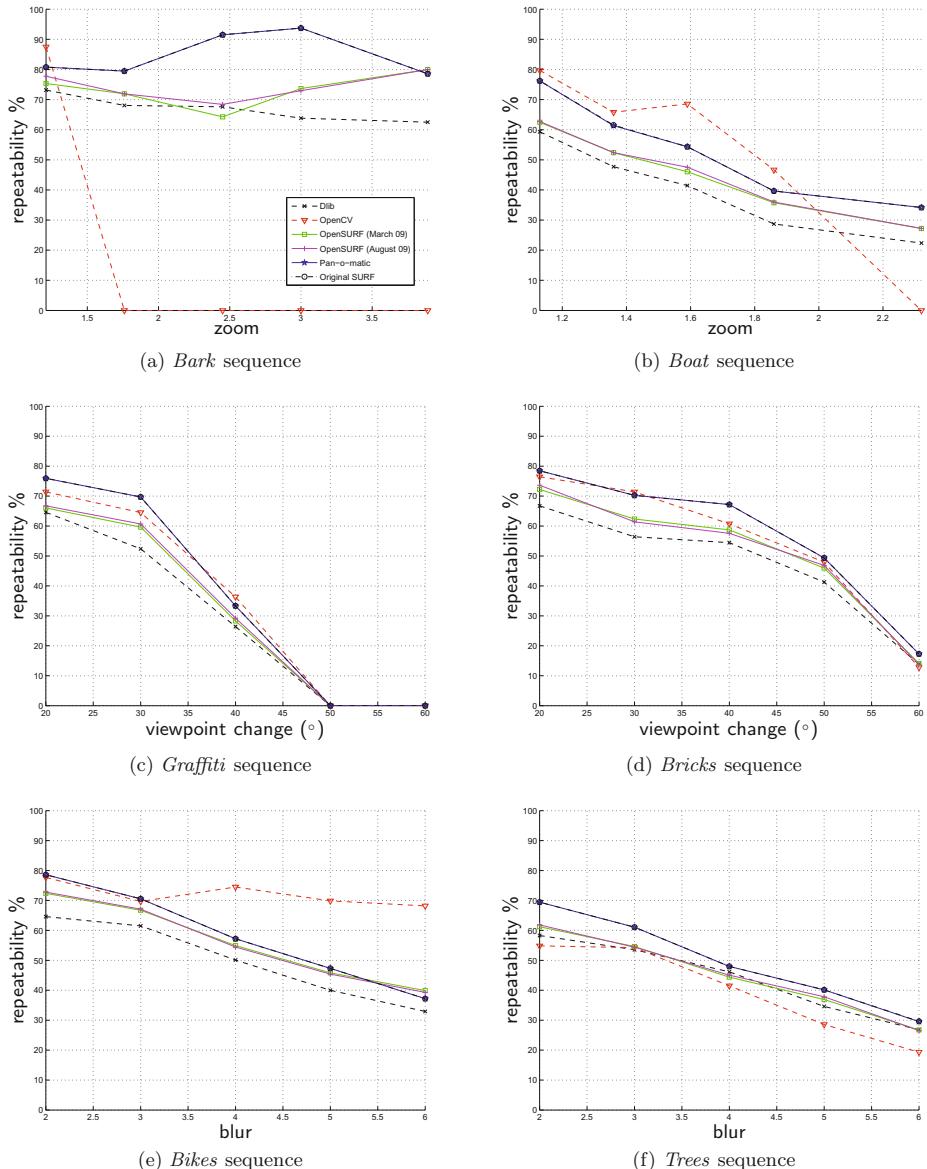


Fig. 5. Repeatability of implementations under various image transformations (pt. 1). Note that the graphs of SURF and Panomatic are identical.

The algorithm runs 1.8 times faster using 2 threads, 3.3 times faster using 4 threads and 5.1 times faster using 8 threads. The single-threaded version runs 3.9 times faster than the original binary SURF library.

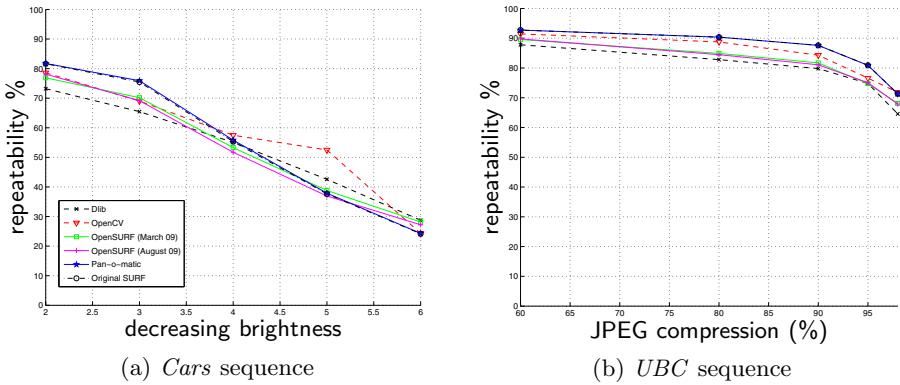


Fig. 6. Repeatability of implementations under various image transformations (pt. 2)

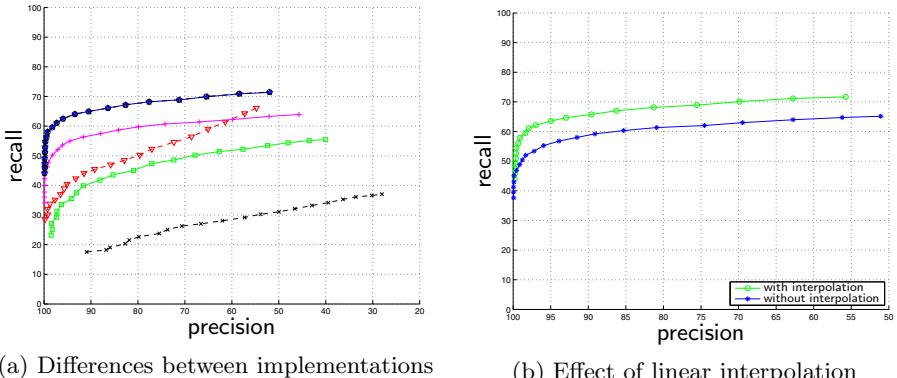


Fig. 7. Evaluation results for descriptor performance

6 Further Reading

The SURF algorithm was introduced in [BTVG06] and described in more detail in [BETvG08]. The authors use the same framework as this publication [MS05, MTS⁺05] to evaluate the algorithm. They show that the detector outperforms SIFT [Low04] as well as Harris- and Hessian-Laplace [MS04] and that the descriptor outperforms SIFT and GLOH [MS05].

The reader should be aware that, although the evaluation framework used in this paper includes a variety of typical image transformations, it focuses mainly on the problem of image registration. The evaluation criteria for selecting an adequate algorithm or, in this case, implementation, are always task-dependent. A discussion of the motivation and validity of evaluations such as the one conducted in this paper is given in [Har94, För96].

Other evaluations have been performed that put a focus on other applications such as the recognition of three-dimensional, non-planar objects based on local grayscale [MP07] or color features [BG09]. In [BG09] and [vdSGS08], the performance of local features for object category recognition is evaluated.

7 Conclusion

In this paper, we have evaluated several open source implementations of the SURF algorithm for detecting and describing local grayscale image features. We have presented an evaluation framework which was previously used in other evaluations. The implementation included in the software Pan-o-matic produces the best results, which are at the same time identical to those of the original implementation.

We have pointed out that the best-performing implementations use interpolation in the descriptor step, as opposed to the description given in the original publications [BTVG06, BETvG08]. We have showed that removing the interpolation from the Pan-o-matic implementation results in lower recall values. We have extended the Pan-o-matic implementation to use multi-threading, resulting in an up to 5.1 times faster calculation.

References

- [AKB08] Agrawal, M., Konolige, K., Blas, M.R.: CenSurE: Center surround extrema for realtime feature detection and matching. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) *ECCV 2008, Part IV*. LNCS, vol. 5305, pp. 102–115. Springer, Heidelberg (2008)
- [BETvG08] Bay, H., Ess, A., Tuytelaars, T., van Gool, L.: Speeded-up robust features (surf). *Journal of Computer Vision* 110(3), 346–359 (2008)
- [BG09] Burghouts, G.J., Geusebroek, J.-M.: Performance evaluation of local colour invariants. *Comput. Vis. Image Underst.* 113(1), 48–62 (2009)
- [BTVG06] Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006*. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
- [Für96] Förstner, W.: 10 pros and cons against performance characterization of vision algorithms Technical report, Institut für Photogrammetrie, Universität Bonn (1996)
- [Har94] Haralick, R.M.: Performance characterization in computer vision. *Computer Vision Graphics and Image Processing* 60(2), 245–249 (1994)
- [Lowe04] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2), 91–110 (2004)
- [MP07] Moreels, P., Perona, P.: Evaluation of features detectors and descriptors based on 3d objects. *International Journal of Computer Vision* 73(3), 263–284 (2007)
- [MS04] Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. *International Journal of Computer Vision* 60(1), 63–86 (2004)
- [MS05] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(10), 1615–1630 (2005)

- [MTS⁺05] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. *International Journal of Computer Vision* 65(1-2), 43–72 (2005)
- [vdSGS08] van de Sande, K.E.A., Gevers, T., Snoek, C.G.M.: Evaluation of color descriptors for object and scene recognition. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 0, pp. 1–8 (2008)
- [Zha10] Zhang, N.: Computing optimised parallel speeded-up robust features (p-surf) on multi-core processors. *International Journal of Parallel Programming* (2010)

A Semantic World Model for Urban Search and Rescue Based on Heterogeneous Sensors

Johannes Meyer², Paul Schnitzspan¹, Stefan Kohlbrecher¹, Karen Petersen¹, Mykhaylo Andriluka¹, Oliver Schwahn¹, Uwe Klingauf², Stefan Roth¹, Bernt Schiele^{1,3}, and Oskar von Stryk¹

¹ Department of Computer Science, TU Darmstadt, Germany

² Department of Mechanical Engineering, TU Darmstadt, Germany

³ MPI Informatics, Saarbrücken, Germany

Abstract. In urban search and rescue scenarios, typical applications of robots include autonomous exploration of possibly dangerous sites, and the recognition of victims and other objects of interest. In complex scenarios, relying on only one type of sensor is often misleading, while using complementary sensors frequently helps improving the performance. To that end, we propose a probabilistic world model that leverages information from heterogeneous sensors and integrates semantic attributes. This method of reasoning about complementary information is shown to be advantageous, yielding increased reliability compared to considering all sensors separately. We report results from several experiments with a wheeled USAR robot in a complex indoor scenario. The robot is able to learn an accurate map, and to detect real persons and signs of hazardous materials based on inertial sensing, odometry, a laser range finder, visual detection, and thermal imaging. The results show that combining heterogeneous sensor information increases the detection performance, and that semantic attributes can be successfully integrated into the world model.

1 Introduction

Modeling the world in complex environments is a crucial aspect on the way toward reliable, intelligent, and autonomous search and rescue robots. As motivated in [1,2,3], it is desirable not only to infer a geometrically interpretable map, but also to integrate semantic attributes to enable high-level scene interpretation. In urban search and rescue (USAR), reliable robots have to provide a semantically meaningful interpretation of objects within a scene (e.g. victims in collapsed buildings) [4]. In unconstrained environments – as is the case in USAR scenarios – relying only on one type of sensor is often insufficient, while fusing complementary information (i.e. information from different types of sensors) enables semantic interpretability of scenes and superior reliability.

The main goal of this paper is to propose a mobile robot system for autonomous detection of victims in USAR scenarios. The system is capable of autonomous navigation and map learning, and localizes victims and objects of

interest in a 3D world coordinate system. Our setting approaches actual search and rescue operations in realism and complexity: Real human victims have to be localized in unstructured environments, even in the presence of background clutter and multiple thermal sources such as office equipment and heating. To this end, we expand the scenario beyond the current RoboCup Rescue competition, in which the environment is built of simple structural elements and a thermal camera is sufficient for victim localization in most cases. By adding objects of interest to the metric map of the environment, we augment the robot's environment with semantic information, which can then be utilized for decision making by human operators. Our system is able to achieve high performance even for cluttered and complex datasets (see Fig. 1). All information is processed onboard and in real-time, as is crucial for realistic rescue deployments.

In our system visual information is supplied to a generic object detector that allows detecting structured objects and assigns them a semantic meaning (e.g. upper bodies of victims or hazardous material signs). To avoid relying on a single source of information, we consider the information from all sensors simultaneously, and derive a generic model that is able to leverage complementary information. As motivated in [5], merging different sources of information helps achieving higher levels of performance in victim detection. However, to the best of our knowledge, this work is the first attempt at integrating and evaluating state-of-the-art visual object detection with an autonomous USAR robot system.

Sec. 2 describes our system, while in Sec. 3 the sensors and detection algorithms for victim and object detection are introduced. Afterwards two approaches for sensor fusion are presented in Sec. 4. Experimental validation is presented in Sec. 5.

Related work. Enhancing geometric world models with semantic information is motivated in [6], where an indoor environment is described by corridors, doors, and types of rooms. Features are extracted from camera images and laser range data, and subsequently classified via hidden Markov models. The paper shows that semantic world models are helpful for behavior planning. In our context, we focus on detecting objects of interest, for example victims, rather than rooms and doors.

State-of-the-art algorithms for people detection frequently rely on only one type of sensor. In particular, a large body of literature exists for people detection in visual camera images and video sequences. Much of the progress in this area has been achieved by combining statistical machine learning techniques with robust image representations. High variability in people poses and appearance is often addressed using part-based models, in which parts either correspond to anatomically meaningful body parts [7], or are automatically inferred from data [8,9]. In [10] an image descriptor based on histograms of oriented gradients (HOG) is proposed, which is combined with discriminative SVM classifiers in order to exhaustively scan an image for people hypotheses. This representation has been further extended in [11] to incorporate motion information and has been used as basis for multiple approaches to people and object detection [12,13,14].

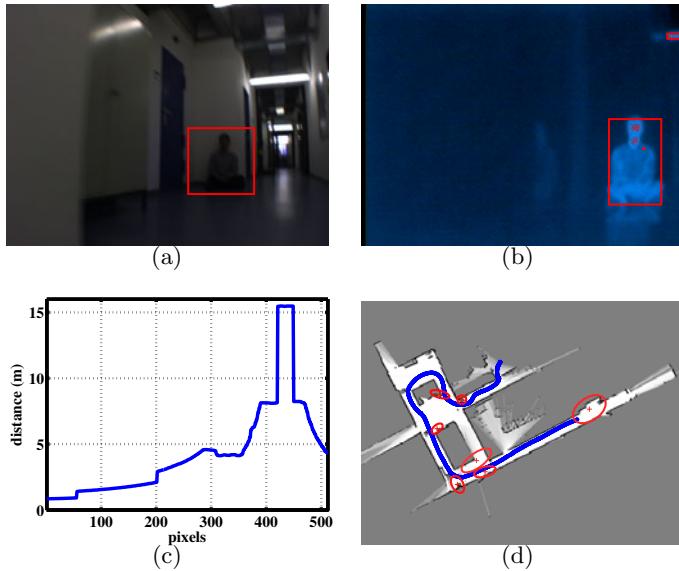


Fig. 1. Examples of sensor and world model data: (a) Visual image with victim detection. (b) Thermal image with heat detections. (c) Range samples along the horizontal axis of the image. (d) Snapshot of the semantic world model with estimated victim locations denoted by the red covariance ellipses.

Although being conceptually simple, HOG-based detectors belong to the best people detection methods published to date [15].

Thermal images can be used to detect upright people [16,17], to distinguish upright people from people lying on the ground [18], and to detect body parts and arbitrary postures of humans [19]. Laser range finders are frequently used to detect upright standing or walking people, either by tracking the upper body [20], the legs [21], or both [22]. All of these algorithms have restrictions and work only in specific situations. Some require upright standing or walking people, others assume to have no other heat sources than humans in the sensing range. These drawbacks can be overcome by using several heterogeneous, complementary types of sensors. However, combinations of several heterogeneous sensors have been shown to perform better than each classifier alone. In [23,24] detections from a laser range finder are used to classify regions of interest for a visual detector, and in [5] several independent classifiers relying on heat, skin color, motion and face detection are fused using Markov random fields.

2 System Overview

Since the locations of victims need to be specified in world coordinates, the robot pose and a metric map have to be estimated using a combination of inertial

sensing and simultaneous localization and mapping (SLAM). These estimates are continuously updated over time and used for integrating victim hypotheses obtained from different sensor types. Simultaneously with the pose and map estimation at each time step, our system generates a set of object hypotheses using a visual object detector and thermal-camera based detector, which are used as a basis for sensor fusion.

We explore two complementary approaches for sensor fusion. In the first, which we denote as explicit sensor fusion, we integrate information from different sensors directly in the sensor space using known transformations between different sensor modalities (i.e. the mapping between thermal and visual images). This allows to use known dependencies of sensor signals to either amplify or attenuate the confidence in the measurements.

In the second approach, which we denote as implicit sensor fusion, the global belief is updated independently for each observation. The advantage of using complementary sensors is realized through accumulation of positive evidence in the world model. Integrating hypotheses into the model requires an association step for matching a hypothesis to an already known object. The case in which previously unknown objects have been found must be considered separately. Once association is established, the matching is taken for granted and the corresponding victim location estimate and evidence is updated by using an extended Kalman filter (EKF). Integration of observations into the global belief state can therefore be considered to be a method for temporal sensor fusion. Additionally, confidence in a hypothesis is influenced by negative observations, where the absence of expected detections or contradictory measurements reduce the confidence value.

When applying implicit sensor fusion the observations from different sensors are integrated independently into the world map, while explicit sensor fusion is an optional step that precedes the integration step, and is primarily used to increase the reliability of observations.

2.1 World Model

World models generally account for a mathematical description of the environment, with different aspects being considered important depending on the application. In our USAR scenario the model is formed by a representation of building geometry and additional semantic information, like the location of people and objects of interest. By applying additional high-level knowledge to the model, it can be easily enriched with more detailed information in future work, e.g. classification of places, a graph of passable paths through a building, or estimates of hazardousness of specific locations. Based on this high-level description of the environment, the robot is able to plan reasonable future actions and – when integrating human operators – is able to deliver valuable information to rescue teams, e.g. to guide them to detected victims.

The robot state vector \mathbf{y}_k contains the estimated 6DOF robot pose as well as translational and angular velocities in the global coordinate system and is updated at discrete timesteps $t = t_k$. The location of objects, including the

victims, is referred to as \mathbf{x}_k^j with j being an index variable over the estimates. The objects are modeled as points, ignoring their spatial extent. In this paper we assume the world to be static apart from the movement of the robot itself. The number of objects is not known in advance. Besides the location information we introduce the probability π_k^j that object j is detected correctly as a measure of confidence, which is incrementally updated with each new sensor reading and typically increases when more detections of the same object occur.

The process of world modeling requires inference in state space from measurements given in sensor-space. Since sensors are error-prone, a probabilistic model description is used here. We choose a Gaussian representation for the continuous state variables, with estimated means $\hat{\mathbf{y}}_k$ and $\hat{\mathbf{x}}_k^j$, and variances C_k and P_k^j , respectively.

2.2 Simultaneous Localization and Mapping (SLAM)

State estimation of the vehicle and a map is performed by two components. A 2D pose and map estimate is provided by a module using incremental maximum likelihood alignment of laser scans with the estimated map. The map is represented by a discrete grid and updated using the log-odd probabilities of occupancy [25,26].

Estimation of the robot state \mathbf{y}_k is performed by an extended Kalman filter (EKF) integrating observations from all available sensors. Attitude estimation is provided by a built-in IMU and compass, while position estimation is provided by wheel encoders and the 2D pose estimation updates from the SLAM module. For the USAR scenarios described in this work, our approach is sufficiently accurate as to not require multiple map hypotheses (e.g. using a Rao-Blackwellized particle filter), or explicit loop closure.

3 Victim and Object Detection

In order to enrich the map with semantic information, we perform on-board detection of objects of interest, which in our case correspond to people and dangerous materials marked with hazmat signs. In this paper we focus on the detection of upper bodies of people, since this allows to detect both standing people as well as possibly injured people sitting on the ground (see Fig. 1), and leave more complex cases for future work. Due to background clutter, partial occlusions and complex articulations, visual people detection is a difficult problem even in this somewhat restricted setting. In particular, state-of-the-art computer vision methods are still severely challenged by this task [12].

Object detection. In order to find initial hypotheses of people and hazmat signs in camera images, we use the popular sliding window approach. In this approach every image is exhaustively scanned over a range of positions and scales; for each position and scale a discriminative SVM classifier is used to make binary decisions about the presence or absence of an object. While seemingly expensive, the sliding-window approach is especially suitable for parallel implementation, since each object location can be examined independently of the rest of the image.

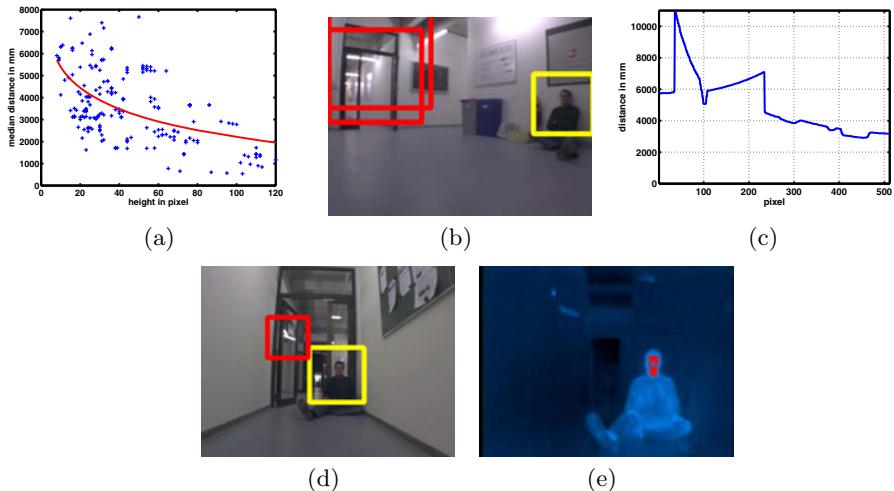


Fig. 2. (a) Correspondence between annotation height and distance. (b) Example detections of frame 561. (c) Scanline of frame 561. (d) Example detections of frame 287. (e) Thermal image of frame 287.

In order to describe the contents of the image at each particular location, we leverage recent results in computer vision and rely on a histogram of oriented gradients (HOG) descriptor [10]. In our system we scan the image with steps of 8 pixels and relative scale factors of 1.05. We use the GPU implementation [14] developed in our group, which allows to achieve real-time performance without sacrificing recognition performance.

The confidence $s_k^{\text{vis}} \in [0, 1]$ of a hypothesis is calculated via a sigmoidal mapping:

$$s_k^{\text{vis}} = \frac{1}{1 + \exp(a \cdot f_k + b)}, \quad (1)$$

where f_k is the SVM score of the hypothesis, and a and b are parameters that are estimated by cross-validation [27].

Object classification. A HOG descriptor is especially well suited for capturing the characteristic shape of an object. However, it has shortcomings when it is necessary to distinguish between objects with similar shape, such as different hazmat signs, all of which have a rhombus shape and differ mainly in color, internal patterns and text. In order to identify hazmat signs we augment the HOG descriptors with color histograms. For each hazmat sign hypothesis of the HOG based detector, we compute a color histogram in LAB color space and use this to perform the final classification of hazmat signs by applying a k-nearest neighbor approach in combination with the χ^2 -distance. As our experiments demonstrate, the combination of HOG and color histograms yields good performance for hazmat sign classification (Sec. 5).

Thermal victim detection. In addition to victim hypotheses from visible light camera images, our system also creates a set of hypotheses based on images from a thermal camera. These thermal hypotheses are generated with a simple procedure that searches the images for large enough groups of connected pixels with temperature values within the human body temperature range. Each such group of pixels is used to generate a hypothesis. Although hypotheses generated by the thermal camera alone are significantly less reliable compared to hypotheses produced by the visual object detector, we found them to be effective in reducing the number of false positives.

We define a simple model for the confidence $s_k^{\text{therm}} \in [0, 1]$ by counting the number of pixels within the person's bounding box that have a temperature close to human bodies. This model is robust to small offsets in corresponding locations in visual and thermal images, which arise due to imprecise synchronization between these modalities.

4 Sensor Fusion

The reliability of the entire victim detection framework can be increased by fusing victim hypotheses from different sensors and across time steps. Intuitively, the confidence of a detected victim should be increased if it is observed in several update steps or by different sensors and on the other hand stay below a certain threshold when it is only spotted once. We employ an extended Kalman filter (EKF) in order to update the locations \mathbf{x}_k^j of victims in our world model and integrate several hypotheses across update steps. In parallel, the confidence π_k^j that the victim is present at the respective location is updated in a separate filter with the respective measurement confidence as described below.

For simplification of notation we assume without loss of generality that at most one hypothesis is observed in every update step k . We define the measurement \mathbf{z}_k to consist of the distance d_k , the bearing angle α_k , and the relative vertical angle β_k between the hypotheses and the robot:

$$\mathbf{z}_k = [d_k \ \alpha_k \ \beta_k]^T = h(\mathbf{x}_k^j, \mathbf{y}_k) + \mathbf{v}_k , \quad (2)$$

where $h(\cdot, \cdot)$ refers to a nonlinear measurement function that projects the victim's position into the world model. \mathbf{x}_k^j and \mathbf{y}_k denote the victim's position estimate and robot state vector respectively. The random vector \mathbf{v}_k is unbiased and uncorrelated Gaussian measurement noise with hand-tuned variance R . The measurement function also depends on the robot's state \mathbf{y}_k , which in turn is estimated with an EKF independently.

Data association. In order to find an optimal matching between measurements and existing estimates of victim locations we use the following probability of measurement \mathbf{z}_k given the index j and the position estimate $\hat{\mathbf{x}}_{k-1}^j$ with variance P_{k-1}^j :

$$p(\mathbf{z}_k | j) \propto \mathcal{N}(\mathbf{z}_k; h(\hat{\mathbf{x}}_{k-1}^j, \hat{\mathbf{y}}_k), R + H_k^j P_{k-1}^j (H_k^j)^T) \quad (3)$$

with a first order approximation H_k^j of the measurement function $h(\hat{\mathbf{x}}_{k-1}^j, \hat{\mathbf{y}}_k)$ at the current estimated means.

Whenever this probability $p(\mathbf{z}_k|j)$ is above a previously defined threshold, we associate the new measurement to the best matching estimate with index $j_k^* = \arg \max_j p(\mathbf{z}_k|j)$. Otherwise, a new estimate is added to the world model as of a previously unobserved victim.

Kalman filter updates. We assume the victims to be static in our setting and therefore no explicit prediction step is needed. The measurement update equations of the Kalman filter are defined as:

$$K_k^j = P_{k-1}^j (H_k^j)^T \left(H_k^j P_{k-1}^j (H_k^j)^T + R \right)^{-1} \quad (4)$$

$$\hat{\mathbf{x}}_k^j = \hat{\mathbf{x}}_{k-1}^j + \lambda_k(\mathbf{z}_k, s_k) \cdot K_k^j (\mathbf{z}_k - h(\hat{\mathbf{x}}_{k-1}^j, \hat{\mathbf{y}}_k)) \quad (5)$$

$$P_k^j = \left(I - \lambda_k(\mathbf{z}_k, s_k) \cdot K_k^j H_k^j \right) P_{k-1}^j \quad (6)$$

where I denotes the identity matrix and s_k refers to the initial score of hypothesis k as will be explained below. The measurement update uses the confidence $\lambda_k(\cdot, \cdot) \in [0, 1]$ of an observation as an additional factor to the gain matrix K_k^j to honor the observation quality and discard uncertain updates. The measurement confidence is of the form

$$\lambda_k(\mathbf{z}_k, s_k) = s_k \cdot \phi(\mathbf{z}_k, d^{\text{laser}}) \cdot \psi(\mathbf{z}_k), \quad (7)$$

where d^{laser} is the distance to the next obstacle measured with the laser scanner. $\phi(\cdot, \cdot)$ imposes a prior on the estimated distance to \mathbf{z}_k and measured distance d^{laser} to the next obstacle and is proportional to a Gaussian with manually defined variance according to the uncertainty in the sensor measurements. $\psi(\cdot)$ refers to a Gaussian height prior with mean 80cm (height of upper bodies) and manually defined variance. By employing $\phi(\cdot, \cdot)$, we ensure that the size of an hypothesis approximately matches the size that we expect, and avoid false positives with inappropriate estimated and measured distances (see Fig. 2(b)). $\psi(\cdot)$ guarantees that all objects appear at the expected height from the robot, while unlikely pitch angles are discarded.

Label confidence update. In the case that a new measurement is associated to a given estimate, we update the estimate's label confidence according to the disjunctive combination of two binary random events, so that confidence is increased with every new measurement:

$$\pi_k^j = \pi_{k-1}^j + \lambda_k \cdot (1 - \pi_{k-1}^j). \quad (8)$$

If no measurement in time step k is available we decrease the label confidence of all victim estimates within the field of view by employing "negative evidence". Negative evidence is information that arises from the fact that the confidence of an estimate can decrease if it is not confirmed by sensor observations. Applying negative evidence to our algorithm helps to decrease the number of false alarms, as many false positives do not reoccur in consecutive time steps.

The negative update is applied to all objects j that should be visible in the image according to the current estimated map and positions, but have no detection event associated for the current time step. Their label confidence is reduced according to

$$\pi_k^j = \frac{p_{\text{miss}} \cdot \pi_{k-1}^j}{p_{\text{miss}} \cdot \pi_{k-1}^j + (1 - \pi_{k-1}^j)} . \quad (9)$$

The probability p_{miss} of missed detections is approximated as the inverse probability of the detector's recall on the trained dataset.

Implicit vs. explicit integration. We evaluate two different fusion schemes: implicit and explicit fusion. These two approaches differ in the way the complementary information of sensors is integrated. In our model this boils down to the treatment of the initial score s_k of an hypothesis in Eq. (7).

In the implicit fusion scheme we consider each hypothesis from both the visual light and thermal sensor as a new measurement that is either associated to a given estimate or enters the world model as a new estimate. In this setting the sensor fusion is implicitly handled with the Kalman filter, since the measurements of both sensors can be used for data association and updating the confidence. Here we directly use the visual or thermal score as initial score s_k :

$$s_k = \begin{cases} s_k^{\text{vis}}, & \text{if visual light hypothesis} \\ s_k^{\text{therm}}, & \text{if thermal hypothesis} \end{cases} . \quad (10)$$

In the explicit integration scheme we compute the overall detection confidence as a weighted sum from the individual scores of complementary sensors, yielding a single observation model where bearing and distance information is taken from the visible light bounding box only:

$$s_k = \gamma_1 \cdot s_k^{\text{vis}} + \gamma_2 \cdot s_k^{\text{therm}} + \gamma_3 \cdot s_k^{\text{laser}} , \quad (11)$$

where $\sum_i \gamma_i = 1$ and the coefficients γ_i are trained with cross-validation. This additive formulation makes the model robust to sensor failures (e.g. due to partial occlusions) and takes relative importance of different sensors into account. The first two components of the mixture correspond to the probability of correct detection given the score of the SVM classifier and the output probability of the thermal victim detector for the same bounding box as defined in Sec. 3.

While more detailed integration of different sensor modalities is possible and we plan to explore it in the future, we opt for this re-estimation approach since it allows to decouple training of the visual object detector from the rest of the system, does not require exact synchronization between different sensor streams, and allows to use simple algorithms for integration of thermal and laser sensors.

In order to model s_k^{laser} , we fit a log-linear model to a set of jointly observed bounding boxes and laser range measurements as shown in Fig. 2(a). s_k^{laser} is set to a Gaussian computed at the difference between the predicted distance from the log-linear model and the median distance measured with the laser range finder. The variance is set by hand.

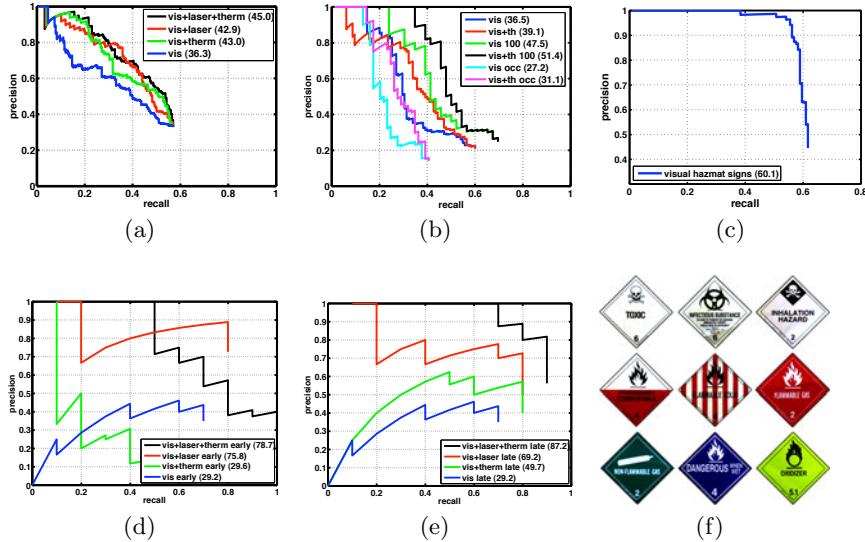


Fig. 3. (a, b) Single-frame people detection performance for different combination of sensors on “Hector Data 1” and “Hector Data 2” datasets. (c) Single-frame hazmat sign detection performance on “Hector Data 1” dataset. (d, e) People detection performance of the full system on the ”Hector Data 1” dataset for explicit and implicit sensor fusion schemes. (f) Collection of different hazmat signs.

5 Experiments

We evaluate the performance of our system on the tasks of people and hazard sign detection. In particular we quantify performance gains due to fusion of multiple sensor modalities and evaluate both detection in single frames and performance of the full system. For the evaluation we use the dataset, which consists of daylight images, thermal images, and laser range scanner and odometry measurements collected while robot was driving along the closed path of approximately 120 meters around the office building. For the sake of single-frame evaluation, we have annotated all people appearing in the daylight images, which are larger than 40 pixels in height. The resulting dataset contains 1480 daylight images with 300 annotated victims corresponding to 10 distinct subjects. Due to difficult illumination conditions, motion blur and large variability in viewpoints, visual people detection in such data is very challenging. At the same time detection of people in thermal images is complicated by presence of multiple background heat sources, such as heating and illumination equipment, computers and other office devices. In order to evaluate the robustness of our system to partial occlusions, we have collected an additional dataset of 28 images with 115 annotated people, 69 of which are partially occluded. We denote these datasets as “Hector Data 1” and “Hector Data 2” respectively¹. In order to demonstrate the

¹ Both datasets are available at <http://www.gkmm.tu-darmstadt.de/rescue>

generality of our method we do not adapt visual people detector to our scenario, although this would likely lead to improved performance, and train our visual detector on the INRIA pedestrian dataset [10], where we have re-annotated the upper bodies of people. For the detection experiments we report the average precision (AP), which measures the area under the precision-recall curve. This is a common comparison measure, for which a perfect detector would achieve 100% AP. In the following we first present the results of single-frame detection of people and hazmat signs, and then evaluate performance of the full system. For the single frame detection the confidence of object hypothesis is computed according to the Eq. 11, while for the full system the confidence is based on the measurements over multiple frames.

Single frame people detection. Fig. 3(a) and Fig. 3(b) show the results of single-frame people detection of our system on the “Hector Data 1” and “Hector Data 2” datasets in form of recall/precision curves.

On the “Hector Data 1” dataset, detector based on visual information achieves 36.3% AP, integration of visual and laser range measurements results in 42.9% AP, and integration of visual and thermal measurements results in 43.0% AP. Integration of all three sensors leads to the best performance of 45.0% AP. The missing detections on this dataset mainly correspond to either very small or very blurry instances.

Similar trends can be observed on the “Hector Data 2”, where images contain less motion blur, but significant number of people is partially occluded. On this dataset we obtain 36.5% AP using visual detector alone, which improves to 39.1% AP by integration visual and thermal detectors. When evaluating only on the partially occluded people we obtain 27.2% AP with visual detector, and 31.1% AP with combination of visual and thermal detectors. These results show that, despite some drop in performance, our system is still producing meaningful detection results even in the case when people are partially occluded. The integration of thermal sensor measurements results in consistent improvement of performance of around 4% AP.

Hazmat sign detection and classification. Fig. 3(c) shows precision-recall curve quantifying single-frame hazmat sign detection performance. On this type of objects we obtain 60.1% AP. Due to smaller intra-class variability the results for hazmat signs are somewhat better than results for people detection. The missing detections are often due to motion blur and hazmat signs at extremely small scales.

We further investigate the performance of our system on hazmat sign classification task, in which the goal is to distinguish between one of the nine hazmat sign classes depicted in Fig. 3(f). For that purpose, we take the detection windows at maximum recall and assign them to one of the given classes or background.

For the classification we follow the procedure based on color histograms, described in Sec. 3. We evaluate two approaches to histogram computation, one in which color histogram is calculated on the entire detection window, and another in which detection window is subdivided into four sub-regions and separate

histogram is computed for each of them. In the latter approach the final descriptor is formed by concatenating histograms of each sub-region. We obtain the recognition rate of 37.5% using histograms based on the entire window, and 58.3% using sub-region based histograms. The improvement is mainly due to better discrimination between hazmat classes with globally similar color distribution, e.g. white/red hazmat signs “Combustible” and “Flammable Solid” shown on Fig. 3(f). Region-based histograms provide better representation of the image in such difficult cases, since they are also capable of capturing the spatial distribution of colors within the detection window.

Full system performance. Finally, we evaluate the capability of our full system to correctly detect and localize people in the environment map. The predicted location and detection confidence of each person hypothesis is inferred by temporal integration of sensor measurements according to the filtering procedure described in Sec. 4. In contrast to single frame evaluation, the detection performance is reported for the whole series of measurements contained in the dataset. The victim is considered to be localized correctly if its predicted location on the map is within 1 meter radius the ground truth annotation, obtained by manual labeling. Multiple hits on the same ground truth annotation are only counted once, where each subsequent hit is considered a false positive.

As can be seen in Figs. 3(d) and 3(e) merging complementary information of heterogeneous devices (vis+laser+therm) outperforms all other settings by a large margin. It achieves 78.7% AP (explicit sensor fusion) and 87.2% AP (implicit sensor fusion) outperforming vis+laser by 2.9% AP and 18% AP respectively. When not using the laser, our framework suffers from placing the victims too far from ground truth annotations. vis+therm achieves 29.6% AP for explicit fusion and 49.7% AP for implicit fusion. The baseline of using only visual information achieves 29.2% AP. The implicit integration scheme achieves a higher precision for vis+thermal+laser and vis+thermal than explicit integration while the latter fusing scheme yields higher levels of recall. Note that in contrast to single-frame evaluation where recall levels are below 60%, the complete system has recall of 90% for implicit and 100% for explicit sensor fusion schemes. This is an important result for search and rescue applications, in which the ultimate goal is to find all of the victims.

6 Conclusion

This paper addresses sensor fusion of heterogeneous sensors with a generic semantic world model. Our framework is able to leverage complementary information for increased reliability in complex USAR scenarios. Geometric maps are enriched with semantic interpretation of scenes by detecting victims and possibly hazardous areas. The importance of sensor fusion and the expressiveness of our model are experimentally evaluated on a complex real world dataset. In future work we will address distributed sensor fusion by using multiple robots.

Acknowledgments. This work was supported by the DFG GRK 1362. The authors are thankful to C. Wojek for providing the GPU HOG implementation.

References

1. Asada, M., Shirai, Y.: Building a world model for a mobile robot using dynamic semantic constraints. In: IJCAI 1989, pp. 1629–1634 (1989)
2. Burgard, W., Hebert, M.: World modeling. In: Siciliano, B., Khatib, O. (eds.) Springer Handbook of Robotics, pp. 853–869. Springer, Heidelberg (2008)
3. Kumar, S., Guivant, J., Durrant-Whyte, H.: Informative representations of unstructured environments. In: ICRA (2004)
4. Tadokoro, S., et al.: Robocup rescue project. Advanced Robotics 14(5), 423–425 (2000)
5. Kleiner, A., Kümmerle, R.: Genetic MRF model optimization for real-time victim detection in search and rescue. In: IROS (2007)
6. Rottmann, A., Mozos, O.M., Stachniss, C., Burgard, W.: Semantic place classification of indoor environments with mobile robots using boosting. In: AAAI (2005)
7. Andriluka, M., Roth, S., Schiele, B.: Pictorial structures revisited: People detection and articulated pose estimation. In: CVPR (2009)
8. Felzenszwalb, P., McAllester, D., Ramanan, D.: A Discriminatively Trained, Multiscale, Deformable Part Model. In: CVPR (2008)
9. Bourdev, L., Malik, J.: Poselets: Body part detectors trained using 3d human pose annotations. In: ICCV (2009)
10. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR (2005)
11. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 428–441. Springer, Heidelberg (2006)
12. Ferrari, V., Marin, M., Zisserman, A.: Progressive search space reduction for human pose estimation. In: CVPR (2009)
13. Schnitzspan, P., Fritz, M., Schiele, B.: Hierarchical Support Vector Random Fields: Joint Training to Combine Local and Global Features. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 527–540. Springer, Heidelberg (2008)
14. Wojek, C., Dorkó, G., Schulz, A., Schiele, B.: Sliding-Windows for Rapid Object Class Localization: A Parallel Technique. In: Rigoll, G. (ed.) DAGM 2008. LNCS, vol. 5096, pp. 71–81. Springer, Heidelberg (2008)
15. Dollar, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: A benchmark. In: CVPR (2009)
16. Jüngling, K., Arens, M.: Feature based person detection beyond the visible spectrum. In: CVPR Recognition Workshops (2009)
17. Davis, J., Sharma, V.: Robust detection of people in thermal imagery. In: ICPR 2004 (2004)
18. Pham, Q.C., Gond, L., Begard, J., Allezard, N., Sayd, P.: Real-time posture analysis in a crowd using thermal imaging. In: CVPR (2007)
19. Markov, S., Birk, A.: Detecting humans in 2d thermal images by generating 3d models. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 293–307. Springer, Heidelberg (2007)
20. Fod, A., Howard, A., Mataric, M.J.: Laser-based people tracking. In: ICRA (2002)
21. Arras, K., Grzonka, S., Luber, M., Burgard, W.: Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In: ICRA (2008)

22. Carballo, A., Ohya, A., Yuta, S.: Multiple people detection from a mobile robot using double layered laser range finders. In: ICRA Workshop (2009)
23. Zivkovic, Z., Kröse, B.: Part based people detection using 2d range data and images. In: ICRA (2007)
24. Gate, G., Breheret, A., Nashashibi, F.: Centralized fusion for fast people detection in dense environment. In: ICRA (2009)
25. Schiele, B., Crowley, J.: A comparison of position estimation techniques using occupancy grids. In: ICRA (1994)
26. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
27. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Advances in Large Margin Classifiers (1999)

Improving Biped Walk Stability Using Real-Time Corrective Human Feedback

Çetin Meriçli^{1,2} and Manuela Veloso¹

¹ Computer Science Department,

Carnegie Mellon University

Pittsburgh, PA 15213, United States

² Computer Engineering Department,

Boğaziçi University

Bebek, Istanbul, Turkey

{cetin,veloso}@cmu.edu

Abstract. Robust walking is one of the key requirements for soccer playing humanoid robots. Developing such a biped walk algorithm is non-trivial due to the complex dynamics of the walk process. In this paper, we first present a method for learning a corrective closed-loop policy to improve the walk stability for the Aldebaran Nao robot using real-time human feedback combined with an open-loop walk cycle. The open-loop walk cycle is obtained from the recorded joint commands while the robot is walking using an existing walk algorithm as a black-box unit. We capture the corrective feedback signals delivered by a human using a wireless feedback mechanism in the form of corrections to the particular joints and we present experimental results showing that a policy learned from a walk algorithm can be used to improve the stability of another walk algorithm. We then follow up with improving the open-loop walk cycle using advice operators before performing real-time human demonstration. During the demonstration, we then capture the sensory readings and the corrections in the form of displacements of the foot positions while the robot is executing improved open-loop walk cycle. We then translate the feet displacement values into individual correction signals for the leg joints using a simplified inverse kinematics calculation. We use a locally weighted linear regression method to learn a mapping from the recorded sensor values to the correction values. Finally, we use a simple anomaly detection method by modeling the changes in the sensory readings throughout the walk cycle during a stable walk as normal distributions and executing the correction policy only if a sensory reading goes beyond the modeled values. Experimental results demonstrate an improvement in the walk stability.

Keywords: complex motor skill acquisition, learning from demonstration, motion and sensor model learning, human-robot interfaces.

1 Introduction

Biped walk learning is a challenging problem in humanoid robotics due to the complex dynamics of walking. Developing efficient biped walking methods on commercial humanoid platforms with limited computational power is even more challenging since the

developed algorithm should be computationally inexpensive, and it is not possible to alter the hardware.

The Nao (Fig.1), is a 4.5 kilograms, 58 cm tall robot with 21 degrees of freedom (www.aldebaran-robotics.com). It does not have separate hip yaw joints for the legs. Instead, both legs have mechanically connected hip yaw-pitch joints perpendicular to each other along the Y-Z plane (Fig.1(c)) and these two joints are driven by a single motor. The Nao is equipped with a variety of sensors including a 3-axis accelerometer, a 2-axis (X-Y) gyroscope, and an inertial measurement unit for computing the absolute torso (upper body of the robot) orientation using accelerometer and gyroscope data. The inertial measurement unit, the accelerometer, and the gyroscope sensors use a right-hand frame of reference (Fig.1(b)). We use the term “Yaw” for rotation along the Z axis, “Roll” for rotation along the X axis, and “Pitch” for rotation along the Y axis throughout the text.

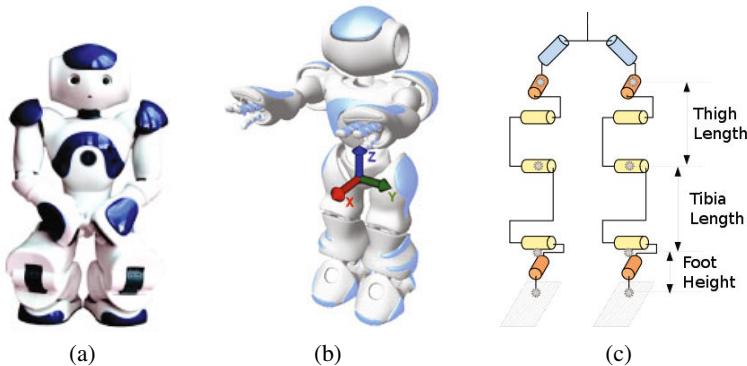


Fig. 1. a) The Nao robot b) The frame of reference for sensors. c) Kinematic configuration of the legs.

Numerous research studies have been published on the biped walk algorithms for the Nao robot since the introduction of the Nao to the RoboCup SPL (www.robocup.org). Graf *et al.* present an omni-directional walking algorithm using parameterized gaits and sensor feedback [1,2]. Liu and Veloso develop an efficient Zero Moment Point (ZMP) search method [3,4]. Gökçe and Akın utilize Evolutionary Strategy (ES) to tune the parameters of a Central Pattern Generator (CPG) based walk [5]. Strom *et al.* present a ZMP based omnidirectional walking algorithm [6]. Czarnetzki *et al.* propose a preview control based method for keeping the ZMP on the desired path [7].

There have also been approaches utilizing the learning from demonstration paradigm for task and skill learning. Nakanishi *et al.* present a method for biped walk learning from human demonstrated joint trajectories using dynamical movement primitives [8]. Grollman and Jenkins propose a learning from demonstration framework called “Dogged Learning” [9] and successfully applied it to learning quadruped walking and a set of skills related to playing soccer on a Sony Aibo robotic dog [10]. Argall *et al.* present a learning from demonstration and corrective feedback method for low level motion planning of a

Segway RMP robot [11,12]. Chernova and Veloso present a sliding autonomy framework for teaching tasks to single robots [13] and multi-robot systems [14].

In this paper, We first describe a method for obtaining a single walk cycle using an existing walk algorithm and how the obtained walk cycle can then be played back to presented an overview of the method, along with initial experimentation on two different walk algorithms. We present experimental results demonstrating how a correction policy learned using an existing walk algorithm is able to improve the walk stability on both the initial algorithm and a second algorithm, showing that the learned correction policy using the proposed method does not depend on the underlying walk algorithm.

We then contribute a biped walk stability improvement algorithm using human feedback consisting of three phases, where the first phase being the walk cycle capture and playback presented in the first part. In the second phase, we present an offline improvement method for the open-loop walk using advice operators. Finally, we introduce a closed-loop feedback policy learning method which uses the corrective human demonstration given in real-time in the form of foot position displacements to learn a mapping from the sensory readings to a corresponding correction value for the positions of the feet. We present experimental results for the performance evaluation of the learned policy against the open-loop playback algorithm, and the open-loop playback algorithm improved using advice operators. The results show improvement at second and third phases over the performance of the initial phase.

2 Proposed Approach

Walking is a periodic phenomenon and consists of consecutive walk cycles which starts with a certain configuration of the joints and ends when the same configuration is reached again. A walk cycle wc is a motion segment of duration T timesteps, where $wc_j(t), t \in [0, T]$ be the command to the joint j provided at timestep t .

Although the Nao robot has a total of 21 joints, for our approach, we use a subset of 12 of them named *Joints*: arm roll, hip roll, hip pitch, knee pitch, ankle pitch, and ankle roll joints for the left and the right arms and the legs.

2.1 Obtaining an Open-Loop Walk

We use an existing walk algorithm as a black-box and we collect a number of walk sequences where the robot is walking forwards for a fixed distance at a fixed speed using the black-box algorithm. We save the sequences in which the robot was able to travel the determined distance without losing its balance. A set of many example walk sequences where the robot walks without falling provide

Many examples of the robot walking without falling provide data D for each $t, t \in [0, T]$, in the form of the commands received for each joint $\vec{D}_j(t)$ and the sensory readings $S(t)$ for the set of sensors *Sensors*. We acquire a single walk cycle wc using D as $wc_j(t) = \mu(\vec{D}_j), j \in Joints, t \in [0, T]$. In addition, we fit a normal distribution on the readings of each sensor at each t $\mu(t), \sigma(t)$ where $\mu_s(t)$ is the mean, and $\sigma_s(t)$ is the standard deviation for the readings of the sensor $s \in Sensors$ at time t in the walk cycle (Fig. 2).

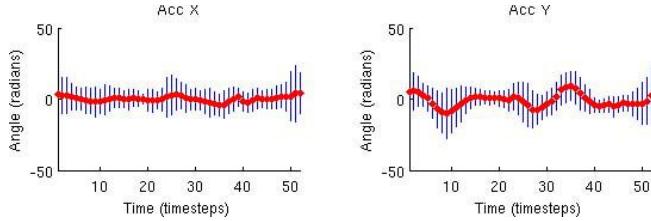


Fig. 2. Distribution of the sensor values over the complete walk cycle for a stable walk sequence. The middle line denotes the mean and the vertical lines denote $\pm 3\sigma$ variance. The X axis is timesteps, and the Y axis is the sensor value.

The actual movement of the robot differs from the desired movement as a result of the various sources of uncertainty associated with the sensing and actuation (Fig. 3) and therefore it is not possible to have an open-loop walk behavior that can walk indefinitely without falling.

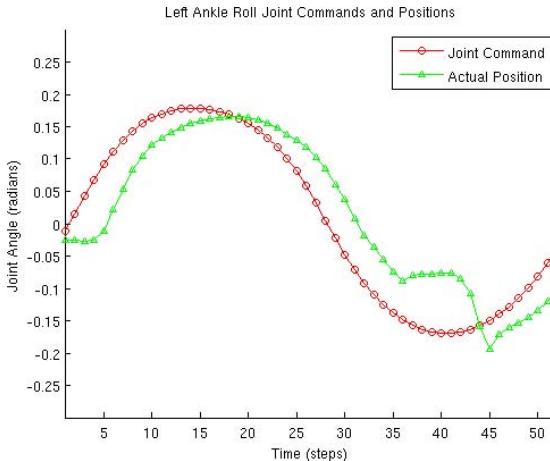


Fig. 3. An example to the actuation error from the ankle roll joint of the left leg. The plot with circles shows the joint commands, and the plot with triangles shows the actual trajectory the joint has followed. The error towards the end is caused by the weight of the robot on the left ankle of the robot while it is taking a right step and is standing on its left leg.

The changes in sensory readings when the robot is about to lose its balance can be used to derive a correction policy by mapping these changes to appropriate modifications on the walk cycle (Fig. 4). The next subsection describes a method for obtaining a closed-loop walk using sensory readings and human demonstration.

2.2 Correction Using Sensor-Joint Couplings

In our previous research, we presented a method where we introduce the idea of using human demonstration to learn a closed-loop correction policy [15]. We used the hip

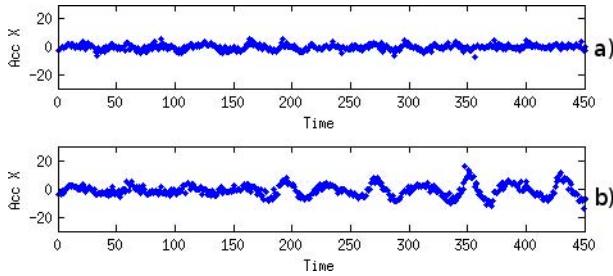


Fig. 4. Sample accelerometer readings: a) a stable walk sequence, and b) a walk sequence in which robot starts losing its balance after walking for some amount of time

roll and the hip pitch joints to apply the correction signals. We defined the correction function for each joint as a transformation function applied on a single sensor reading. At each timestep, we compute the correction values for all joints $j \in Joints$ using the recent sensor readings and the defined correction functions. We then add the calculated values to the joint command values in the walk cycle for that timestep before sending the joint commands to the robot. The noisy nature of sensors causes fluctuations in the sensory readings which may result in jerky motions and therefore loss of balance when used directly to generate a correction signal. We smooth the sensory readings using running mean smoothers. We use human demonstration to learn the mapping function from sensor readings to the correction signals. The human demonstrator provide the correction signals in the form of angle offsets to the joint commands using a wireless game controller interface. We model the received demonstration data as a function of accelerometer data by fitting normal distributions.

We used the walking algorithm proposed by Liu and Veloso which uses online ZMP sampling [3,4] for learning the correction policy. The efficiency of learned feedback policy is then evaluated using the default walk algorithm provided by Aldebaran Robotics with default parameters and 30 timesteps per walking step (W1), and Liu and Veloso's walking algorithm [4,3] based on online ZMP sampling (W2).

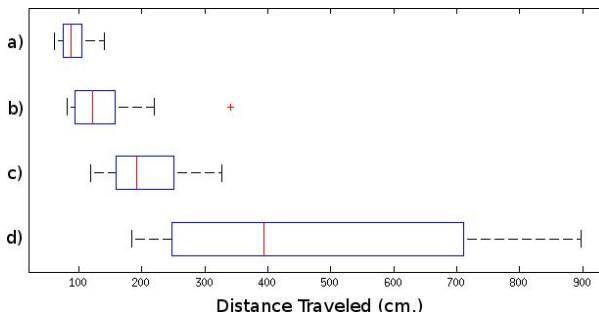


Fig. 5. Performance evaluation results: a) W1, open-loop, b) W1, learned policy using accelerometer readings, c) W2, open-loop, d) W2, learned policy using accelerometer readings

For each algorithm, 10 runs with original open-loop method and 10 runs using the learned policy for correction were conducted and the distance traveled before falling was recorded. The results are given in Fig.5. The plus sign is an outlier, the maximum distance that the learned policy on original Aldebaran walk traveled, the lines within the boxes mark the median, the marks at both ends of boxes indicate minimum and maximum distances, and the left and right edges of boxes mark 25th and 75th percentiles, respectively. The learned policy has improved the performance of both algorithms significantly despite that the policy was derived only using Liu and Veloso's algorithm.

Following up this experimentation, we extend the correction framework to include offline advice operators and multi-joint corrections with locally weighted regression as the function approximator.

2.3 Advice Operators

Advice Operators Policy Improvement (A-OPI) is a method for improving the execution performance of the robot in a human-robot LfD setup [12]. Advice operators provide a *language* between the human teacher and the robot student, allowing the teacher to give *advice* as a mathematical function to be applied on the observations and/or actions. The resulting data is then used to re-derive the execution policy. Advice operators are especially useful in domains with continuous state/action spaces where the correction must be provided in continuous values.

We use A-OPI for correcting the obtained walk cycle in its open-loop form based on human observations of the executed walk behavior. We define three advice operators that are applied on the walk cycle:

- **ScaleSwing(f):** Scales the joint commands of hip roll joints (along X axis) in the walk cycle by a factor f where $f \in [0, 1]$. Hip roll joints generate the lateral swinging motion while walking.
- **ChangeFeetDistance(d):** Applies an offset of d millimeters to the distance between the feet along Y axis.
- **SetArms($angle$):** Raises or lowers the arms by $angle$ radians along the Y-Z plane with respect to their baseline.

After a set of iterations consisting of execution of the walk behavior, receiving advice from the teacher, and revising the walk cycle accordingly, an improvement has been achieved. The initial and improved versions of hip roll joint values to generate lateral swinging motion are shown in Fig.6 as an example.

2.4 Using Human Demonstration for Learning Correction Policy

We introduced a wireless feedback delivery method without touching the robot in [15]. The proposed feedback method utilizes the Nintendo Wiimote commercial game controller [16] to provide corrective demonstration to the robot (Fig.7). The Wiimote controller and its Nunchuk extension are equipped with accelerometers which not only measure the acceleration of the controllers, but also allow their absolute roll and pitch orientations to be computed. The computed roll and the pitch angles are in radians and they use the right-hand frame of reference.

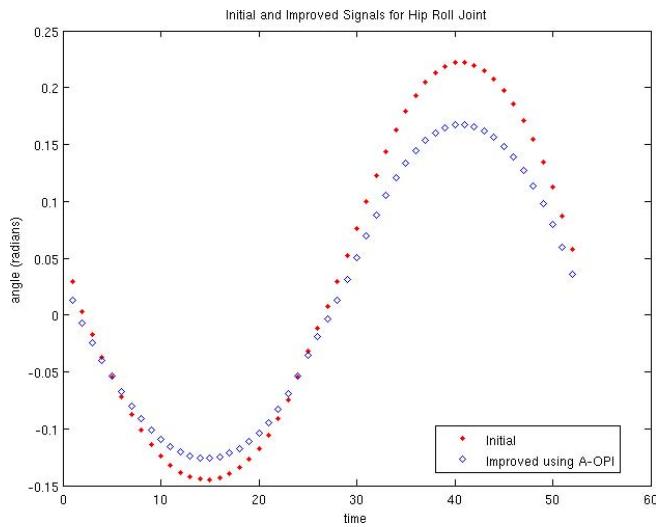


Fig. 6. Initial and improved joint commands for hip roll joints generating swinging motion while walking

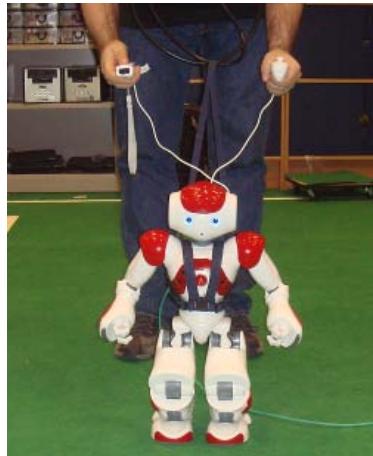


Fig. 7. A snapshot from a demonstration session. A loose baby harness is used to prevent possible hardware damage in case of a fall. The harness neither affects the motions of the robot nor holds it as long as the robot is in an upright position.

A scaling factor γ is applied on the Wiimote readings before they are sent to the robot. We used $\gamma = 50$ in our implementation. The changes in the orientations of the Wiimote handles (in radians) are mapped to the foot position displacements on the robot (in millimeters). Four measurable axes of the controller handles, namely Nunchuk yaw, Nunchuk pitch, Wiimote yaw, and Wiimote pitch, are used to control the displacement

of the left foot along the X axis, the left foot along the Y axis, the right foot along the X axis, and the right foot along the Y axis, respectively (Fig.8).

2.5 Correction Using Sensor-Feet Position Couplings

At each timestep of playback, the vector of joint command angles for that timestep is used to calculate relative positions of the feet in 3D space with respect to the torso using forward kinematics. The calculated corrections (in the autonomous mode), or the received corrections (during the demonstration) are applied on the feet positions in 3D space and the resulting feet positions are converted back into a vector of joint command angles using inverse kinematics.

Due to the physically connected hip-yaw joints of the Nao, inverse kinematics for feet positions cannot be calculated independently for the feet. Graf *et al.* propose an analytical solution to inverse kinematics of the Nao presenting a practical workaround for the connected hip-yaw pitch joints constraint [1]. We use a simplified version of this approach by assuming the hip-yaw joints to be fixed at 0 degrees for the straight walk.

The demonstrator uses the wireless interface to modify the robot's motion in real time while the robot is walking using the refined open-loop walk cycle. The correction values received during the demonstration are recorded synchronously with the sensory readings, tagged with the current position in the walk cycle. Each point in the resulting demonstration dataset is a tuple $\langle t, \vec{S}, \vec{C} \rangle$ where t is the position in the walk cycle at the time when this correction is received, \vec{S} is the vector of sensory readings, and \vec{C} is the vector of received correction signals with $\vec{S} = \{Acc_X, Acc_Y\}$ being the accelerometer readings, and $\vec{C} = \{C_X^{left}, C_Y^{left}, C_X^{right}, C_Y^{right}\}$ being the received correction values for the left foot along the X axis, the left foot along the Y axis, the right foot along the X axis, and the right foot along the Y axis, respectively.

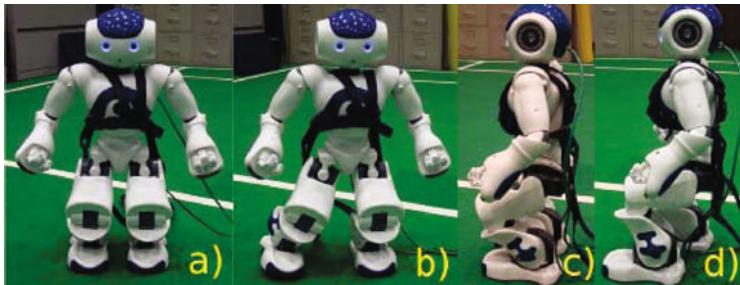


Fig. 8. Example corrections using the Wiimote. Rolling the Wiimote to the right takes the right leg of the robot from its neutral posture (a) to a modified posture along the Y axis (b). Similarly, tilting the Wiimote forward brings the right leg of the robot from its neutral posture (c) to a modified posture along the X axis (d).

We utilize locally weighted regression with a Gaussian kernel [17] for generalizing a policy using the recorded correction and sensor values. For each received sensor reading vector \vec{S} , we calculate a correction vector \vec{C} as follows:

$$d_i = e^{-\sqrt{(\vec{S} - \vec{S}_i(t))^T \Sigma^{-1} (\vec{S} - \vec{S}_i(t))}}$$

$$\vec{C} = \frac{\sum_i d_i \vec{C}_i(t)}{\sum_i d_i}$$

where Σ is the covariance matrix of the sensory readings in the demonstration set, $\vec{C}_i(t)$ is the i^{th} received correction signal for the walk cycle position t , $\vec{S}_i(t)$ is the i^{th} sensory reading for the walk cycle position i , $\vec{S}(t)$ is the current sensory reading, \vec{C} is the calculated correction value to be applied, and t is the current position in the walk cycle.

The calculated correction values are applied only if any of the sensor values are not in the range $\mu_t \pm K\sigma_t$ (i.e., an abnormal value is read from that sensor) where K is a coefficient, and t is the current position in the walk cycle. In our implementation, we chose $K = 3$ so the correction values are applied only if the current sensory readings are outside the $\mu_s(t) \mp 3\sigma_s(t)$, corresponding to the %99 of the variance of the initial sensory model. Algorithm 1 uses sensor-foot position couplings to perform a closed-loop walk.

Algorithm 1 Closed-loop walking using sensor-foot position couplings. Pos_{left} and Pos_{right} are the positions of the feet in 3D space.

```

 $t \leftarrow 0$ 
loop
   $\vec{S}(t) \leftarrow readSensors()$ 
   $\vec{S}(t) \leftarrow smooth(\vec{S}(t))$ 
   $Pos_{left}, Pos_{right} \leftarrow forwardKine(wc(t))$ 
  if  $(\mu_s(t) - K\sigma_s(t) \leq S_s(t) \leq \mu_s(t) + K\sigma_s(t))$  then
     $C_{left}, C_{right} \leftarrow 0$ 
  else
     $C_{left}, C_{right} \leftarrow correction(\vec{S}(t))$ 
  end if
   $Pos_{left} \leftarrow Pos_{left} + C_{left}$ 
   $Pos_{right} \leftarrow Pos_{right} + C_{right}$ 
   $NextAction \leftarrow inverseKine(Pos_{left}, Pos_{right})$ 
   $t \leftarrow t + 1 \text{ (mod } T)$ 
end loop

```

3 Experimental Results

We evaluated the performance of the proposed method on a flat surface covered with regular RoboCup SPL field carpet. We used the walking algorithm proposed by Liu and Veloso as the black-box open-loop algorithm. The duration of the extracted walk cycle is 52 individual timesteps, approximately corresponding to one second. During two

demonstration sessions of about 18 minutes, a total of 53014 data points are recorded. 19428 data points corresponding to about 373 walk cycles are selected as good examples of corrective demonstration by visually inspecting the demonstration data based on the changes in the sensory readings towards the recovery of the balance.

We evaluated the following algorithms:

- Initial open-loop playback walk.
- Open-loop playback walk after offline correction using advice operators.
- Closed-loop playback walk using the learned policy from real-time corrective demonstration.

For each algorithm, we made 10 runs and we measured the distance traveled before falling. The results are given in Fig.9. Although an improvement has been achieved by the sole application of the advice operators, the learned policy was able to improve the stability furthermore. Both algorithms were able to reach 1130 centimeters, which was the maximum distance available in the experimental setup. While the open-loop playback walk with advice operators was able to reach the limit only once, the learned policy was able to reach the limit several times.

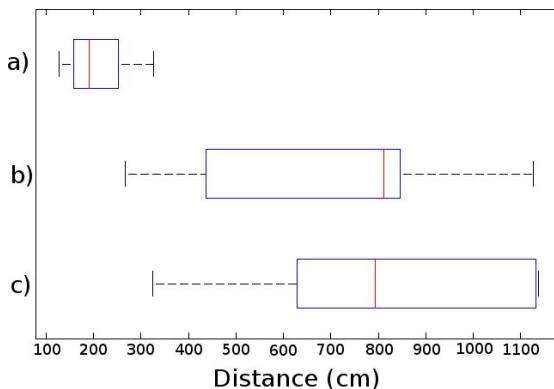


Fig. 9. Performance evaluation results: a) initial open-loop playback walk, b) open-loop playback walk improved using advice operators, c) closed-loop playback walk using learned correction policy

4 Conclusions

In this paper, we presented a method for learning a corrective policy for improving the walk stability on the Nao humanoid robot. Our method plays a single walk cycle obtained using an existing walk algorithm back to obtain an open-loop walk behavior and uses real-time corrective human demonstration in the form of single joint angle corrections delivered using Wiimote wireless game controllers to learn a correction policy for the open-loop playback walk. In the first part, we investigated the possibility of using a learned policy on a different walk algorithm and presented experimental results

showing that the learned policies do not depend on the underlying walk algorithm. In the second part, we proposed an extension over the initial version where the corrective feedback signals are in the form of foot position displacements and the sensory readings recorded at the time of correction signal are then used to learn a mapping from the sensory readings to a corresponding correction value. We also introduced an offline improvement using advice operators to improve the stability of the open-loop walk cycle. We presented experimental results demonstrating the learned policy outperforms the initial open-loop and improved open-loop using advice operators.

Addressing the delay between the perception and the actuation of the demonstrator, generalizing the proposed three-phase approach to a multi-phase learning framework applicable to other skill learning problems, investigating better policy derivation methods, improving the demonstration interface usability, relaxing the flat surface assumption to cope with uneven terrain, and extending the balance maintenance capability to endure against moderate pushes in adversarial domains (i.e., the robot soccer) are among the issues we aim to address in the future.

Acknowledgments

The authors would like to thank Stephanie Rosenthal, Tekin Meriçli, and Brian Coltin for their valuable feedback on the paper. We further thank the Cerberus team for their debugging system, and the CMWrEagle team for their ZMP-based robot walk. The first author is supported by The Scientific and Technological Research Council of Turkey Programme 2214.

References

1. Graf, C., Härtl, A., Röfer, T., Laue, T.: A robust closed-loop gait for the standard platform league humanoid. In: Zhou, C., Pagello, E., Menegatti, E., Behnke, S., Röfer, T. (eds.) Proceedings of the Fourth Workshop on Humanoid Soccer Robots in conjunction with the 2009 IEEE-RAS International Conference on Humanoid Robots, Paris, France, pp. 30–37 (2009)
2. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.-H.: B-Human 2009 Team Report. Technical report, DFKI Lab and University of Bremen, Bremen, Germany (2009), http://www.b-human.de/download.php?file=coderelease09_doc
3. Liu, J., Chen, X., Veloso, M.: Simplified Walking: A New Way to Generate Flexible Biped Patterns. In: Tosun, O., Akin, H.L., Tokhi, M.O., Virk, G.S. (eds.) Proceedings of the Twelfth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2009), Istanbul, Turkey, September 9-11. World Scientific, Singapore (2009)
4. Liu, J., Veloso, M.: Online ZMP Sampling Search for Biped Walking Planning. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), Nice, France (September 2008)
5. Gokce, B., Akin, H.L.: Parameter optimization of a signal-based biped locomotion approach using evolutionary strategies. In: Tosun, O., Akin, H.L., Tokhi, M.O., Virk, G.S. (eds.) Proceedings of the Twelfth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2009), Istanbul, Turkey, September 9-11. World Scientific, Singapore (2009)

6. Strom, J., Slavov, G., Chown, E.: Omnidirectional walking using ZMP and preview control for the NAO humanoid robot. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 378–389. Springer, Heidelberg (2010)
7. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems* 57, 839–845 (2009)
8. Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., Kawato, M.: Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems* 47 (2-3), 79–91 (2004); *Robot Learning from Demonstration*
9. Grollman, D.H., Jenkins, O.C.: Dogged learning for robots. In: International Conference on Robotics and Automation (ICRA 2007), Rome, Italy, pp. 2483–2488 (April 2007)
10. Grollman, D.H., Jenkins, O.C.: Learning elements of robot soccer from demonstration. In: International Conference on Development and Learning (ICDL 2007), London, England, pp. 276–281 (July 2007)
11. Argall, B., Browning, B., Veloso, M.: Learning from demonstration with the critique of a human teacher. In: Second Annual Conference on Human-Robot Interactions (HRI 2007) (2007)
12. Argall, B., Browning, B., Veloso, M.: Learning robot motion control with demonstration and advice-operators. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008) (2008)
13. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34 (2009)
14. Chernova, S., Veloso, M.: Teaching collaborative multirobot tasks through demonstration. In: Proceedings of AAMAS 2008, the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems, Estoril, Portugal (May 2008)
15. Meriçli, Ç., Veloso, M.: Biped walk learning through playback and corrective demonstration. In: AAAI 2010: Twenty-Fourth Conference on Artificial Intelligence (2010)
16. Nintendo. Nintendo - Wii Game Controllers (2007),
<http://www.nintendo.com/wii/what/controllers>
17. Atkeson, C., Moore, A., Schaal, S.: Locally weighted learning. *AI Review* 11, 11–73 (1997)

A Review of Shape Memory Alloy Actuators in Robotics

Mohammad Mahdi Kheirikhah, Samaneh Rabiee, and Mohammad Ehsan Edalat

Islamic Azad University, Qazvin Branch, Faculty of Industrial & Mechanical Engineering,
Nokhbegan Boulevard, Qazvin, Iran

kheirikhah@qiau.ac.ir, rabiee1982@yahoo.com,
m_ehsan_e@yahoo.com

Abstract. Shape Memory Alloys (SMAs) have been used for a wide variety of applications in various fields such as robotics. If these materials subjected to an appropriate thermomechanical process, they have ability to return to their initial shape. Often, they are used as actuators in robotic applications. The purpose of this paper is to present a brief review of literatures which using SMA in different robots' structure. First an introduction about shape memory effect of these materials will present. Then an assessment of done researches in application of these materials in robots' structure will accomplish and is devoted to the following area of robotics: Crawler, jumper, flower, fish, walker, medical and Biomimetic robotic hand.

Keywords: Shape Memory Alloys, robotic, review, actuator.

1 Introduction

SMA constitute a group of metallic materials with the ability to recover a previously defined length or shape when stand under a thermodynamics process.

Although a relatively wide variety of alloys present the shape memory effect (SME), but only those that can recover from a large amount of strain or generate an expressive restitution force are of commercial interest [1]. The typical attribute of these materials is that their parameters can't be determined by simple interpolation of properties of alloys included in the compound. Superplasticity, superelasticity, acid resistance and SME make these compounds technically important [2].

It has been found that many materials exhibit the shape memory effect (SME), such as Cu-Zn, Cu-Zn-Se, Fe-Mn-Se, Au-Cd, etc. The most common SMA is a nickel-titanium alloy known as NiTiNOL. This SMA is believed to be one of the most important candidates for smart materials [3]. SMA based on Ni-Ti are the alloys most frequently used in commercial applications because they combine good mechanical properties with shape memory effect[1].

In 1938, Greninger and Mooradian first observed the SME for copper-zinc alloys (Cu-Zn) and copper-tin alloys (Cu-Sn). Yet nearly 30 years elapsed until Buehler and his colleagues applied in 1965 for the first patent for a NiTiNOL, from the Naval Ordnance Laboratory. Near the end of the 1960s, Raychem developed the first industrial SMA applications in aeronautics with the Cryofit connector for F-14 airplane hydraulic circuits. Meanwhile at the University of Iowa, Andreasen's interest in dental alloys led

to the implantation of the first superelastic dental braces made from Ni-Ti in 1975. Buehler suggested using SMA in dentistry for different implants that could retain the shape memory [4].

The transition from one form of crystalline structure to another creates the mechanism by which the shape change occurs in SMAs. This change involves transition from a monoclinic crystal form (martensite) to an ordered cubic crystal form (austenite)[5]. The austenite phase is stable at high temperature, and the martensite is stable at lower temperatures. So typical SMA, the phase transformation between the two phases, martensite and austenite, is accompanied by variations in its resistivity [6]. The particular mechanical behavior of these materials, related to the existence of a structural phase transformation, allows their use as actuator in many applications such as: aerospace, instrumentation, robotics, biomaterials (medical prostheses or Microsystems) and the other application [7].

SMAs are usually available in the form of a wire, pipes, springs or ribbons. So, it can be used as a low-volume actuator in low-space where it is not possible to use huge actuator. In robotics, the SMAs represent a very interesting alternative within the field of classical drives, such as the electric or hydraulic motors. Thus the drives based on metals with the shape memory effect are the subject of research in many institutions, which are interested in the research of robotics. In this paper, there are presented an assessment of application of SMA actuators in robotics' different branches such as crawler robots, jumper robots flower robots, fish robots, walker robots, medical robots and Bio-mimetic robotic hand.

2 Crawler Robot

Liu and Liao [8] presented the development and testing of a snake robot that uses SMAs as actuators in 2004. An eight-segment robot was designed to move similar to the rectilinear motion of a natural snake. A pair of SMA wires had been implemented into each segment. One of the SMA wires in each segment was heated at a time, and it acted like a muscle to change the shape of the segment. A prototype robot was built, and it could move well with the desired locomotion. As shown in Fig. 1, when the lower wire is activated, the distance between its two ends is extended to 4 cm, and the distance between its two ends of the upper wire is shorten to 2 cm.

In the same year, Lee et al [9] introduced a novel bio-mimetic micro robot with simple mechanism using SMA to generate earthworm-like locomotion. A two-way linear actuator using SMA and silicone bellows had been applied to the micro

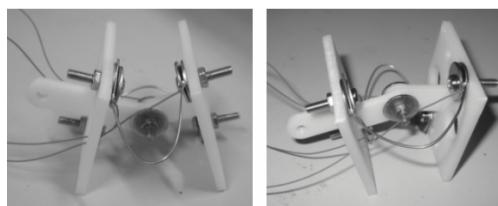


Fig. 1. Segments with activated SMA wires [8]

robot. Fig. 2 shows the locomotive principle of the proposed micro robot. The front needles clamp a contact surface and the rear body slides forward when SMA spring is contracted by heating. After the contraction of the SMA spring, the deformation energy of the silicone bellows makes the SMA spring elongate when it cools. At that time, the rear needles clamp the contact surface and the front body slides forward. Finally, the bellows' spring force is equal to that of SMA spring as initial equilibrium state.

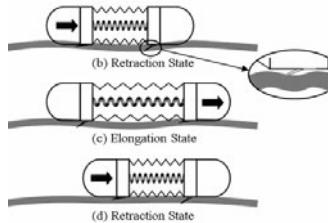


Fig. 2. Principle of locomotion [9]

The undulatory locomotion of living earthworms had been investigated deeply from the biological point of view by Menciassi et al [10] in 2004, but attempts of replication of earthworm models in real size were limited. Their robot had some modules that each module was actuated by one or more SMA springs. Preliminary tests demonstrated that the earthworm prototypes can move with a speed of 0.22 mm/s, thus approximating the behavior of biological earthworms.

Also in 2004, by investigating the biological field, the Menciassi et al [11] developed artificial earthworms by mimicking the structures and locomotion principles of real ones. Prototypes with or without micro-legs (which affect the locomotion performance) had been developed.

Along them, in 2004 Qin et al [12] that presented the design of a SMA driving micro-wheeled-robot, which had a 45mm×15mm×30mm size. Their paper shows the design principles of a resilient-rigid coupling SMA actuator (RRSA). This robot was controlled by Yao et al [13] in the same year. In this robot the SMA spring was heated by pulse current by PIC controller (a kind of single-chip) and was cooled by air.

Pipe inspection is a very important issue in construction. The inspection of low diameter canalizations is a pending issue nowadays, however it would help to repair and maintain a large amount of installations. Conventional in-pipe moving mechanisms for pipe inspection, driven by electromagnetic motors, have large volume and mass. The SMA actuator can be an alternative for a small-sized in-pipe moving mechanism due to its great power-to-weight ratio and simple structure. In 2005, Gambao et al [14] presented a robot that was able to move inside pipes of less than 26mm diameter and negotiate bends while carrying a camera to make an efficient in-pipe search. Each module of this robot had three degrees of freedom (DOF) with 3 SMA wires.

In year 2005, using twelve novel SMA linear actuators, which could stretch and shrink along its axis and could bear some radial force and bend, a miniature worming robot with a cubic form was presented by Yu et al [15]. Besides, the robot had a certain ability of passing through pipelines with large curvature or variable radius in a certain span.

Menciassi et al [16] expanded the development of segmented artificial crawlers with passive hook-shaped frictional microstructures in 2006. There were described the mechanical model, the design and the fabrication of a SMA-actuated segmented microrobot, whose locomotion had been inspired by the peristaltic motion of Annelids, and in particular of earthworms (*Lumbricus Terrestris*).

Caterpillars are some of the most successful scansorial and burrowing animals and yet they lack a hard skeleton. Their hydrostatic body and prolegs provide astonishing fault-tolerant maneuverability and powerful, stable, passive attachment. Trimmer et al [17] in 2006 described some of the biomechanics of caterpillar locomotion and gripper. They described their recent work to build a multifunctional robotic climbing machine based on the biomechanics and neural control system. This robotic caterpillar under development was a contoured cylinder constructed from highly elastic silicone rubber. It moved by SMA springs as actuators, bonded directly to the inside of the body wall. Also a kind of gripper was designed for this robot that provided its vertical movements by 3 SMA springs.

In the next year, 2007, also an investigation on the structure of caterpillar's body was presented by Kate et al [18] where in each segments of this robot there were used 2 SMA wires as an actuator too.

Spring type SMA actuators are selected to fabricate an inchworm-like moving mechanism that consists of clamping and moving modules (Fig.3). For selection of proper operating type (a bias type or a differential type) for clamping module and moving module, displacements and dynamic characteristics of each operating type were investigated by Lee and Kim [19] in 2008. A moving speed of 34 mm/min and traction force of 0.4N were obtained from the driving experiment in a pipe with the diameter of 39mm.

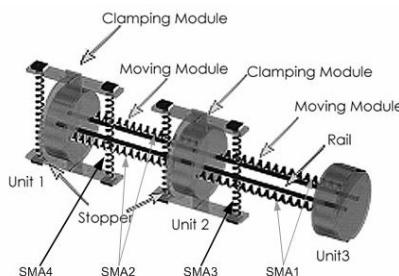


Fig. 3. Structure of in-pipe moving mechanism [19]

3 Jumper Robots

Locomotion over rough terrain has been achieved mainly by rigid body systems including crawlers and leg mechanisms. Sugiyama and Hirai [20] presented an alternative method of moving over rough terrain, one that was employed deformation in 2004.

The circular soft robot consisted of a circular elastic shell with a set of soft actuators inside. The robot has eight SMA actuators which deforms the robot body. Fig. 4 shows a sequence of snapshots of the prototype jumping. The prototype can jump a distance of 80mm, which is twice its diameter.

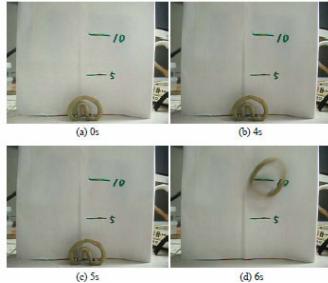


Fig. 4. Circular soft robot jumping [20]

In 2005, the principle of crawling and jumping as performed through deformation of a robot body was described by Sugiyama et al [21]. Then, in a physical simulation, they investigated the feasibility of the approach. Next, authors showed experimentally that prototypes of a circular robot and a spherical robot could crawl and jump.

In the same year, Shioitsu et al [22] created two prototypes of a spherical soft robot to assess experimentally the feasibility of a deformable robot crawling and jumping. Prototype A was for crawling and consisted of 18 SMA coils and shells made of spring steel. The core inside the spherical body included circuits to drive SMA coils, a microprocessor, and a serial communication circuit. Prototype B was for both crawling and jumping. This prototype consisted of 22 SMA coils – 18 for crawling and jumping and 4 for jumping. Circuits to drive SMA coils and a microcomputer were outside of the prototype.

4 Flower Robots

As a smart home service robot, the flower robot has various intelligent functions, such as moving mechanism, sensing ability, and home appliance functions. Especially, the moving function is very important function among the various function. The moving function of flower robot has consisted of the blooming of flower, the swaying of the stem and the stirring of the leaves in the wind.

Huang et al [23] in 2007 focused on the swaying of the stem structure. As an actuator for stem structure, authors adopted coil type SMA and then proposed silicone stem structure with 3 coil type SMA. They designed and fabricated the stem structure with 8mm of diameter and 50mm of length.

After a year, in 2008, Huang et al [24], focused on the movement of the flower robot structure. As an actuator for flower structure, they adopted coil type SMA and proposed silicone stem, petal and leafage structure used 6 coil type SMA. There were embedded 3 coil type SMA actuators in a silica gel rod. In the center hole of the



Fig. 5. The movement of stem and flower petal [24]

silicone rod, a spring and a wire with one side of SMA were connected to drive the petal of the flower robot. In the design of leafage, a SMA wire was bonded to one surface of the leafage. The fashion of motion in leafage was same with the stem structure. Fig.5 shows the movement of stem and flower petal.

5 Fish Robots

Fishes are regarded as highly maneuverable and effortless swimmers due to their extraordinary capability of moving in aquatic environs in terms of noiseless locomotion, rapid turning, fast starting and long distance cruise. Fish robots are inspired from natural fishes. In some of them, SMA is used as an actuator.

In 2006, Tao et al [25] presented the design of a caudal peduncle actuator, which was able to furnish a thrust for swimming of a robotic fish. The caudal peduncle actuator was based on concepts of ferromagnetic shape memory alloy (FSMA) composite and hybrid mechanism that could provide a fast response and a strong thrust. The caudal peduncle actuator was inspired by Scomber Scombrus. The morphology of an average size Scomber Scombrus was investigated, and a 1:1 scale caudal peduncle actuator prototype was modeled and fabricated. The tail was a composite consisted of superelastic NiTi SMA framework and two polymer impermeable membranes, which were adhered mutually by a superglue.

After two years, in 2008, Cho et al [26] presented the design and fabrication of a centimeter-scale propulsion system for a robotic fish. SMA spring actuators were customized to provide the necessary work output for the microrobotic fish. The flexure joints, electrical wiring and attachment pads for SMA actuators were all embedded in a single layer of copper laminated polymer film. Finally this package was sandwiched between two layers of glass fiber. Instead of using individual actuators to rotate each joint, each actuator rotated all the joints to a certain mode shape and undulatory motion was created by a timed sequence of these mode shapes.

During this year, an embedded SMA wire actuated bio-mimetic fin was presented by Wang et al [27]. Fish swims by their undulating body and/or fins. To simplify engineering modeling, the undulating swimming movement was assumed to be the integration of the movements of many flexible bending segments connected in parallel or in series. The musculature of a cuttlefish fin was investigated to aid the design of the bio-mimetic fin. SMA wires acted as muscle fibers to drive the bio-mimetic fin just

like the transverse muscles of the cuttlefish fin. During the bending phase, elastic energy was stored in the elastic substrate and skin, and during the return phase, it was released to power the return movement.

6 Walker Robots

Walker robots are a kind of robots which can move with some legs and something like them. They can be in various areas like a gecko, hexapods and so many other kinds of bio-mimetic robots.

The design, fabrication, and evaluation of two novel bio-inspired climbing robots were presented by Menon and Sitti [28] in 2005. Both were inspired by the locomotion of Geckos, a highly skilled natural climber. There were developed for terrestrial and extra-terrestrial environments. The first relatively large robot actuated by conventional motors was designed to operate at both in Earth and space scenarios. The second robot, whose motion was controlled using SMA actuators and size could be miniaturized to few centimeters scale, was designed for terrestrial applications. Preliminary prototypes of these robots were developed. One of these robots had a composite material frame and SMA wires provided motion that mimicked gecko muscles (Fig. 6). Also in the other paper, these authors [29] presented some other kinds of analysis of this gecko robot.

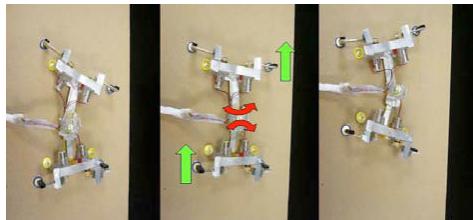


Fig. 6. Locomotion of Rigid Gecko Robot (RGR)[28]

In 2006 Nishida et al [30] presented a study on the development and control of a micro biped walking robot using SMAs. They proposed a flexible flat plate (FFP) consisted of a polyethylene plate and SMAs. Based on a detailed investigation of the properties of the SMA-based FFP structure, they developed a lightweight micro walking robot incorporating multiple SMA-based FFPs. This walking robot had four DOF and was controlled by switching the on-off current signals to the SMA-based FFPs.

Hoover et al [31] presented the design, fabrication and testing of a novel hexapedal walking millirobot using only two actuators in 2008. Actuation was provided by SMA wire. First, SMA was integrated very nicely with the smart composite microstructures (SCM) process. Alignment and routing features could be directly achieved into the composites to enable easy attachment and tensioning of the SMA wire. On the Second, SMA wire provided a force in a single direction.

In the same year, Berry and Garcia [32] presented the design and testing of a SMA and spring steel flexure actuator to use in a meso-scale 18 DOF hexapod. In the field of walking robots, the use of SMA as an actuator is mainly limited to micro-scale

applications, in which they considered robots measuring less than 5cm in any dimension micro-scale.

7 Medical Robots

One of the other application areas of SMAs is in medical robots. In 1999, Reynaerts et al [33] described a prototype gastro-intestinal intervention system based on an inch-worm-type of mobile robot which was a kind of vehicle for inspection through the colon. The robot consisted of three main modules that all these modules were actuated by SMA elements.

Also in 2001, Mi et al [34] presented the robot mainly consisted of soft mobile mechanism for earthworm locomotion and turning mechanism based on SME. The soft mobile mechanism contacted colon wall with air-in inflatable balloons, so the robot had better soft and non-invasive properties. The turning mechanism could be actively bent by SMA components. Therefore, this colonoscopic robot had good safety, lower working strength of surgeon and higher efficiency of colonoscopy.

The other kinds of these automotive tools and robots are Paris intestinal endoscope which were investigated by Cepolina and Michelini [35] in 2004. The Laboratoire de Robotique de Paris (L.R.P.) endoscope had been made by a series of modules articulated to each other by pin joints. On every link, two SMA springs had antagonist configuration to change the relative orientation. Although, there were investigated on Leuven intestinal worm and Pisa intestinal worm which used SMA as an actuator.

8 Bio-mimetic Robotic Hands

The ideology of this research is to utilize advanced actuators to design and develop innovative, lightweight, powerful, compact, and dexterous robotic technology. The key to satisfying these objectives is the use of advanced or smart materials, such as SMAs to power the joints of a prosthetic hand, and other dexterous robotic hands.

DeLaurentis and Mavroidis [36] in 2000 presented the mechanical design of a four fingered, fourteen DOF dexterous robotic hand, patterned human anatomy. A light-weight aluminum four DOF finger prototype had been made. The robotic hand concept was presented based on the using of SMA artificial muscles.

Also in 2002, DeLaurentis and Mavroidis [37] presented the mechanical design for a new five fingered, twenty DOF dexterous hand patterned human anatomy and actuated by SMA artificial muscles. Two experimental prototypes of a finger, one fabricated by traditional means and another fabricated by rapid prototyping techniques, were described and used to evaluate the design. The using of SMA actuators combined with the rapid fabrication of the non-assembly type hand to reduce considerably its weight and fabrication time.

After two years, Hino and Maeno [38] described a miniature robot finger which used SMA as the actuator. The miniature robot finger proposed in this paper, was driven by SMA wires. The structure of the robot finger imitated the musculo-skeletal system of humans, since SMA wires exhibited nonlinear features similar to human muscles.

Fig.7 show miniature five fingered robot hand was developed by Maeno and Hino [39] in 2006, for dexterous manipulation of small tissues and parts in medical and industrial fields. The size of this robot hand was about one third of human hands. It had 4 DOF per a finger that was almost the same as humans. The entire DOF of the hand was 20. The hand was driven by SMA wire actuator.

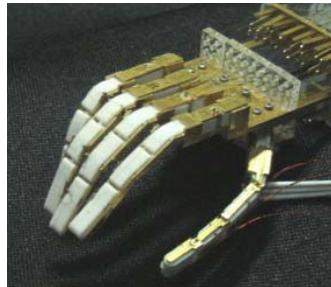


Fig. 7. Developed five-fingered robot hand [39]

Ashrafiuon et al [40] in the same year proposed a small planar 3 DOF robot arm actuated by two SMA actuators and a servomotor. The servomotor drove the first link while the SMA actuators rotated links 2 and 3. There was also an additional servomotor operating the gripper.

A finger spelling robot hand was also recently developed by Terauchi et al [41] which was driven by a combined action of a DC motor and SMA wires. The hand developed had 20 joints and DOF. The DC motor actuated the first joint (knuckle) of the finger. The second and third finger joints were directly driven by SMA wires. A pair of SMA wires was used in a push-pull configuration to actuate one finger joint.

Price [42] in 2007 presented SMA enabled devices promise to be of major importance in the future of dexterous robotics, and of prosthetics in particular. He investigated the design, instrumentation, and control issues surrounding the practical application of SMAs as artificial muscles in a three-fingered robot hand.

Also in the same year, O'Toole and McGrath [43] proposed mechanical design of a 12 DOF SMA actuated artificial hand. It was proposed that the SMA wires be embedded intrinsically within the hand structure which will allow for significant flexibility for use either as a prosthetic hand solution, or as part of a complete lower arm prosthetic solution.

A new bio-mimetic tendon-driven actuation system for prosthetic and wearable robotic hand applications was presented by Bundhoo et al [44] in 2008. It was based on the combination of compliant tendon cables and one-way SMA wires that form a set of agonist–antagonist artificial muscle pairs for the required flexion/extension or abduction/adduction of the finger joints. The performance of the proposed actuation system was demonstrated using a 4 DOF (three active and one passive) artificial finger tested, also developed based on a bio-mimetic design approach.

9 Conclusions

Based on the above discussions, the following conclusions can be drawn:

1. The SMA actuators can be an alternative for small robots due to their great power-to-weight ratio and ease of construction.
2. NiTinol alloys are compatible with human body, thus they can be used as either finger tendons or actuators in the medical robots.
3. The SMA actuators can be used in pipe inspection robots to move into the pipes and repair a large amount of installations.
4. One of most important applications of SMA actuators is in crawler robots such as snakes or earthworm robots.
5. The SMA actuators can be produced required loads for slowly or highly movement of some robots such as flower or fish robots.
6. Another application of SMA actuators is in walker robots such as gecko, hexapods and so many other kinds of bio-mimetic robots.

References

1. Machado, L.G., Savi, M.A.: Medical applications of shape memory alloys. *Brazilian Journal of Medical and Biological Research* 36, 683–691 (2003)
2. Vasina, M.: Untraditional Actuators For Robotics - Shape Memory Alloy. Dept. of Control and Instrumentation, FEEC, BUT
3. Chung, C.Y.: Applications of Shape Memory Alloys. Department of Physics and Materials Science, City University of Hong Kong
4. Mantovani, D.: Shape Memory Alloys: Properties and Biomedical Applications. *JOM* (Octobre 2000)
5. Wakjira, J.F.: The VT1 Shape Memory Alloy Heat Engine Design. Virginia Polytechnic Institute and State University, Copyright (2001)
6. Yee, H., Featherstone, R.: A new control system for fast motion control of SMA actuator wires. Dept. Systems Engineering, Research School of Information Sciences and Engineering, The Australian National University, Canberra, Australia (2004)
7. Hessissen, L., Essaaidi, M.: Thermomécanical Behavior of Polycrystalline Shape Memory Alloys in Martensitic Transformation. *European Journal of Scientific Research* 23(3), 474–481 (2008) ISSN 1450-216X
8. Liu, C.Y., Liao, W.H.: A Snake Robot Using Shape Memory Alloys. In: Proc. IEEE Int. Conf. on Robotics and Biomimetics, China (2004)
9. Young, P.L., Byungkyu, K., Moon, G.L., Jong-Oh, P.: Locomotive Mechanism Design and Fabrication of Biomimetic Micro Robot Using Shape Memory. In: Proc. IEEE Int. Conf. on Robotics & Automation, New Orleans (April 2004)
10. Menciassi, A., Gorini, S., Pernorio, G., Dario, P.: A SMA artificial earthworm. In: Proc. IEEE Int. Conf. on Robotics and Automation, New Orleans (2004)
11. Menciassi, A., Gorini, S., Pernorio, G., Weiting, L., Valvo, F., Dario, P.: Design, Fabrication and Performances of a Biomimetic Robotic Earthworm. In: Proc. IEEE Int. Conf. on Robotics and Biomimetics, China (2004)
12. Qin, C.J., Ma, P.S., Yao, Q.: A prototype micro-wheeled-robot using SMA actuator. Research Institute of Robotics, Mechanical Engineering, Shanghai JiaoTong University, Shanghai 200030, China (2004)

13. Yao, Q., Jin, S., Ma, P.S.: The Micro Trolley Based on SMA and its Control System. *Journal of Intelligent and Robotic Systems* 39, 199–208 (2004)
14. Gambah, E., Hernando, M., Brunete, A.: Multiconfigurable Inspection Robots for Low Diameter Canalizations. In: 22nd International Symposium on Automation and Robotics in Construction ISARC, Ferrara Italy, September 11-14 (2005)
15. Yu, H., Ma, P., Cao, C.: A Novel In-pipe Worming Robot Based in SMA. In: Proc. IEEE Int. Conf. on Mechatronics & Automation (2005)
16. Menciassi, A., Accoto, D., Gorini, S., Dario, P.: Development of a biomimetic miniature robotic crawler. *Auton. Robot.* 21, 155–163 (2006), doi:10.1007/s10514-006-7846-9
17. Timmer, B.A., Takesian, A.E., Sweet, B.M., Rogers, C.B., Hake, D., Rogers, J.: Caterpillar locomotion: A new model for soft-bodied climbing and burrowing robots. In: 7th International Symposium on Technology and Mine Problem, Monterey, CA, May 2-5 (2006)
18. Kate, M., Bettencourt, G., Marquis, J., Gerratt, A., Fallon, P., Kierstead, B., White, R., Trimmer, B.: SoftBot: A soft-material flexible robot based on caterpillar biomechanics. Tufts University, Medford, MA 02155
19. Lee, S., Kim, B.: Design parametric study based fabrication and evaluation of in-pipe moving mechanism using shape memory alloy actuators. *Journal of Mechanical Science and Technology* 22, 96–102 (2008)
20. Sugiyama, Y., Hirai, S.: Crawling and Jumping by a Deformable Robot. In: Proc. Int. Symp. on Experimental Robotics, Singapore (June 2004)
21. Sugiyama, Y., Shiotsu, A., Yamanaka, M., Hirai, S.: Circular/Spherical Robots for Crawling and Jumping. In: Proc. IEEE Int. Conf. on Robotics and Automation, Barcelona (April 2005)
22. Shiotsu, A., Yamanaka, M., Matsuyama, Y., Nakanishi, H., Hara, Y., Tsuboi, T., Iwade, T., Sugiyama, Y., Hirai, S.: Crawling and Jumping Soft Robot KOHARO. Dept. Robotics, Ritsumeikan Univ., Kusatsu, Shiga 525-8577, Japan (2005)
23. Huang, H.L., Park, S.H., Park, J.O., Yun, C.H.: Development of stem structure for flower robot using SMA actuators. In: IEEE Int. Conf. on Robotics and Biomimetics, ROBIO, December 15-18 (2007)
24. Huang, H.L., Park, S.H., Park, J.O.: Shape Memory Alloy based Flower Robot. In: 39th International Symposium on Robotics, Seoul, Korea (October 2008)
25. Tao, T., Liang, Y.C., Taya, M.: Bio-inspired Actuating System for Swimming Using Shape Memory Alloy Composites. *International Journal of Automation and Computing* 4, 366–373 (2006)
26. Cho, K.J., Hawkes, E., Quinn, C. Wood R.J.: Design, fabrication and analysis of a body-caudal fin propulsion system for a microrobotic. In: IEEE Int. Conf. on Robotics and Automation, ICRA, May 19-23 (2008)
27. Wang, Z., Hang, G., Wang, Y., Li, J., Du, W.: Embedded SMA wire actuated biomimetic fin: a module for biomimetic underwater propulsion. School of Mechatronics Engineering, Harbin Institute of Technology, PO Box 421, Harbin 150001, Heilongjiang, People's Republic of China (2008)
28. Menon, C., Sitti, M.: Biologically Inspired Adhesion based Surface Climbing Robots. In: IEEE Int. Conf. on Robotics and Automation Barcelona, Spain (April 2005)
29. Menon, C., Sitti, M.: A Biomimetic Climbing Robot Based on the Gecko. *Journal of Bionic Engineering* 3, 115–125 (2006)
30. Nishida, M., Tanaka, K., Wang, H.O.: Development and control of a micro biped walking robot using shape memory alloys. In: Proc. IEEE Int. Conf. on Robotics and Automation, May 15-19 (2006)
31. Hoover, M.A., Steltz, E., Fearing, R.S.: ROACH: An autonomous 2.4g crawling hexapod robot. University of California, Berkeley, CA 94720 USA
32. Berry, M., Garcia, E.: Bio-inspired shape memory alloy actuated hexapod robot. Laboratory for Intelligent Machines Systems. Cornell University, Ithaca, NY, USA 14853 (2008)

33. Reynaerts, D., Peirs, J., Van Brussel, H.: Shape memory micro-actuation for a gastro-intestinal intervention System. *Sensors and Actuators* (1999)
34. Mi, Z.N., Gong, Z.B., Qian, J.W., Mi, Z.W., Shen, L.Y.: Study on moving Principle of Colonoscopic Robot. *Journal of Shanghai University (English Edition)* 5(2), 143–146 (2001) ISSN1007-6417
35. Cepolina, F., Michelini, C.R.: Robots in medicine: A survey of in-body nursing aids. In: International Symposium on Robotics, France (March 2004)
36. De Laurentis, K.J., Mavroidis, C.: Development of A Shape Memory Alloy Actuated Robotic Hand. The State University of New Jersey, USA (2000)
37. DeLaurentis, K.J., Mavroidis, C.: Mechanical design of a shape memory alloy actuated prosthetic hand. *Technology and Health Care* 10 (2002)
38. Hino, T., Maeno, T.: Development of a Miniature Robot Finger with a Variable Stiffness Mechanism using Shape Memory Alloy. International Simposium on Robotics and Automation, Querétaro México, August 25-27 (2004)
39. Maeno, T., Hino, T.: Miniature five-fingered robot hand driven by sape memory alloy actuator. In: Proceedings of the 12th IASTED International Conference on Robotics and Applications, Honolulu, Hawaii, USA (August 2006)
40. Ashrafiou, H., Eshraghi, M., Elahinia, M.H.: Position Control of a Three-Link Shape Memory Alloy Actuated Robot. *Journal of Intelligent Material Systems and Structures* 17, 381 (2006) doi: 10.1177/1045389 X06056780
41. Terauchi, M., Zenba, K., Shimada, A., Fujita, M.: Controller Design on the Fingerspelling Robot Hand using Shape Memory Alloy. In: SICE-ICASE International Joint Conference, Busan, South Korea (October 2006)
42. Price, A.D., Jnifene, A., Naguib, H.E.: Design and control of a shape memory alloy based dexterous robot hand. *Smart Mater. Struct.* 16, 1401–1414 (2007)
43. O'Toole, K.T., McGrath, M.M.: Mechanical Design and Theoretical Analysis of a Four Fingered Prosthetic Hand Incorporating Embedded SMA Bundle Actuators. *Proceedings of World Academy of Science, Engineering and Technology* 25 (November 2007) ISSN 1307-6884
44. Bundhoo, V., Haslam, E., Birch, B., Park, E.J.: A shape memory alloy-based tendon-driven actuation system for biomimetic artificial fingers, part I: design and evaluation. In: *Robotica* ©. Cambridge University Press, Cambridge (2008) doi: 10.1017/S 026357470800458X

Biologically Inspired Mobile Robot Control Robust to Hardware Failures and Sensor Noise

Fabio DallaLibera^{1,3}, Shuhei Ikemoto², Takashi Minato³, Hiroshi Ishiguro^{3,4},
Emanuele Menegatti¹, and Enrico Pagello¹

¹ Department of Information Engineering, University of Padua, Padova, Italy

² Department of Adaptive Machine Systems, Osaka University, Osaka, Japan

³ ERATO, Japan Science and Technology Agency, Osaka, Japan

⁴ Department of Systems Innovation, Osaka University, Osaka, Japan

Abstract. Some bacteria present a movement which can be modeled as a biased random walk. Biased random walk can be used also for artificial creatures as a very simple and robust control policy for tasks like goal reaching. In this paper, we show how a very simple control law based on random walk is able to guide mobile robot equipped with an omnidirectional camera toward a target without any knowledge about the robot's actuators or about the robot's camera parameters. We verified, by several simulation experiments, the robustness of the random biased control law with respect to failures of robot's actuators or sensor damages. These damages are similar to the ones which can occur during a RoboCup match. The tests show that the optimal behavior is obtained using a bias which is roughly proportional to the random walk step, with a coefficient dependent on the physical structure of the robot, on its actuators and on its sensors after the damage. Finally, we validated the proposed approach with experiments in the real world with a wheeled robot performing a goal reaching task in a Middle-Size RoboCup field without any prior knowledge on the actuators and without any calibration of the very noisy omnidirectional camera mounted on the robot.

1 Introduction

When considering challenging environments like forests [1], planetary explorations [2] or game fields where the robots can collide to each other, it is almost impossible to predict in advance all the problems which could arise in the task execution and the possible failures the robot hardware might encounter. However, the current technology seems still to be lacking in providing reliable robots able to cope with hardware failure or uncertainties. Most techniques require a previous identification of the possible accidents that can occur to the robot and the design of specific workarounds for each of them. Some advanced self-modeling techniques were presented [3] but they request intensive computation to estimate the current status of the robot from the sensor information.

Conversely, very simple living beings like bacteria can cope with very complex and variable environments, and often present a highly adaptive and robust

behavior despite their structural simplicity. In particular, bacteria are able to sense the concentration of nutrients and direct their movements toward the food molecules while escaping from poisoning substances, without any complex planning strategy or fault detection system. This process is called *chemotaxis*.

The behavior of Escherichia Coli, in the following referred as E. Coli, has been deeply studied [4]. These organisms utilize a simple biased random walk for their movement. In particular, this bacterium has only two way of moving, rotating clockwise or counter-clockwise. When it rotates counter-clockwise the rotation aligns its flagella into a single rotating bundle and it swims in a straight line. Conversely clockwise rotations break the flagella bundle apart and the bacterium tumbles in place. The bacterium cannot therefore choose the direction of its movement, but just keeps alternating clockwise and counterclockwise rotations. In absence of chemical gradients the length of the straight line paths, i.e. the counter-clockwise rotations, is independent of the direction. The bacterium therefore essentially performs a random walk. In case of a positive gradient of attractants, like food, E. Coli instead reduces the tumbling frequency. In other terms, when the conditions are improving the bacterium proceeds in the same direction for a longer time. This simple strategy allows to bias the overall movement toward increasing concentrations of the attractant despite the simplicity of the mechanism and the difficulties in precisely sensing the gradient.

In this paper, we propose a bioinspired method that applies the same principle for robust mobile robot navigation. Actually this kind of behavior has already been applied for controlling a mobile robot [5,6]. Literature shows that while gradient descent is faster for tracking a single source, the biased random walk performs better in the presence of multiple and dissipative sources and noisy sensors and actuators. Demonstrations of the effectiveness of introducing a random term in the algorithm for preventing the robot from ending up in local minima are also provided.

However, the robustness to hardware damages and noisy sensory information achievable by biased random walk were not fully exploited. Expressly, in [6] the hardware already provides two basic movements, proceed straight and change direction randomly, and the biased random walk is performed at the behavior level. This approach limits the robustness for unexpected hardware failures. In fact, if due to hardware failures one of these basic movements does not operate as expected in many cases it will not be possible to accomplish the tasks. In our work, conversely, a biased random walk is executed directly in the *motor command space*, i.e. the behaviors themselves are determined online through the random walk. This gives great robustness in case of hardware failures since new behaviors that exploit the current hardware behavior are found online by biased random walk.

In general, performing a random walk in the motor command space allows to determine at runtime how to exploit the dynamics of the hardware, which can change due to hardware failures. With the approach presented in this work, therefore, there is no need to explicitly identify the failure and use preprogrammed alternative behaviors, which can be difficult to design beforehand [7]. We will

show that the proposed bioinspired method is able to provide high adaptability despite the extreme simplicity of the algorithm and the absence of any additional hardware to cope with failures. We would like to stress that other, more task specific techniques could probably be designed to achieve higher performance for the navigation task, but in this work we are interested in showing the biologically inspired approach of biased random walks is sufficient to achieve the task and that it is robust to hardware failures and sensor noise.

The remainder of the paper is organized as follows: Section 2 will present the details of our control algorithm. Section 3 reports simulation results indicating that a biased random walk in the space of the actuator signals is able to drive the robot toward the target even without any knowledge of the robot structure (e.g. the orientation of the wheels). We then show that the robot is able to reach the target in presence of several hardware damages such as obscuration of part of the camera image or variation of the size, change of the rotation axle or uncontrollability of one wheel.

Another interesting point analyzed in this work is the effect of varying the noise and bias amplitudes. If the bias is too strong the system keeps repeating the same behavior even for small positive gradients, wasting time executing very small condition improvements. Conversely, if the noise is too strong the system changes its behavior very often trying many inefficient behaviors. We can therefore expect to have an optimal balance between noise and bias (see Fig. 1).

Results show that in our settings the performance of the behavior is determined essentially solely by the ratio of the two quantities, with an optimal ratio value dependent on the hardware state. We then verified by practical experiments with a real robot that our approach is applicable to real world problems with noisy sensors and actuators, as highlighted in Section 4. We conclude by section 5 illustrating future works.

2 Control Algorithm

As stated in the introduction our approach takes inspiration from the chemotaxis of *E. Coli*, which can be interpreted under the very general framework of biological fluctuations [8]. Expressly assuming to have a continuous time system we can model it by the equation

$$\dot{u} = \alpha A(x)f(u) + \beta\eta. \quad (1)$$

where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the control signal. The function $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is a deterministic function and η is a random variable. These two quantities are multiplied by two constant scaling coefficients, $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$. The output values of f are multiplied by $A : \mathbb{R}^n \rightarrow \mathbb{R}$ which is a function of the state, called “activity”, that indicates the fitness, or “quality” of a particular condition.

Intuitively, when the conditions improve, the value of $A(x)$ increases and control becomes mainly deterministic (approximately $f(u)$) whereas, when the conditions worsen, the control becomes more and more stochastic. As a practical

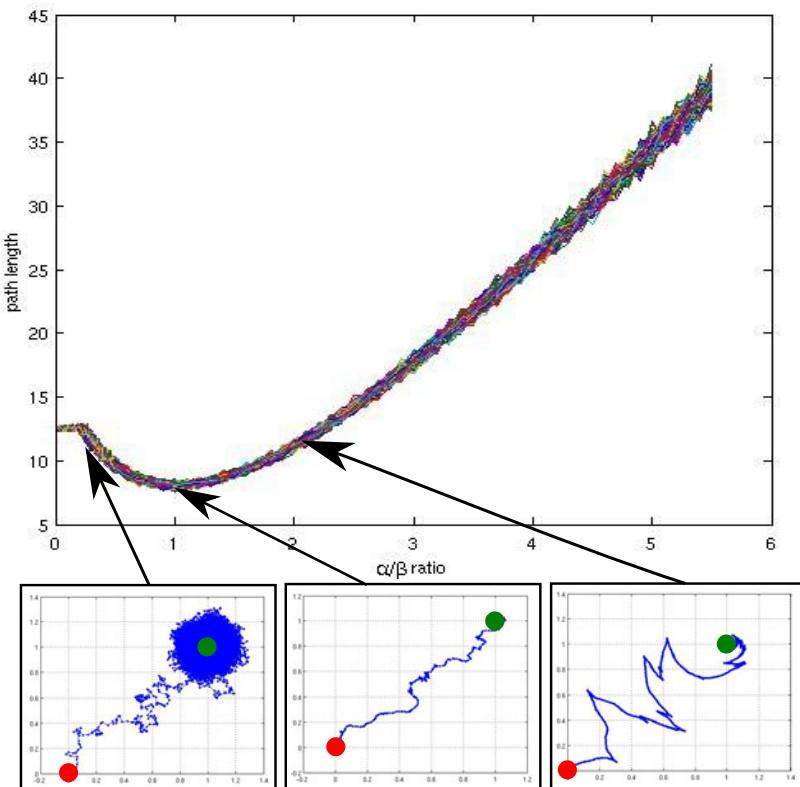


Fig. 1. Effect of the bias in random walk. The top figure plots the relationship between the bias to noise ratio and the path length, obtained by 100 tests. In the three graphs in the lower part of the figure the red circle denotes the start point and the green one denotes the target.

example in the E. Coli case, when the bacterium doesn't sense any food gradient it proceeds by a random walk, given by straight (deterministic) movements alternated by random changes in the direction due to tumbles. When the food concentration increases the frequency of tumbles is reduced and therefore the behavior becomes more and more deterministic.

This simple, biologically inspired framework revealed to be very robust and perform well in many applications of artificial systems, for instance for robot navigation, control of pneumatic actuated robotic arms or even routing in overlay networks [9].

While the robot presented in [6] has basic behaviors already implemented at a low level, we aim at having the system determining on-line also these motion primitives. A big advantage of this approach is that it can recover from unexpected and even quite serious damages in the robot hardware, giving the system

a high robustness without the need of introducing any self modeling or damage identification.

In other terms, using the approach presented in [6] when the robot finds good conditions it increases the frequency of “deterministic” commands, i.e. “go forward”. If due to a hardware failure, the behavior corresponding to such commands will be different from the expected one the task will not be accomplished. Imagine for instance to have a mobile robot with two wheels powered by independent motors and that due to an encoder problem one of the motors starts to rotate in the opposite direction. In this case when the “go forward” motor command is provided the robot spins around itself and the target will never be reached. With our approach, instead, the robot will explore new motor commands, until it finds that rotating the motors in opposite directions the distance from the target can be decreased.

In our experiments, the control signal u corresponds to the angular velocity of each motor and the state x corresponds to the information coming from the camera, i.e. the number of pixels whose color is similar to the target color.

To simplify as much as possible we decided to set $f(u)$ as follows:

$$f(u) = \frac{u}{\|u\|} \quad (2)$$

i.e. we maintain only the direction of u , and

$$A(x) = \text{sgn}\left(\frac{dx}{dt}\right) \quad (3)$$

where sgn is the sign function.

Concretely, we simply apply a bias in the current direction if we are following an increasing gradient, and bias in the opposite direction if the values of the activity is decreasing and no bias in case of a constant activity.

3 Robustness under Hardware Failures

Using ODE¹, we simulated a mobile robot equipped with three spherical wheels. The two front wheels are directly actuated by two independent motors whose maximum velocity is 0.5 rad/s while the rear wheel is free to rotate in any direction. The task is to reach a red hemisphere of radius 4 m placed at a distance of 30m. The robot is equipped with an omni-directional camera and the value of x fed to the controller is the number of red pixels in the image, determined by a filter that given the RGB components of each pixels counts the pixels whose R component is more than double the maximum of the G and B component values.

The controller receives information on the red pixels with a sampling frequency of 0.2Hz a value and provides a 2 dimensional velocity command u .

¹ Open Dynamics Engine, a free library for simulating rigid body dynamics. For details see <http://www.ode.org>

We chose to employ such a low sampling frequency to validate the robustness of the method even in case of low cost hardware with very poor performances. The controller implements the discrete time equivalent of equation 1, expressly

$$u_{t+1} = u_t + \alpha A(x) \frac{u_t}{\|u_t\|} + \beta \eta \quad (4)$$

where

$$A(x) = sgn(x_t - x_{t-1}) \quad (5)$$

We simulated four types of damages (see Figure 2):

1. the right wheel size is reduced to two thirds of its normal size
2. the right wheel becomes uncontrollable, i.e. its movement is completely random
3. the right wheel rotation axis direction is turned 90 degrees along the Z axis and becomes parallel to the longitudinal axis, i.e. the rotation of the wheel instead pushing the robot forward and backward pushes the robot to the left or to the right
4. 20% of the camera image becomes obscured

We assumed $\eta \sim \mathcal{N}(0, 1)$ as a Gaussian variable of variance one and studied the behavior for several values of α (the scaling coefficient of the bias) and β (the scaling coefficient of the noise). In particular for each condition (no damage or one of the damages listed) we determined the time spent contacting with the goal over 20000 seconds. Each condition is simulated one time for 128 different positions of the target, in particular assuming the robot's chassis is placed at $(0,0)$ we set the target in each of the positions as $(R \cdot \cos(\theta_i), R \cdot \sin(\theta_i))$, $\theta_i = \frac{i \cdot 2\pi}{128}$, $i = \{0, \dots, 127\}$ where $R = 30$ m.

The bias should be strong enough to drive the robot toward the target in short time but small enough not to keep performing the same action if the reduction of the distance to the goal is too little. We can expect therefore the existence of an optimal value of the bias. This intuitive idea can be easily exemplified simulating the movement of a point in the Cartesian plane. Expressly suppose a point to be initially at the origin and assume the presence of a target point, $(1, 1)$ in Fig. 1. Define the velocity as

$$v_{t+1} = \alpha \frac{v_t}{\|v_t\|} * sgn(\Delta) + \beta \mathcal{N}(0, 1) \quad (6)$$

where Δ is the variation of the distance to time t to time $t-1$. If the α/β ratio is too low, although the bias eventually drives the point in proximity of the target, much time is required to reach the goal because of unnecessary path deviations. Conversely, suppose to have a very high α/β ratio. Then, when the velocity becomes nearly tangential to the path that leads to the target, even though it is slightly in the direction of the target, the point keeps moving in that direction, which results in traveling long paths that only achieve a short reduction of the distance to the goal.

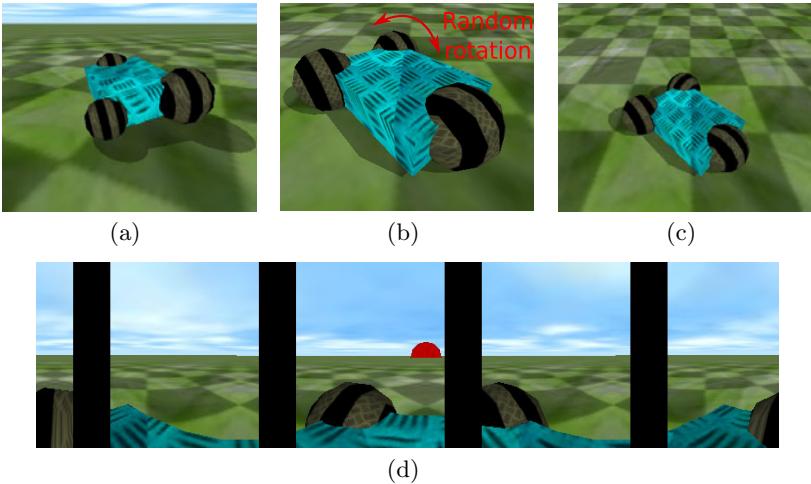


Fig. 2. The four damages simulated. (a) Reduced size of one wheel. (b) Uncontrollability of one wheel (c) Rotation of the rotation axis of one wheel (note the direction of the stripes) (d) Obscuration of part of the acquired image.

We can ask ourselves whether the optimal bias is constant or changes in case of hardware damages. The reaching experiment was therefore repeated for different settings of the values of α and β both for the undamaged robot and for the four robot conditions described beforehand.

Figure 3 depicts the results for the undamaged robot and for the four damages previously listed. The x and y axis indicate the values of α and β respectively, while the color represents the performance, in terms of ratio between the time spent touching the target and the total simulation time (20000 seconds). For all damages the graphs presents non zero values, i.e. the robot is able to reach the target and touch it. It is worth to notice that some damages, especially the rotation of the rotation axis of one wheel, completely changes the effect of motor commands. However, our simple algorithm is able to identify new efficient motor commands on the fly, without the need of any failure detection.

As expected, a completely deterministic behavior ($\beta = 0$) is often not able to drive the robot to the target, since, without “exploration” of motor commands done by the random part, the system can just provide a single type of motor command. Similarly when $\alpha = 0$ the probability of touching the goal with a completely random movement is so low that in no experiment the robot could reach the target within the simulation time.

We can see that the color zones are approximately triangles departing from the origin, i.e. the performance depends just on the ratio between α and β and not on their value.

Figure 3(f) shows the average performance for various $\frac{\alpha}{\beta}$ ratios. We notice that for the first type of damage (reduced wheel size) a ratio close to 2 gives the best performances, while in the case of changed rotation axis the best performance

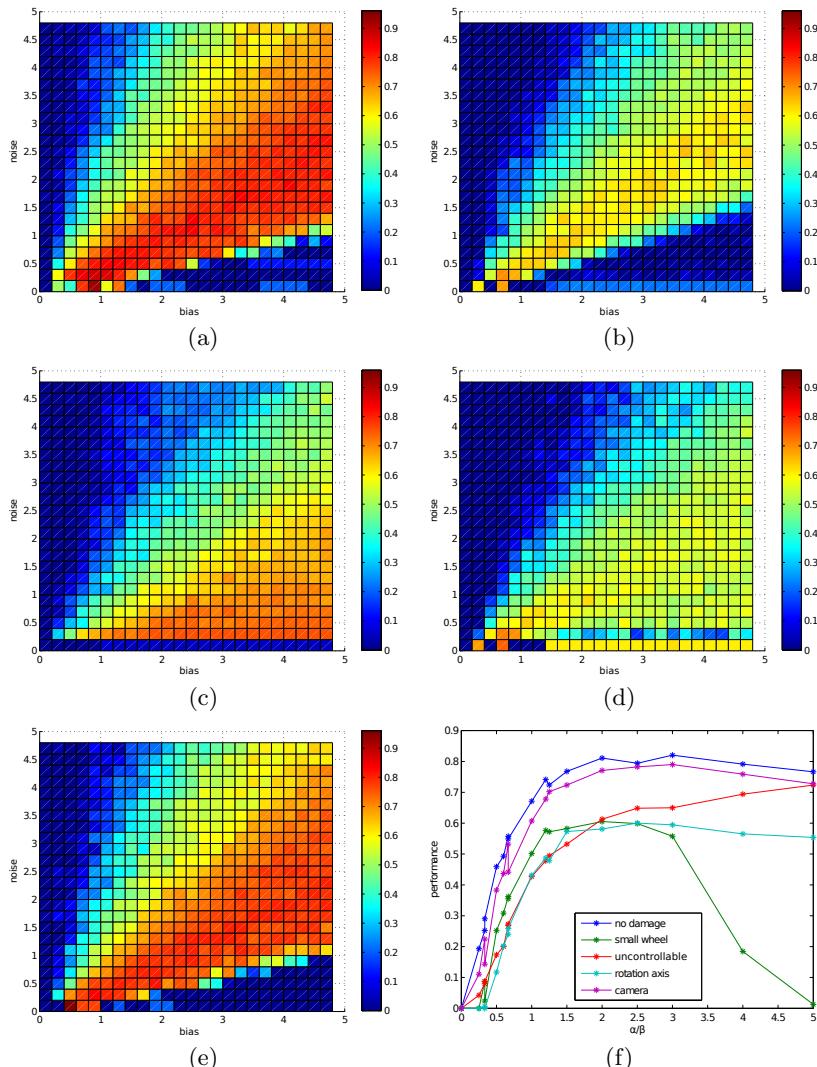


Fig. 3. Performances (ratio between the time spent touching the target and the total simulation time) of the robot for various settings of α and β : (a) Without any damage, (b) Reduced size of one wheel, (c) Uncontrollability of one wheel, (d) Rotation of the rotation axis of one wheel, (e) Obscuration of part of the image, (f) plot of α/β vs. average performance

is obtained with $\frac{\alpha}{\beta} \approx 2.5$. The undamaged robot and the robot with damaged camera instead performs best with $\frac{\alpha}{\beta} \approx 3$. For the uncontrollable wheel higher values for $\frac{\alpha}{\beta}$, around 5, gives the best performance. In this case, probably the noise introduced by the hardware itself reduces the noise required in the control signal.

Observing Fig. 3, we notice that some damages seems easier to recover than others, in detail the performance decreases more abruptly when the size of one wheel is reduced and when the rotation axis is changed by 90 degrees, than when the camera is partially obscured or when one wheel become uncontrollable. In these cases a lower $\frac{\alpha}{\beta}$ is more beneficial, i.e. intuitively speaking when the task is difficult it appears that the more stochastic the control is the better it is.

4 Robustness to Real World Noise

Often the effect of noise, environment uncertainties and modeling errors prevents algorithms to work in a real world setup. We validated the practical applicability of our algorithm by using a mobile robot, namely B12 by Real World Interface. The robot was equipped with a 640×480 Logitech webcam placed on an omnidirectional mirror and an off-the-shelf mobile PC. A red blanket placed on a chair was used as target, as visible in Fig. 4

As in the simulation experiment the control loop consists in acquiring the image, counting the number of the red pixel, performing the biased random

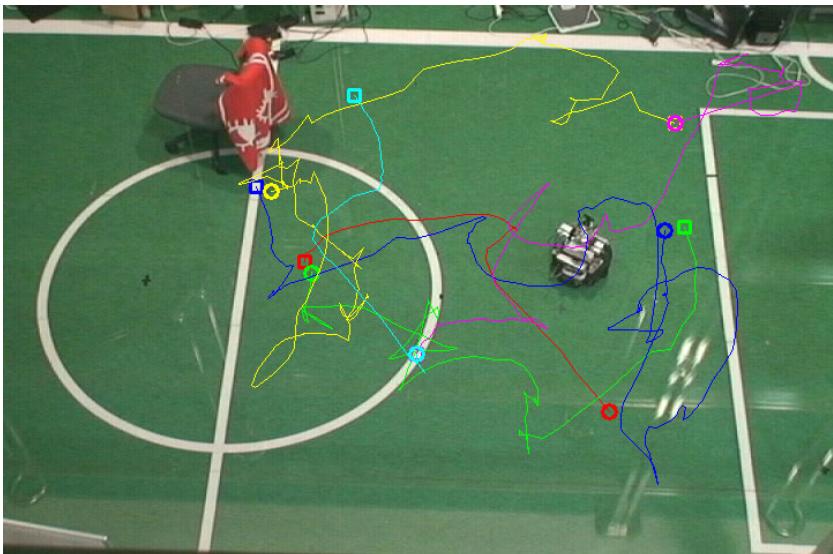


Fig. 4. Paths covered by the robot for six different initial positions and goal settings. Each path is drawn with a different color. Circles indicate starting points and squares indicate end points (for some subsequent paths the starting point is the ending point of the previous path.)

walk and setting the motor velocities. Once again to show the applicability of our approach even in case of very low-cost hardware, no particular hardware choice or software optimization were performed, leading to a quite slow and jittery control loop time, 861 ms with a standard deviation of 7.3 ms.

It is interesting to notice that the motor command space of the robot used in the experiment is quite different from the one of the simulated robot. Expressly, while the simulated robot has two driven-wheels independently actuated and a third rear castor wheel, the real robot is a Synchro Drive platform, in which one motor drives the wheels, while the other changes the wheel orientation. However, since our approach makes no assumption on the hardware, no change in the algorithm is required when passing from the simulation setup to the real robot experiment. The only parameter adjustment required was the α/β ratio. For the real world experiment, we adopted $\alpha = 0.8, \beta = 0.025$, determined experimentally with four trials to understand the order of magnitude.

The motors are equipped with encoders, which were used exclusively for data logging and reconstruction of the robot path. The short length of the paths covered by the robot in the experiments actually make the error accumulated by the encoders negligible. A validation was nonetheless performed using image processing on a video captured from the top of the experiment field.

Figure 4 shows some of the paths covered by the robots for different goal and robot initial position settings. Initial distance were set with values from 164 to 364 cm, and for all the 11 tests performed by the robot achieved target reaching,

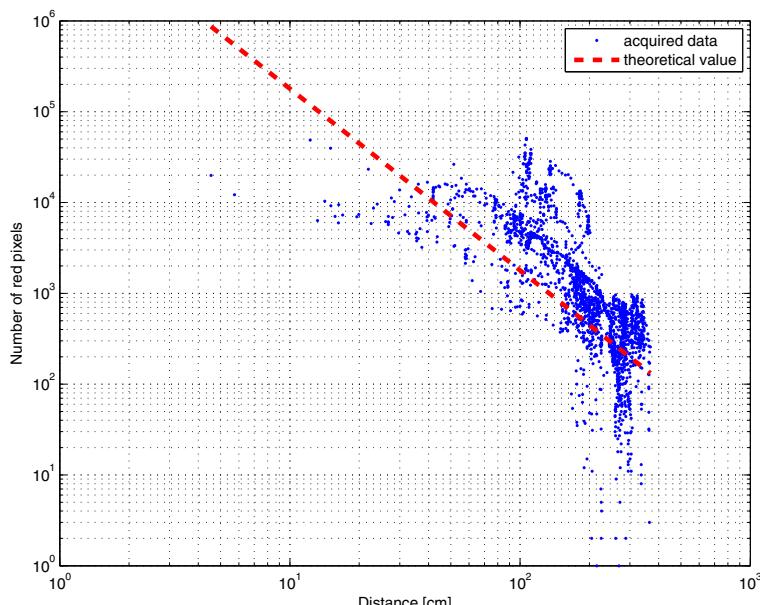


Fig. 5. Relationship between the distance from the goal and the number of red pixels measured during the experiments

with times ranging from 32 to 167 seconds. The measured path lengths were in average 3.38 times the optimal path. Notice that the performances could be improved by tuning the α/β ratio.

To illustrate the robustness to the real world noise Fig. 5 reports the relationship between the distance in a straight line to the goal, obtained off-line from the encoder values, and the number of red pixels measured during the robot motion. We can notice that, although the number of pixels is approximately a decreasing function of the distance, for similar distances we often have very different values for the number of pixel and conversely for a similar number of pixels the distances can be very different. Impressively, despite these strong uncertainties, in the actual target distance our simple algorithm was always able to drive the robot to the goal in a reasonably short time.

5 Conclusions and Future Works

In this paper, we presented an approach for mobile robot control based on biased random walk. The algorithm here proposed is actually biologically inspired, and in particular is based on the Escherichia Coli chemotaxis. These bacteria perform a random walk, but they bias their movement toward food by decreasing the number of random direction changes when the concentration of attractants increases.

We showed that biased random walk in the motor command space can be sufficient to drive a robot toward a goal without any knowledge of the robot structure. More importantly, since we make no assumptions on the actuators of the robot, in case of hardware failure the control is able to adapt to the new hardware conditions without the need of self modeling or failure identification.

We validated the methodology both by simulation and real world experiments. In detail, we showed that a mobile robot equipped with an omnidirectional camera is able to reach the goal even in case of severe damages to the sensors and actuators. The robot was hindered by damages similar to the ones which could occur during a Middle-Size RoboCup match. Our simple algorithm proved to be effective even in case of strong environmental noise, a big problem for all real world setups. We finally provided an experimental study on the optimal bias in the random walk, and showed that performances can be optimized by adjusting the ratio between α and β . The results suggest that the optimal bias is proportional to the random term with a coefficient dependent on the hardware. Initial tests seems to suggest that the bias should be higher in case of stronger damages, but this should be investigated more systematically with a clear definition of the “strength” or “hardness” of damages. Automatic estimation of the optimal bias will be tackled in future works. We notice, though, that even if the value is not optimal, the robot is still able to accomplish the task, i.e. choosing the optimal value of the ratio can increase the performances but suboptimal values are still sufficient to achieve the task. Future works will also aim at identifying whether other types of noise, like Levy walk, are more efficient than Gaussian Noise. The current experiments will finally be extended by the inclusion of fixed and moving obstacles to study the collision avoidance performance of our algorithm.

This paper solely aims at proving the feasibility and robustness of the very simple and biologically inspired approach of biased random walk in the motor command space. Obtaining high performance in terms of reaching time is out of scope. No performance comparisons were therefore performed with classical approaches like potential field navigation or reinforcement learning, for which we could actually expect better performances. However, we are able to notice that the underlying assumptions are different. In fact, when virtual force fields [10] approaches are used the results of the motor commands must be known a priori, while our approach does not assume any knowledge on the underlying actuators, as clearly appears by observing the difference in the structure of the simulated and real robot. Using reinforcement learning the ignorance on the motors can be overcome, but an appropriate and careful definition of states and rewards is required to allow successful learning.

References

1. Saiki, Y.M., Takeuchi, E., Tsubouchi, T.: Vehicle localization in outdoor woodland environments with sensor fault detection. In: ICRA, Pasadena, USA, pp. 449–454 (2008)
2. Plagemann, C., Fox, D., Burgard, W.: Efficient failure detection on mobile robots using particle filters with gaussian process proposals. In: IJCAI, Hyderabad, India, pp. 2185–2190 (2007)
3. Bongard, J., Zykov, V., Lipson, H.: Resilient machines through continuous self-modeling. *Science* 314(5802), 1118–1121 (2006)
4. Adler, J.: The sensing of chemicals by bacteria. *Scientific American* 234, 40–47 (1976)
5. Holland, O., Melhuish, C.: Some adaptive movements of animats with single symmetrical sensors (1996)
6. Dhariwal, A., Sukhatme, G.S., Requicha, A.A.G.: Bacterium-inspired robots for environmental monitoring. In: IEEE International Conference on Robotics and Automation (ICRA 2004), New Orleans, USA, pp. 1436–1443 (2004)
7. Scheutz, M., Kramer, J.: Reflection and reasoning mechanisms for failure detection and recovery in a distributed robotic architecture for complex robots. In: IEEE International Conference on Robotics and Automation (ICRA 2007), Roma, Italy, pp. 3699–3704 (2007)
8. Yanagida, T., Ueda, M., Murata, T., Esaki, S., Ishii, Y.: Brownian motion, fluctuation and life. *Biosystems* 88, 228–242 (2006)
9. Leibnitz, K., Wakamiya, N., Murata, M.: Biologically inspired self-adaptive multi-path routing in overlay networks. *Communications of the ACM* 49, 63–67 (2006)
10. Arkin, R.: Motor schema based navigation for a mobile robot: An approach to programming by behavior. In: IEEE International Conference on Robotics and Automation (ICRA 1987), Raleigh, USA, pp. 264–271 (1987)

TOPLEAGUE and BUNDESLIGA MANAGER: New Generation Online Soccer Games

Ubbo Visser

Department of Computer Science
University of Miami
1365 Memorial Drive
Coral Gables, FL, 33146, USA
visser@cs.miami.edu

Abstract. This paper describes a new generation online soccer manager. More than 210,000 users operate TOPLEAGUE and the OFFICIAL BUNDESLIGA MANAGER, complex real-time soccer simulators that are based on actual data of real professional soccer players. The underlying technology is a hierarchical three-tier multiagent system that consists of autonomous BDI agents and allows dynamic group structures (e.g. an emergent situation for a wing attack). The German Bundesliga, one of the most prestigious soccer leagues in the world, adopted this AI system for their official web site. The online games run seamlessly in a web browser with a state-of-the-art 3D visualization engine. Fundamental research within the domain of the RoboCup simulation league is the basis for this technology. We will describe the architecture of the multiagent system (MAS) in this paper, discuss motion capture techniques for graphical animation, and reveal details about user acceptance of the games.

Keywords: Robotic Entertainment, Teamwork and Heterogeneous Agents, Motion Capture, Facial Expressions, Soccer Simulation, Manager Games.

1 Introduction and Related Work

A vast amount of games have been developed for the mass market over the past decade. Not only can we observe more than ten types of games (e.g. adventure games, casual games, sports games [1,22]), we can also state that most of the games have been produced for specific hardware (e.g. consoles, game consoles, Wii consoles). Online games have also been developed in great numbers [10], however, web browser restrictions prevented them from being as complex - especially in their graphical animation - as their console counterparts. One reason for this is the computer's configuration (e.g. real-time rendering, older computers have limited rendering capabilities). Some hardware related features (e.g. vertex shaders) may not work on older machines. Another reason is security (applets for instance cannot change the users configuration without prior permission). Our research reveals that a significant number of users (approx. 10%) have not installed their graphic drivers properly.

Nevertheless, online games have become more and more complex, especially in the last few years. We see a merge between the technologies, e.g. consoles have Internet

access now and can download the latest updates while still having an advantage of a defined hardware. On the other hand, the market share of online games (and mobile applications) are increasing [14]. We can even argue for a paradigm shift as nowadays computers and network bandwidth reach a point where complex real-time online games are possible for the mass market.

The RoboCup simulation league offers a number of research targets. Among them are complex simulation [16,15], real-time situation estimation [4,28,34,12] and behavior of agents [11,35,21]. Starting off as a simulation team in the early years of this decade we generated the idea to combine the new game paradigm shift with autonomous robots and sports data from real professional soccer players. We have developed a simulation engine where the agents comply to weekly updated performance indicators of real players (e.g. accuracy of shooting, tackle performance) and act according to tactical preferences of the user who operates the game.



Fig. 1. Autonomous agents in the OFFICIAL BUNDESLIGA MANAGER

Game play. The general idea for the user is to pick a virtual version of a professional soccer team and act as a coach. Games follow the regular season and the official game days follow the game days of the professional leagues. The user adds soccer knowledge for the next game while choosing formation, tactics, and training sessions for his team. Each game is simulated and a newly developed and high quality 3D graphics engine allows the user to see the match in a web browser thereafter (cf. figure 1). The user coaches a real team (e.g. Bayern Munich) and has the same problems to solve than the real coach in the real world (e.g. players that are suspended in the real world are also suspended in the virtual world). The games have the same rhythm than the reality, however, TOPLEAGUE games will be played one day earlier.

The paper is organized as follows: the architecture including simulator and graphics engine of the game is presented in the next section. We then describe our animation

process discussing our latest research results. We include a section of user acceptance where we discuss feedback and give statistics of the users of this technology.

2 Architecture

All data are held in a relational database (MySQL). A number of modules as middleware have been developed ensuring the scalability of our game. A scheduler is responsible for the list of games that need to be evaluated (see figure 2a).

2.1 Simulator

The simulator is developed for a sports game in general. In this instance, however, it is used to simulate a soccer game. It consists of a soccer environment with the necessary agents (two teams, referee, substitute players). There is no physics involved in the game with one exception: the ball has an accurate model for gravity, air friction as well as bouncing effects on the ground and for the posts. The friction varies with weather conditions, a rainy playground has less friction for example. Each team has optimal vision and accurate information about the world. They can communicate to each other and have a 100% access to each others world model. Each team is organized in a hierarchy. Within the simulation, the team decides about the general strategy (e.g. offense play, wing attacks in general). The agents in the next level (stable team structures such as defense, middle field, forwards) balance out the given general strategy with their own goals (e.g. defense within an attack or distribution of the players within an attack). The lowest level consists of agents that act and react according to the given overall goals and the concrete play situation based on their personal skills.

All agents 'exist' within an agent environment (cf. figure 2b). The environment's changes are sensed by each agent with the help of sensors leading to a world model. A perception of the world induces the reflection model that updates the internal world

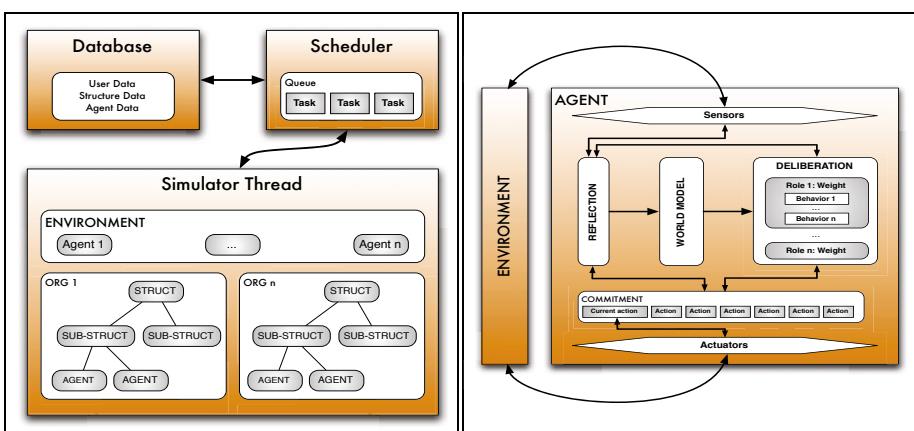


Fig. 2. a) Setup of MAS with MySQL DB and scheduler, b) Agent architecture

model of the agent with the actual perceptions and that validates the current action sequence taken by the agent (current commitment). A new deliberation cycle will be induced if the current action sequence is no longer valid (e.g. change of game situation). Within the deliberation cycle the various roles of an agent will be weighted based on the current situation evaluation. A role of an agent is directly linked to a list of behaviors, which can be used for the calculation of the action selection. A planning process is carried out that calculates the action sequence for the agent.

Input parameters. All real professional soccer players are grouped into five skill groups. The skills are calculated for all players of same position in same group based on Opta Sports Data statistics¹. After this, we are ranking players within one group. If players don't play due to sick leave or suspension, minimum skills will be taken for those players. Often, bad player skills can be observed (e.g. player haven't played or played but did not score or perform well). In order to rectify this situation we perform small changes in skills randomly.

A general input factor is a random seed for the pseudo random number generator. Team settings include but do not exhaust home team/away team, formation, and heterogeneous players. Tactics include team, shooting and passing behavior, side of attack, rules for ball holding and rules for tactical changes throughout the match. The user is also able to define certain roles such as captain, penalty scorer, free-kick and corner-kick player. Player settings includes the level of aggressiveness and deployment.

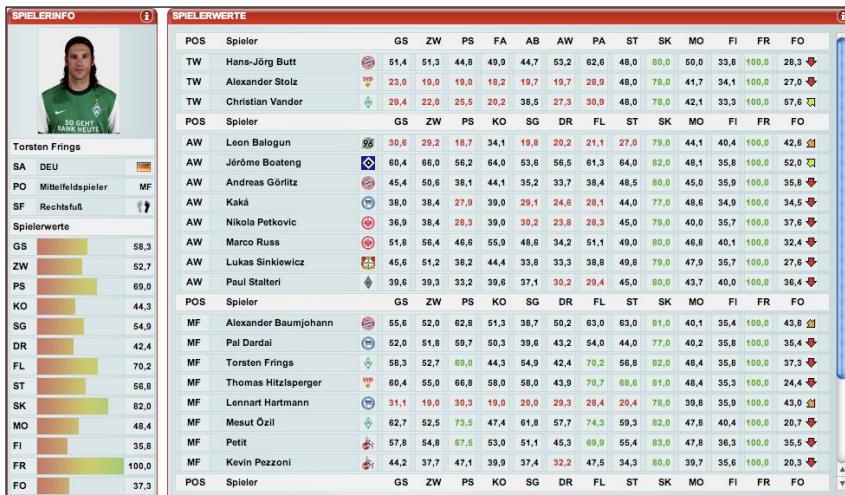


Fig. 3. Example for skills, GS is overall strength; ZW is tackle performance etc. The numbers are in %, different colors (or grey-colors) show skill classes, green (light grey) is a good classification, red (dark grey) is a negative classification.

¹ <http://www.optasportsdata.com>

The player attributes/skills (cf. figure 3) are based on Opta Sports Data. Thirteen skills are used either internally or externally. An example for internal skills that are only used inside our simulator is *penalty_kick*. An example for the an external skill that is also shown to the end user via web front-end is *duel_strength*. Here is the equation for the penalty kick:

$$\text{penalty_kick} = \left(\frac{p_{goal} + p_{tar}}{p_{miss} + p_{post} + p_{goal}} + \frac{rf_{goal} + rf_{post} + rf_{tar}}{rf_{tot}} + \frac{lf_{goal} + lf_{post} + lf_{tar}}{lf_{tot}} \right) / 3. \quad (1)$$

where p = penalty, tar = target, rf = right foot, lf = left foot, and tot =total.

2.2 3D Visualization Engine

The simulator produces a < 250K log-file that contains the positions of the players, the referees and the ball at each given time slot. The simulator runs with 10Hz. The log-file is then stored on one of our servers and can be requested from each user via web service. Technically, the log-file is requested by an applet on the client side that shows the game and various resolutions/options that are tailored for a variety of graphic cards.

All animation scenes are visualized in this graphics engine. It has been developed from scratch and guarantees high quality pictures in a web browser. The kernel is also domain independent and carries most of the features. Another module contains the 3D model for the stadium, scenes, textures, animations and interaction GUI for the soccer stadium. FaceIntegrator, developed in-house, is a tool that contains scenes and textures among 3D models. It also contains an implementation for the generation and integration of real soccer player images for our autonomous agents (cf. figure 8 on page 238).

The graphics engine contains features that include static and dynamic models, (hand-made) animations, skeletons (soon also bipeds), terrain description, object motions for animations, motion differences for animations, bezier curves, vertex-, fragment-, and geometry shader, potential fields, and some configuration files. A detailed description of features would be out of scope of this paper. However, we would like to point out two important features in this paper: the bone/joint system and the dynamic scene generation.

2.3 Quaternion Based Bone/Joint System

A quaternion-based bone/joint system allows a biunique interpolation of object rotations in 3D space. This is not possible within the Euler space because different object rotations can be combined in arbitrary orders and may end up in the same end-rotation position. Quaternions and their biunique interpolation characteristics allow us to fluently animate bone systems of dynamic objects with the help of a few key frames. An advantage is that we can animate objects at any given time independently from the frequency rates. The animated objects can be rendered in real time with software and hardware (shaders). Figure 4 shows our level-of-detail dependent on the use of the bone systems. Objects in the foreground show more bones as objects in the background. In this scene, players and banners are animated objects. Note that the level-of-detail does not show the hands of players in the background.

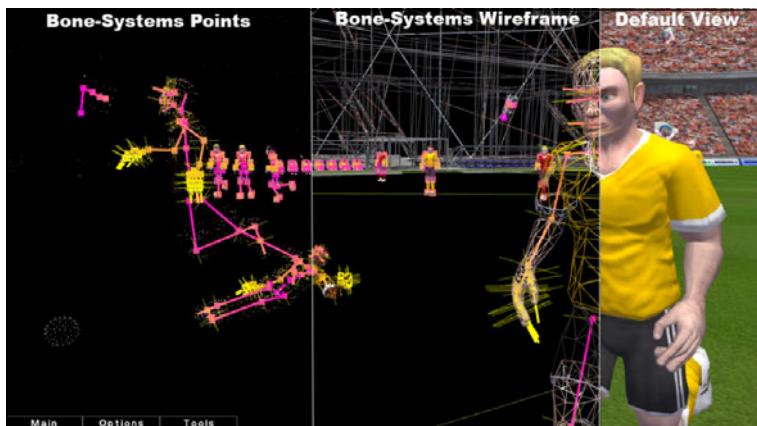


Fig. 4. Quaternion based bone/joint system with level-of-detail: the players in the background do not have as many bones as the players in the foreground

2.4 Dynamic Scenes

Users have the option to differentiate between various camera settings. All cameras can be used with a zoom. The cameras focus on an area close to the ball. In addition to the ball position in the game, more ball positions are estimated for potential future ball positions. This gives an illusion that the 'camera man' would take the scene live and would not know future ball positions. All ball positions are smoothed by cubic curves (Bezier). Using curves allows us a 3rd degree interpolation, which is significantly more dynamic than a 1st degree interpolation (linear interpolation). The soccer simulator marks special events such as goals, fouls, offside situation etc. These marks are the basis for a game situation evaluation which creates a suitable camera path. Figure 5 shows two different camera paths. The upper row shows a foul situation, the lower row a goal situation. Each game situation is unique and creates a unique camera path.



Fig. 5. Dynamic camera scenes: the upper row shows a foul situation, the lower row a goal situation

3 Motion Capture for Body Motions and Facial Expressions

Character animation is an important part of a graphical game engine. The restriction of an applet (or better: the client machine and the applet) does not allow us to create high-resolution models that we can see in console based games. Motion capture techniques are well-known and can be used for character animation.

There are many different approaches in the area of motion capture. Approaches contain non-optical systems such as inertial systems (inertial sensors, biomechanical models and sensor fusion) [33], electromechanical motion [31], and electromagnetic systems [17]. Other techniques are optical. Among these are semi-passive imperceptible markers [29], pure vision-based, markerless techniques [18,9] as well as active markers [37].

The most popular approach to motion capture today is to use reflective markers to the body and/or face of an actor, and track these markers in images acquired by multiple calibrated video cameras [32,13,30]. The marker tracks are then matched, and triangulation is used to reconstruct the corresponding position and velocity information.

Furukawa & Ponce [8] state that any motion capture system is limited by the temporal and spatial resolution of the cameras, and the number of reflective markers to be tracked. Matching becomes difficult with too many markers that all look alike or are getting lost while labeling. They state further that on the other hand, although relatively few (say, 50) markers may be sufficient to recover skeletal body configurations, thousands of markers may be needed to accurately recover the complex changes in the fold structure of cloth during body motions, or model subtle facial motions and skin deformations (see also [36]).

Our system is a mixture of hand-made animations and those that are made with the help of motion capture techniques (reflective markers). Our Motion Capture Lab is equipped with a modern Vicon System. It consist of ten 4-megapixel cameras with 120 Hz, two video cameras for ground truth, four force plates to measure forces on ground, facilities for facial expressions, and an electromagnetic device capture voltage in max 16 muscle groups. Figure 6 outlines the various steps (workflow) that have been performed to get a motion onto animated characters that can be used for a web browser game.

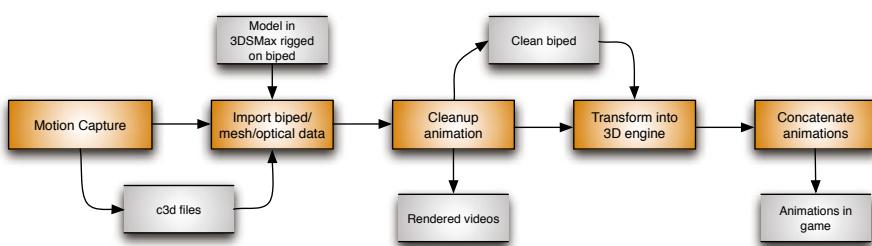


Fig. 6. Motion Capture Process from capturing to animation

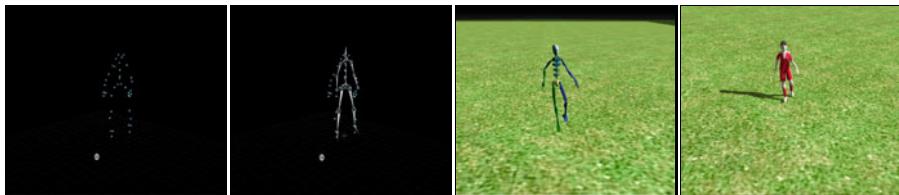


Fig. 7. Motion Capture Steps: a) raw markers in 3D space; b) data cleaned and assigned to bones; c) cleaned again and mapped onto biped; d) fully rendered video

3.1 Modeling Avatars

There are three main approaches to obtain facial models. Firstly, 3D scanning of faces. Face models can be complete representations of a polyhedron that is obtained with the help of the faces' surroundings scanning algorithm. These algorithms can use the viewing sphere and the perspective projection concepts (known as the K-M view space model) [20,19,3]. Secondly, available software products that come with pre-defined models such as FaceGen², Poser³ or DAZ-Studio⁴. FaceGen has been created using statistical modeling of the shape and appearance of human faces. FaceGen uses a similar approach to Blanz's [2], although the implementation differs in the details, and much additional work has been done to add features and allow integration in the character-creation pipeline. Poser offers eight ready-to-pose and textured humans of different ethnicities. Lastly, hand-made models with other software products such as Poly Modeling with 3D Studio Max or ZBrush. The basic idea here is to create characters from scratch using generic spheres from which polygons are used to create the desired shape.

We use existing software tools (e.g. FaceGen) and develop necessary tools (e.g. FaceIntegrator) ourselves (see figure 8 and 9 for the outcome).

3.2 Facial Expression Generation

Much work is being performed to animate believably avatars [24,5,27,25]. To generate facial expressions, the approach taken by many research group has been to use the well known Facial Action Coding system [7] to extrapolate different facial areas that are then independently activated according to predefined pattern [24,23,26]. These approaches address the difficult notion of expressing mixed emotions, but because it is based on Ekman's static encoding of the face, no specific emphasis is given to the dynamics of the facial expression generation. However, as pointed out in a study on communicating facial affect in computer-mediated communication and in affective computing applications, if there is a tradeoff to be chosen between showing highly realistic facial images (i.e. high spacial resolution, full color) at the expense of motion (e.g. low temporal resolution, lags), the motion should be retained as it represents the most important factor in communicating facial affect [6].

² <http://www.facegen.com>

³ <http://my.smithmicro.com/>

⁴ <http://www.daz3d.com>

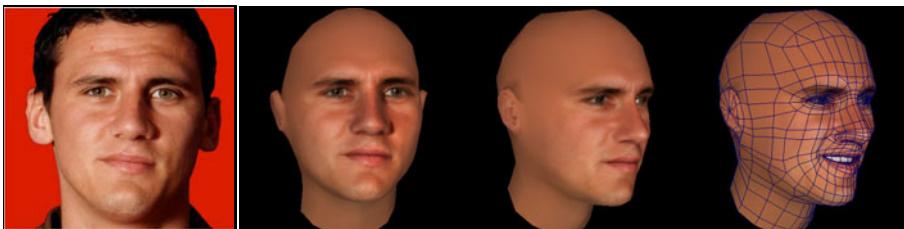


Fig. 8. A soccer player's picture projected on a morph with FaceGen



Fig. 9. Avatars with expressions of emotions within a browser applet

Morph target animation is a method that is used as an alternative to skeletal animation. Morph target animation is stored as a series of vertex positions which are moved to a different position in each keyframe.

Facegen has the ability to easily generate morphs of multiethnic avatars, which is an added advantage for our multi-cultural avatar dialog system. We generated a small number of dynamic facial expressions (see figure 9) using our simple player avatar model, following the dynamic principles for facial expression generation in humans.

4 Results and User Acceptance

This R&D project is ongoing for the last past couple of years. The game has been developed and is available online on www.bundesliga.de and www.topleague.de. We have currently more than 210,000 users that are signed up and play this game. Since launching in 2007, we have a steady and consistent growth of 500-1,200 users per week. Statistics also prove that a large part of users are coming back regularly. The activation level of our customers (i.e. unique user comes back at least once within that time frame) is

shown for 30 and 90 days. The current numbers regarding returning users (30/90) days are: OFFICIAL BUNDESLIGA MANAGER (20%, 30%), TOPLEAGUE (15%, 40%). These numbers are very competitive for the free market.

To date (April 26, 2010), we have simulated and delivered nearly 9 million unique games (8,746,843 to be exact). 31,5 million page impressions (PIs) have been created by users with 617,162 visits each month for the last 12 months in average. This is a ration of approximately 1/20 visits/PI. In contrast, a regular news web site has a ration of 1/7 or 1/8. This indicates that the average user is staying on our web site quite long.

I detailed analysis with Webalizer⁵ and Google Analytics⁶ reveals that the average time on site per user is > 12 minutes (time of analysis: Aug 1, 2009 - Nov 15, 2009, daily resolution). The bounce rate is below 8% and thus a very good rate according to Google Analytics (< 20%). We also have approximately 25-30% new visits which indicates that 70-75% are returning users.

User support is an integrated piece of the game. Support is realized with tutorials, context-sensitive help functions as well as user forums. In addition, we have several godfathers, users, that take over work for other users voluntarily. We also have a group of editors that monitor the discussions in the forums and give feedback to the development team.

5 Conclusion

We have described a R&D project that has its roots and the spirit from the RoboCup simulation league. We have developed an online game that is available on the free market. Nearly three years after launching we can state that the proof of concept has been successful. The technical key features of our technology can be identified as a unique AI simulation engine with an authenticity module allowing same player skills in virtual world than in the real world. This is a new paradigm on online sports games and can be seen as the closest soccer simulation game to reality. Another key feature is the visualization which is a 3D state-of-the-art graphics engine for a web browser.

Acknowledgements. We would like to thank the following individuals and organization for their help in order to conduct this R&D project: Dennis Pachur, Dirk Lohwasser, aitainment GmbH, Bremen, Germany. Adam McMahon, Justin Stoecker, Armando Locay, AI & Games Group, Department of Computer Science, University of Miami. Lester Melendez, Florida International University Dr. Asfour, Moataz Eltoukhy, Biomedical Engineering Lab, University of Miami.

References

1. Bethke, E.: Game development and production. Wordware Publishing, Plano (2003)
2. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: SIGGRAPH 1999: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 187–194. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1999)

⁵ <http://www.mrunix.net/webalizer/>

⁶ <https://www.google.com/analytics>

3. Bowyer, K.W., Chang, K., Flynn, P.: A survey of approaches and challenges in 3d and multimodal 3d + 2d face recognition. *Comput. Vis. Image Underst.* 101(1), 1–15 (2006)
4. Choppy, C., Bertrand, O., Carle, P.: Coloured Petri Nets for Chronicle Recognition. In: Kordon, F., Kermarrec, Y. (eds.) Ada-Europe 2009. LNCS, vol. 5570, pp. 266–281. Springer, Heidelberg (2009)
5. De Rosis, F., Pelachaud, C., Poggi, I., Carofiglio, V., de Carolis, B.: From greta's mind to her face: modeling the dynamics of affective states in a conversational embodied agent. *International Journal of Human-Computer Studies* 59(1-2), 81–118 (2003)
6. Ehrlich, S.M., Schiano, D.J., Sheridan, K.: Communicating facial affect: it's not the realism, it's the motion. In: CHI 2000 Extended Abstracts on Human Factors in Computing Systems, p. 252. ACM, New York (2000)
7. Ekman, P., Friesen, W.: Facial Action Coding System. Consulting Psychologists Press Inc., Palo Alto (1978)
8. Furukawa, Y., Ponce, J.: Dense 3d motion capture for human faces. In: CVPR, Miami Beach (2009)
9. Horaud, R., Niskanen, M., Dewaele, G., Boyer, E.: Human motion tracking by registering an articulated surface to 3d points and normals. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 31(1), 158–163 (2009)
10. Hsu, C.L., Lu, H.P.: Consumer behavior in online game communities: A motivational factor perspective. *Computers in Human Behavior* 23(3), 1642–1659 (2007)
11. Intille, S.S., Bobick, A.F.: Recognizing planned multiperson action. *Comput. Vis. Image Underst.* 81(3), 414–445 (2001)
12. Jenkins, O.C., Matarić, M.J.: A spatio-temporal extension to Isomap nonlinear dimension reduction. In: The International Conference on Machine Learning (ICML 2004), Banff, Alberta, Canada, pp. 441–448 (July 2004)
13. Kurihara, K., Hoshino, S., Yamane, K., Nakamura, Y.: Optical motion capture system with pan-tilt camera tracking and real time data processing. In: ICRA, Washington DC, USA, vol. 2, pp. 1241–1248. IEEE, Los Alamitos (2002)
14. Sector Games Media Board Berlin-Brandenburg. Online study for sector games. Technical report, Media Board Berlin-Brandenburg (2007)
15. Miene, A., Lattner, A., Visser, U., Herzog, O.: Dynamic-preserving qualitative motion description for intelligent vehicles. In: IEEE Intelligent Vehicles Symposium, Parma, Italy, pp. 642–646. IEEE, Los Alamitos (2004)
16. Miene, A., Visser, U., Herzog, O.: Recognition and Prediction of Motion Situations Based on a Qualitative Motion Description. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 77–88. Springer, Heidelberg (2004)
17. Mitobe, K., Kaiga, T., Yukawa, T., Miura, T., Tamamoto, H., Rodgers, A., Yoshimura, N.: Development of a motion capture system for a hand using a magnetic three dimensional position sensor. In: International Conference on Computer Graphics and Interactive Techniques, Boston, MA, p. 102. ACM SIGGRAPH (2006)
18. Moeslund, T.B., Hilton, A., Krüger, V.: A survey of advances in visionbased human motion capture and analysis. *Computer Vision and Image Understanding* 103(2-3), 90–126 (2006)
19. Mokrzycki, W.S., Salamonczyk, A.: Generating 3d multiview exact polyhedron representation by scanning faces surroundings. *MG&V* 15(3), 525–536 (2006)
20. Moreno, A.B., Sánchez, A., Frías-Martínez, E., Vélez, J.F.: Three-dimensional facial surface modeling applied to recognition. *Eng. Appl. Artif. Intell.* 22(8), 1233–1244 (2009)
21. Nakanishi, R., Maeno, J., Murakami, K., Naruse, T.: An Approximate Computation of the Dominant Region Diagram for the Real-Time Analysis of Group Behaviors. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 228–239. Springer, Heidelberg (to appear, 2010)

22. Parberry, I., Kazemzadeh, M.B., Roden, T.: The art and science of game programming. In: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, p. 514. ACM, New York (2006)
23. Pelachaud, C., Bilvi, M.: Computational model of believable conversational agents. In: Hugot, M.-P. (ed.) Communication in Multiagent Systems. LNCS (LNAI), vol. 2650, pp. 300–317. Springer, Heidelberg (2003)
24. Pelachaud, C., Carofiglio, V., Poggi, I.: Embodied contextual agent in information delivering application. In: Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems, Bologna, Italy (2002)
25. Pelachaud, C., Maya, V., Lamolle, M.: Representation of expressivity for embodied conversational agents. In: Proceedings of the AAMAS, Embodied Conversational Agents: Balanced Perception and Action, vol. 4 (2004)
26. Poggi, I., Pelachaud, C., de Rosis, F., Carofiglio, V., de Carolis, B.: Greta. a believable embodied conversational agent. In: Stock, O., Zancanaro, M. (eds.) Multimodal Intelligent Information Presentation, vol. 27, pp. 3–27. Springer, Heidelberg (2005)
27. Predinger, H., Ishizuka, M.: Life-Like Characters. Springer, New-York (2004)
28. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition, pp. 267–296. Morgan Kaufmann Publishers Inc., San Francisco (1990)
29. Raskar, R., Nii, H., de Decker, B., Hashimoto, Y., Summet, J., Moore, D., Zhao, Y., Westhues, J., Dietz, P., Inami, M., Nayar, S., Barnwell, J., Noland, M., Bekert, P., Branzoi, V., Bruns, E.: Prakash: Lighting-aware motion capture using photosensing markers and multiplexed illumination. ACM Trans. Graph. 26(3), 36 (2007)
30. Sementille, A.C., Lourenço, L.E., Brega, J.R.F., Rodello, I.: A motion capture system using passive markers. In: ACM SIGGRAPH International, Singapore, pp. 440–447. ACM, New York (2004)
31. Thomas, M.D., McGinley, J.A., Carruth, D.W., Blackledge, C.: Crossvalidation of an infrared motion capture system and an electromechanical motion capture device. In: Digital Human Modeling, Seattle, WA (2007)
32. Vicon (2009), <http://www.vicon.com>
33. Vlasic, D., Adelsberger, R., Vannucci, G., Barnwell, J., Gross, M., Matusik, W., Popović, J.: Practical motion capture in everyday surroundings. ACM Transactions on Graphics (TOG) 26(3), 35 (2007)
34. Warden, T., Lattner, A., Visser, U.: Real-time spatio-temporal analysis of dynamic scenes in 3D soccer simulation. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 366–378. Springer, Heidelberg (2009)
35. Wendler, J.: Automatisches Modellieren von Agenten-Verhalten. PhD thesis, Humboldt- Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II (2003)
36. White, R., Crane, K., Forsyth, D.: Capturing and animating occluded cloth. ACM Transactions on Graphics (SIGGRAPH) (2007)
37. Yu, Q., Li, Q., Deng, Z.: Online motion capture marker labeling for multiple interacting articulated targets. Computer Graphics Forum 26(3), 477–483 (2007)

Human vs. Robotic Soccer: How Far Are They? A Statistical Comparison

Pedro Abreu¹, Israel Costa², Daniel Castelão²,
Luís Paulo Reis¹, and Júlio Garganta²

¹ Faculty of Engineering of University of Porto , Department of Informatics,
Laboratory of Artificial Intelligence and Computer Science,
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
pha@fe.up.pt, lpreis@fe.up.pt

² Faculty of Sports of Porto University, Center of Investigation, Formation,
Innovation and Intervention in Sport, Rua Dr. Plácido Costa,
91 - 4200-450 Porto, Portugal
071103004@fade.up.pt, 081111010@fade.up.pt, jgargant@fade.up.pt

Abstract. In soccer games, a performance indicator is defined as a selection of action variables that aims to define all aspects of accomplishment of the game goals. However their perception during the match is extremely difficult. Over the years, soccer has been used in many research areas including the robotic international soccer competition, RoboCup. The aim of this research project is to present a comparison study, performed to detect similarities between these two games (Human versus Robotic Simulation 2D soccer). Having an off-line automatic event detection tool as a base, a collection of final game statistics was done and the Mann-Whitney test was used to verify their statistical significance. The results show that the most frequent events occurred in both types of game are successful passes. In what concerns stopped game situation types, in both types of games, the most frequent one is the Throw in situation (Human-59,8%, versus Robotic-74,1%) and the less frequent is the Corner situation (Human-13,7%, versus Robotic-10,3%). Some differences still reside, especially in the frequency of set pieces and the action prior the goal.

Keywords: Soccer Game Analysis, Data Mining Algorithms, Robotics, Artificial Intelligence.

1 Introduction

Today, the game results in a game of soccer determine the success of the team. The coach tries to obtain helpful knowledge to create specific training situations, with the goal of increasing the teams competitiveness and improve its results[2]. In order to obtain good performance indicators to help soccer coaches in the detection of the trends of players and teams, the analysis system must be constituted by complex algorithms. These algorithms are normally developed by researchers in artificial intelligence (AI) area and tested in simple simulated

environments, which usually enhance the interest of the researchers community. As one of the most popular and practiced sports in the world, soccer has been used in many research areas. RoboCup [5] is an international research project whose main objective is the promotion of AI and Intelligent Robotics. Basically, the research problem behind this project is the Robotic Soccer, where a number of distinct technologies are needed to construct a real or virtual Robotic team capable of playing a soccer game with a set of distinct rules. In this research work, we will focus only in the RoboCup simulation 2D league. This league is based in a system called soccerserver [8], which enables two teams of eleven players to play a soccer match in a simulated 2D environment. The system allows research in many different areas and many works appeared in several domains like flexible planning [12][13], intelligent communication systems [12][6] among others. The main goal of this research project is to measure the level of similarity between the soccer practiced by virtual Robots and Humans. To reach it, a soccer language was defined and subsequently used in software that was developed to calculate Robotic game statistics (number of passes, shots, goals, outsides, offsides) based on the log files of the 2d simulation league games. The analysis of the Human Soccer matches was based on audiovisual tapes and the data were registered on observation spreadsheets that have been developed for this project. The results, are based on the analysis of six games (three Robotics and three Human) corresponding to the latest Final Game competition. The remainder of this paper is organized as follows: Section 2 describes the methodologies used in this research work. Section 3 exposes the results achieved and finally in the last section a discussion is presented and final conclusions are presented.

2 Method

The human soccer data used in this study has been approved by a review board and the robotic data is available online (public domain-available at <http://ssl.robocup-federation.org/ftp/2d/log/>). In a soccer competition only the best teams, could reach the final game competition. In this research project, six games, corresponding to different tournament finals were selected in order to identify the similarities and differences between Robotic and Human Soccer. Regarding this goal, three Human Soccer Games (European 2004, 2008 and World Championship 2006 finals) and three Robotic Soccer Games (Robocup 2006, 2007 and 2008 finals) were chosen. In order to achieve the study objective, a set of soccer concepts were defined and a sequential analysis technique was used, to better characterize the game events. These concepts are represented in Table 1. The first definition presented in this soccer language is called a Kick. Generically, this event is based on the increase of the ball velocity. In this situation, many events could occur: a pass, a shot or even a goal. A successful pass occurs when a player kicks a ball and, after a finite interval of time, a teammate receives it. If an opponent player intercepts the ball, during this process, the event will be marked as an intercepted pass. In this project, a shot event was divided in three categories: a shot on target, an intercepted shot and a shot.

Table 1. Definition of the Generic Soccer Concepts

Generic Soccer Concept	Start Condition	Constraints	Final Condition
Kick	When a Player Increases the Ball Velocity Vector	Not Applicable	Not Applicable
Pass	When a Player Kicks the Ball	An Opponent Player does not Intercept the Ball	A Teammate receives the Ball
Shot	When a Player Kicks the Ball	An Opponent Player does not Intercept the Ball	The Ball Takes the Direction of the Goal Line
Goal	When a Player Kicks the Ball	The Ball does not Leave the Field	The Ball passes Through the Goal Line
Offside	When a Player Kicks the Ball	An Opponent Player does not Intercept the Ball	A Teammate that receives or is expected to receive the ball is not in a valid position

A shot on target occurs when a player kicks the ball in the goal direction. On the other hand, if the ball does not have the goal direction and left the field, this event will be marked as a shot. Finally if, after a shot on target or a shot, an opponent player intercepts the ball, this event will be selected as a intercepted shot. The other two events are directly based on FIFA rules. Concerning the offside rule, sometimes in soccer matches, when an offside situation occurs, if a player of the opponent team captures the ball, the referee, normally does not interrupt the game to mark the offside. In our project this situation was classified as an intercepted offside. Regarding the detecting of this set of events, two tools were used according to the different realities.

Robotic Soccer. Using the Robocup 2d simulation league log files as a base, a tool capable to automatically calculate the final game statistics was developed. The log files were produced by the soccerserver at the end of every Robotic game and contained detailed information regarding the game, such as position of the players and the ball in the field in each cycle or players stamina, among others. In this work, to calculate the final game statistics, only the information related to the players and ball position were used. The Robotic game in the 2d simulation league presents some differences when compared to Human Soccer. Each game is composed by 6000 cycles, there were no differences in the field characteristics that could influence the way of play and the teams always attack for the same side. Attending to these particularities, and to perform a sequential game analysis, a vector was constructed. Each vector position is designated as a scene and corresponds to one game cycle. Each position has information concerning players and ball position, etc. After breaking the game into a vector structure, a detection event algorithm will analyze it, starting with the detection of the kick events. Once this first analysis is done, the algorithm will try to identify the game events that happened in the match according to the start conditions specified in Table 1.

Human Soccer. From the DVDs with the Human Soccer Games, a spreadsheet was created in order to classify the different events of the games using a method

of observation. The main features that this tool supports are: identify all players that participated in the match, the different set of events, the duration of the event and finally, filter all events by time. This spreadsheet is capable to display, at the end of the monitoring process, the final game statistics.

Data Analysis. All data was analyzed using Microsoft Excel 2007 with a specific statistic package- Analyse-it version 2.20 (available at <http://www.analyse-it.com/blog/2009/5/analyse-it-2-20-released.aspx>). To perform the comparison between quantitative variables a non-parametric test, Mann-Whitney test, was used and the level of statistical significance was set at $p<0.05$.

3 Results

When the final game statistics (Table 2) are evaluated, in what concerns to passes ($U=21.5$ $p<0.01$), in the Human Soccer the successful passes represents almost 80% of the total executed in the game (with 22% of missed passes). Comparing to the Robotic passes reality, where the number of successful passes is lower than 66% and the missed ones are higher than in Human Soccer. However, the successful passes are the most frequent event in both realities. In the shots area, there were also statistical differences between these two realities ($U=86,5$ $p<0.01$) with the predominance of the shot on target (69,5% for Human and 47,6% for Robotic respectively). In the outsides group (Goal Kick, Corner and Throw-In) the most often event is the Throw-in. This particular situation is more frequent in Robotic soccer, but the other two considered events (Goal Kick and Corner) occur more in Human Games ($U=48,5$ $p<0.01$).

Table 2. Generic Comparison between Human versus Robotic Soccer

Groups	Final Game Statistics	Human Soccer			Robotic Soccer		
		Average per Game	Standard Deviation	Percentage	Average per Game	Standard Deviation	Percentage
Pass	Sucessfull Pass	544	117,9	77,9%	176,3	48,8	65,9%
	Missed Pass	154,3	21,3	22,1%	91	20,6	34,1%
Shots	Shot on Target	17,3	1,6	69,55%	3,3	0,5	47,6%
	Shot	2,7	0,5	11,64%	1,3	0,9	19,1%
	Intercepted Shot	4,3	1,7	18,81%	2,3	0,5	33,3%
Not Applicable	Goal	1	0,5	100%	2,1	1,2	100%
Offside	Offside	6,6	2,4	100%	1,6	0,8	18,76%
	Intercepted Offside	0	0	0%	4,3	1,2	81,24%
Outside	Goal Kick	17,4	2,4	26,37%	3,9	1,6	15,6%
	Corner	9,1	1,6	13,76%	2,3	1,4	10,3%
	Throw-in	39,3	5,7	59,87%	14,8	5,4	74,1%

3.1 Goal View

Doing a more careful examination of the goal event, and starting with the analysis of the goal scoring in a time-basis of half games it is relevant to note

that 75% or more of the goals in both scenarios were scored in the first half of the match and only 25% or less were scored in the second half of the game (Figure 1) ($U=19.5$ $p>0.05$). If the period of the match that contains the most goals (the 1st half) is divided in thirds of half, it is visible that goals scored by Robotic teams happen mostly in the first and second parts (more than 80% of the total amount of goals were scored in these two periods), while in Human Soccer this is not true (Figure 2). Observing the type of offense during the scored goals (Figure 3) it was detected that the Human and Robotic worlds have antagonistic behaviors ($U=44.5$ $p>0.05$). In the Robotic scenario, the offense style that more often allowed scoring a goal was the counter-attack situation (60% of the total), while in the Human reality was the set pieces (75% of the total situations). In the three Human Soccer Games, no goal was scored in a counter attack situation. The other offense situation (organized offense) corresponds to 25% and 20% of the total scored goals in Human and Robotic realities respectively. In the Robotic competition, all goals resulted from "combination play" (Figure 4), contrarily to Human Soccer, where the goals resulted from different situations: Direct Shot (25%), Combination Play (25%) and Long Pass (50%). These differences, however are not statistically significant ($U=101.5$ $p>0.05$). These actions prior to the goal interfere with the frequency of the Set piece observed in the two realities (in this work a set pieces occurs in the game when, after a Throw-in, Goal Kick or a Corner situation being detected, a pass combination between teammates (always involving four players or less) is performed and the time spent by the team to achieve the penalty area is relatively short (depending of the field area). It is not strange that in Robotic Soccer only the Throw-in was observed and it constitutes only 30% of the total scored goals. In Human Soccer, 53,3% of the goals were preceded by a Corner-Kick and 26,6% by a Penalty situation (Figure 5). The area of the field from where the offensive attempt was materialized was recorded and the findings indicate similarities between the two realities ($U=38.0$ $p>0.05$). In the Robotic world 80% of the goals

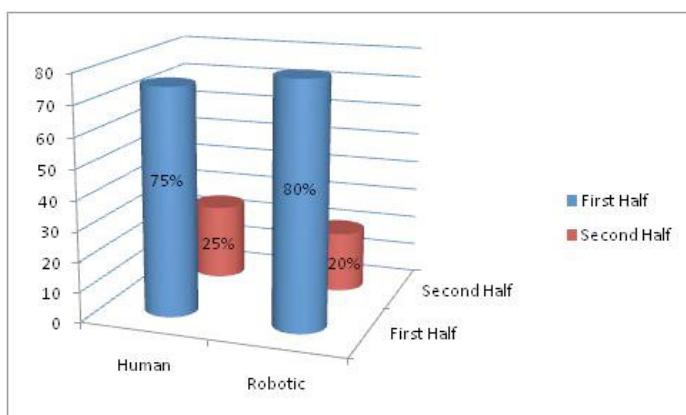


Fig. 1. Frequency of scoring by period of time (first and second half) and by reality

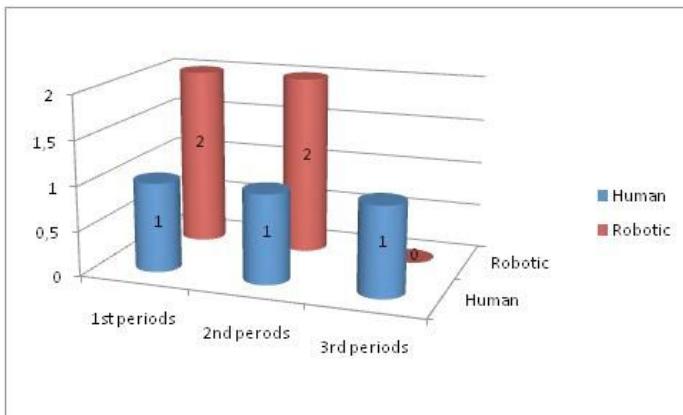


Fig. 2. Frequency of scoring by thirds of half and reality

were scored from the Penalty Area, and in the Human Soccer, all scored goals were from inside of the Penalty Area (50% from the Penalty Area and the other 50% from the Goal Area) (Figure 6).

3.2 Final Game View

Conducting now a more detailed analysis of each game, we started by analyzing the Human games. We observed that two of them ended in the regular time period, with only a goal scored (2004 and 2008 finals) and only one game (2006 final) ended in the penalties session. In what concerns to the executed passes in the regular time of the matches it is easy to note that, all over the years, the frequency of successful passes increased and the opposite was verified in the missed passes (Figure 7). Having only the passes as comparison criteria, we concluded that the most balanced final match was the 2006 final, and, curiously, this game was also the only one that needed extra time to find the winner. In the 2004 final, the winning team had less successful passes executed (171 versus 216) and in 2008 the winning team also presents less successful passes executed (318 versus 353), when compared to their opponent. However, in this final case, the successful passes of the Spanish Team constituted more than 84% of the total executed against 83% of their opponent, unlike the 2004 final game. If the analysis focuses on other statistics such as Shot (Figure 8), in the two first games the results show that the winning teams had less shots (almost in the three types) against their opponents but still won the games. However, in the third analyzed game, the winning team (Spain) had almost four times more shot on target events against Germany and, in the other type of shots only on the intercepted shot did these two teams present equivalent values. Making the same analysis for the RoboCup, and starting with the pass reality (Figure 7), the game where a more prominent difference between the two teams is verified is the 2007 final. This match was also the only one that did not need extra time to find the winner. Doing a parallel with the 2004 Human final game, a similar fact

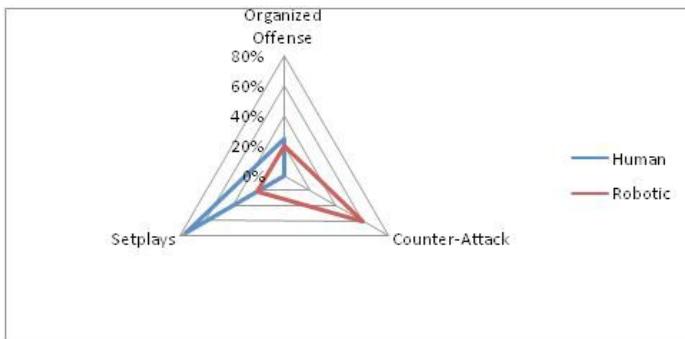


Fig. 3. Type of offense style used to score a goal by reality

happened: the winner team needed to do less successful passes compared to their opponent. In the other two games, the teams obtained similar values turning this factor into a balance indicator between the teams. Over the years, there were not many differences in terms of shot statistics in the Robotic games (Figure 8). A possible explanation for this is that these teams did not do many shots in the game. Although, the number of shots was in average similar in the two compared realities, the other two types of shots show very different frequencies.

4 Discussion

With this project work, we can conclude that Human soccer is more competitive, when compared with the Robotic one. This evaluation is sustainable by the presence of six different teams in three consecutive Human finals, compared to only two teams in the other reality. In a first instance, this fact could be explained by the rules of the Human soccer competitions. When a European Championship occurs, many powerful National Teams in this sport like Brazil, cannot participate in this competition. However, in the final games used in this work, only European Teams participated, even in the 2006 World Championship Final, which means that in the recent years the European Teams were better than the others. In what concerns to RoboCup, the official numbers show that the number of participants has increased every years, and therefore the presence of the same two teams in three consecutive finals can be explained by the fact that these teams have achieved, over the years, high performance levels, making it difficult for any younger teams to compete against them. In terms of game rules, some differences were detected in these two realities. In the extra time period, in the Robotic environment, the game immediately ends if a team scores a goal. In the Human reality this rule has been eliminated in international competitions since the World Cup 2002. In what concerns to the penalty shootout section and, in order to increase the research challenge, in the Robotic competition a rule from the ice Hockey and 7th soccer was used which consists in shooting the ball not in the eleven meters mark, but allowing the marker player will has

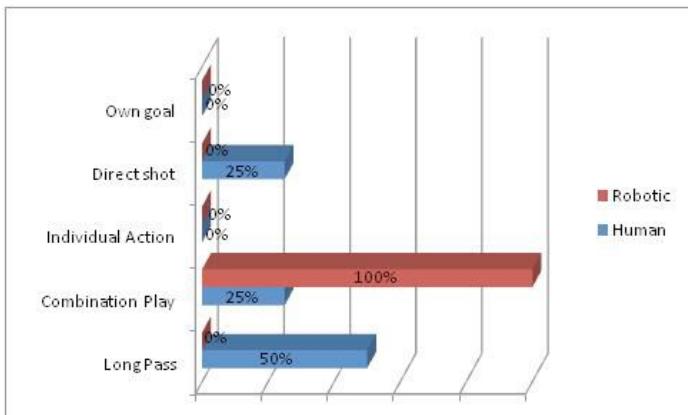
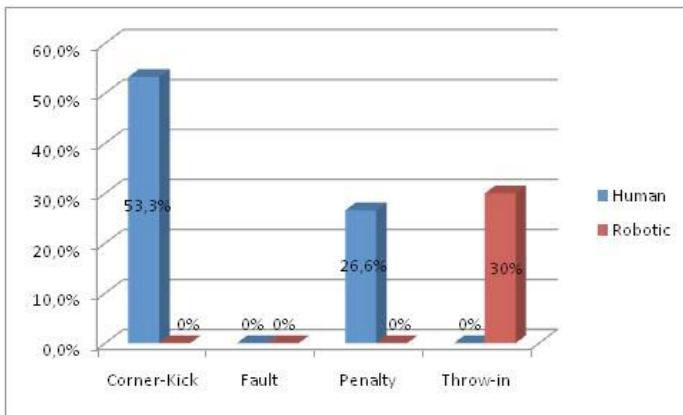
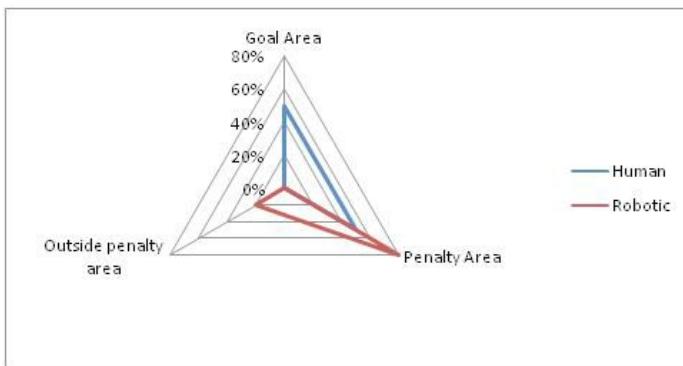


Fig. 4. Action prior to goal by reality

the freedom to lead the ball from the midfield and to shot from it in any field region. Over the years, many were the researchers that have tried to evaluate the development of Human Soccer, focusing their studies in goal characteristics [2]. In order to do a higher level comparison between the two treated realities a parallel with some research works will be produced. In 1968 Reep and Benjamin [10] analyzed more than 3000 matches and concluded that approximately 80% of all goals resulted from a sequence of three passes or less and a goal is scored every 10 shots. Although this study seems to be dated, in the past few years many were the studies that confirm these findings using different FIFA World Cup finals [3]. In our research games, the teams scored 9 goals (4 in the Human games and 5 in the Robotic games) and 66, 7% of them resulted from a sequence of three passes or less. Also, if the analysis splits the two realities, the conclusions show that 75% of Human scored goals resulted from a sequence of two passes or less (usually in set pieces), against 60% for the Robotic reality. For the second finding from Reep and Benjamin, and if the shot definition covers the three defined types, none of the games confirms the theory. In the Robotic scenario, only 2 of 3 games had goals and in these particular matches, the teams only need 9 shots to score 2 goals (2006 final) and in another match only 8 shots are needed to score 3 goals (2007 final). From the Human perspective, in the 2004 final, only one goal was scored and 27 shots were counted; in the 2006, final 19 shots were counted, and 2 goals scored and finally, in the 2008 final, 23 shots were counted and only 1 goal scored. Even if the definition of the shot used by Reep only covered the shot and shot on target or only the shot target, the extracted results would still not confirm his theory. Continuing analyzing the goals and its characteristics, over the years many were the studies that aggregated the goal scoring in reference to time of accomplishment supporting that the frequency of goal scoring is time dependent. The results produced in our research work show that the highest percentage of goal scoring, in both realities, happened in the first half of the game. However when the match was divided

**Fig. 5.** Frequency of set pieces types**Fig. 6.** Area from which the goals were scored

in 15 min periods, the results show a correlation between the 2 first periods (in the first half) in the Robotic environment. On the other hand, in the Human world, the results did not show any type of correlation between 15 min periods and scored goals. Although many studies demonstrated that, at the end of the games, the performance of the teams drop because of the greater deterioration in physical condition, and sometimes the losing team takes some risks, pushing players forward, trying to create goal opportunities, conceding not often further goals [11]. Others are that did not find a correlation between goals and time period like ours [7]. Regarding the type of offense during goals, a database was constituted, from the 1982 Tournament by Piecniczk [9] that concluded that 27% of the goals were resultant of a quick offense and only 28% were through organized offensive actions. From the 1990 World Cup Tournament Dufour [1] concluded that this trend had inverted: 88% of the goals came from an organized

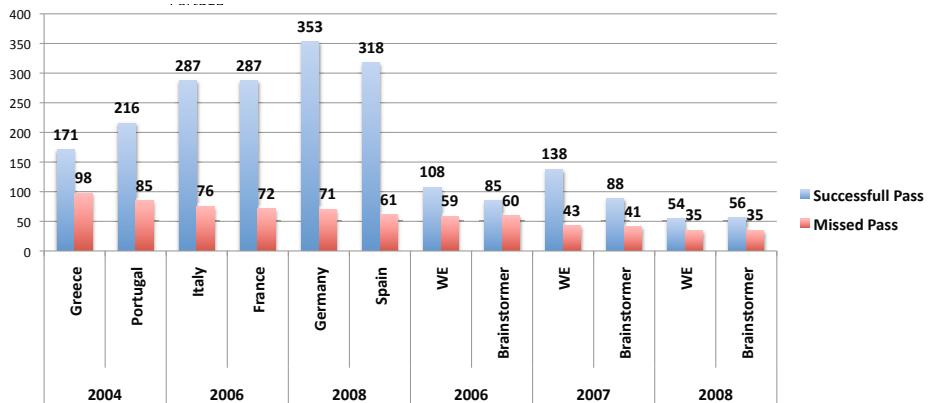


Fig. 7. Pass Statistic in Human and Robotic Game Finals by match

offense and 12% from a strike offense. Nowadays, the execution of set pieces constitutes an import part of a team tactics as well as in the outcome of a game. Like our study, many others claim that more than 1/3 of the scored goals in many competitions resulted from set pieces [2]. On the other hand, we observed that, in the Robotic world, the most used movement to score goals was the counter attack (60% of the total scored goals). Examining the actions that lead to the goal, our findings show that in Robotic world all process resulted from a combination plays (100%). In the Human environment the results were a bit different. The long passes represent the highest percentage of cases (50%) followed by direct shot (25%) and combination play (25%). Doing a parallel with the scientific literature the findings were similar to the results obtained in the Human game finals. The long passes represent the highest percentage (more than 34%) followed by the combination play (29,3%) and individual actions (17,1%) [14]. Analyzing the set pieces more common before the goal and, in consequence of the results previously explained; in the Robotic environment only 30% of the scored goals were preceded by a set pieces (Throw-in situation). In the other analyzed reality the most common set pieces was the Corner Kick (more than 57%), followed by the Penalty situation (more than 26%). However, the achieved results produced by some research works [4] present some different characteristics, indicating that, in spite of the corner Kick having a good percentage in terms of goals scored (27% and 24,4% respectively, the most relevant set piece was the free-kick, which represent 37% and 39%, respectively. Another recent study [14] showed that the major set piece observed before the goals scored was the corner situation (40% followed by the free kick situation, with 30%). Although the comparison of these studies provides, to a certain point, dissimilar results, even in the Human final scenario, it is clear that the corner kick situations occupy a higher percentage of the produced goals. In the Robotic environment, the unique set piece before the goal was the throw in. In what concerns to the area where the final effort

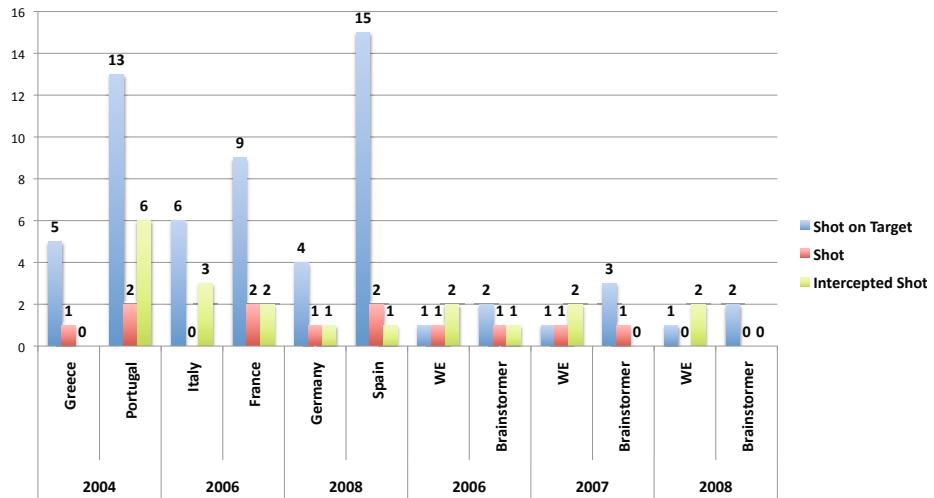


Fig. 8. Different Types of Shots in Human and Robotic Game Finals by match

was materialized, the findings of our research indicates that the majority of the Robotic goals were scored from inside the penalty box (80%). However, in the other observed reality the frequency was divided between the penalty area (50%) and the goal area (50%). In the literature, the results achieved were very similar to our study. In the 2002-03 Champions League season Michailidis [7] concluded that more than 64% of the goals were scored from inside the penalty area and 36, 5% from the goal area. Other studies, produced for instance by Dufour [1], indicated that 81,8% and 80% of goals were scored from inside the penalty area and 16% and 15% from the goal area, respectively. Summarizing, our study shows some similarities between the two realities analyzed in what concerns to the frequency of outside situation and also in the relation between successful and missed passes. The main differences still reside in the action prior the goal and the frequency of set pieces.

Acknowledgments. The first and second author are supported by FCT under grant SFRH / BD / 44663 / 2008 and by the PA grant E07D400279BR respectively. This work has been partially funded by FCT Project ACORD - Adaptable Coordination of Robotic Teams (PTDC/EIA/70695/2006).

References

1. Dufour, W.: Computer-assisted scouting in soccer. In: Reilly, T., Lees, A., Davids, K., Murphy, W.J. (eds.) SF, London, pp. 160–166 (1993)
2. Garganta, J., Maia, J., Basto, F.: Analysis of goal-scoring patterns in European top level soccer teams. In: Reilly, T., Bangsbo, J., Hughes, M. (eds.) SF III, pp. 246–250. E and FN Spon, London (1997)

3. Hughes, M., Franks, I.: Analysis of passing sequences, shots and goals in soccer. *JSS* 23(5), 509–514 (2005)
4. Jinshan, X., Xiakone, C., Yamanaka, K., Matsumoto, M.: Analysis of the goals in the 14th World Cup. In: Reilly, T., Clarys, J., Stibbe, A. (eds.) SF II, pp. 203–205. E. and FN Spon, London (1993)
5. Kitano, H., Tambe, M., Stone, P., Veloso, M., Noda, I., Osawa, E., Asada, M.: The RoboCup Synthetic Agents Challenge. In: Kitano, H. (ed.) RoboCup 1997. LNCS, vol. 1395, pp. 62–73. Springer, Heidelberg (1998)
6. Lau, N., Reis, L.: FC Portugal - High-level Coordination Methodologies in Soccer Robotics. In: Lima, P. (ed.) Robotic Soccer, pp. 167–192. Itech Education and Publishing, Vienna (2007)
7. Michailidis, C., Michailidis, I., Papaikovou, G., Papaikovou, I.: Analysis and evaluation of way and place that goals were achieved during the European Champions League of Football 2002-2003. *Sports Organization* 2(1), 48–54 (2004)
8. Noda, I., Matsubara, H., Hiraki, K., Frank, I.: Soccer Server: A Tool for Research on Multiagent Systems. *Applied AI* 12, 233–250 (1998)
9. Piecniczak, A.: Preparation of football teams for Mundial Competition in 1986. Communication to 9th UEFA course for National Coaches (1983)
10. Reep, C., Benjamin, B.: Skill and chance in ball game. *JRSS*, A 131, 581–585 (1968)
11. Reilly, T.: Energetics of high intensity exercise (soccer) with particular reference to fatigue. *JSS* 15, 257–263 (1997)
12. Reis, L.P., Lau, N.: FC Portugal Team Description: RoboCup 2000 Simulation League Champion. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 29–40. Springer, Heidelberg (2001)
13. Reis, L.P., Lau, N., Oliveira, E.C.: Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In: Hannebauer, M., Wendler, J., Pagello, E. (eds.) ECAI-WS 2000. LNCS (LNAI), vol. 2103, pp. 175–197. Springer, Heidelberg (2001)
14. Yiannakos, A., Armatas, V.: Evaluation of the goal scoring patterns in European Championship in Portugal 2004. *IJPAS* Vol 6(1), 178–188(11) (2004)

Learning Powerful Kicks on the Aibo ERS-7: The Quest for a Striker

Matthew Hausknecht and Peter Stone

Department of Computer Science, The University of Texas at Austin
`{mhauskn,pstone}@cs.utexas.edu`

Abstract. Coordinating complex motion sequences remains a challenging task for robotics. Machine Learning has aided this process, successfully improving motion sequences such as walking and grasping. However, to the best of our knowledge, outside of simulation, learning has never been applied to the task of kicking the ball. We apply machine learning methods to optimize kick power entirely on a real robot. The resulting learned kick is significantly more powerful than the most powerful hand-coded kick of one of the most successful RoboCup four-legged league teams, and is learned in a principled manner which requires very little engineering of the parameter space. Finally, model inversion is applied to the problem of creating a parameterized kick capable of kicking the ball a specified distance.

1 Introduction

Robocup is an international research and education initiative aiming to develop a team of fully autonomous humanoid robots that, by the year 2050, can win against the human-soccer world championship team. In annual Robocup competitions, teams of robots are pitted against each other in a soccer match with both field and rules resembling their human equivalents. For robots, as for humans, the ability to effectively attack the goal and score is crucial. Thus powerful, accurate kicks are highly prized.

Typically, kicks and other motion sequences are painstakingly created and manually tuned. In the last few years however, machine learning has optimized motion sequences associated with the skills of walking quickly [9,13,12], receiving passes [11], and capturing the ball [6]. The use of machine learning to optimize a skill has the benefits of removing human bias from the optimization process and, in many cases, reducing the amount of human labor required to create a top notch motion.

We optimize kicking distance on the Sony Aibo ERS-7 using Hill Climbing and Policy Gradient algorithms. The resulting learned kick is compared to the most powerful hand-coded kick of one of Robocup's perennial top competitors. Experiments conducted over multiple robots show the learned kick significantly outperforms this team's most powerful hand-coded kick.

Like scoring, good passing is an important team skill. Unlike scoring, the most powerful kick is not always the best choice for successful passing. While a powerful kick may be able to move the ball to the receiving Aibo, the superfluous

velocity complicates the process of receiving the pass and introduces possible bounce-back concerns. To address these issues, we create a parameterized kick capable of accurately moving the ball a desired distance. As in human soccer, the passing Aibo needs only estimate the distance to the receiver and adjust the power of its kick accordingly.

The remainder of the paper is organized as follows: Section 2 presents the background of learning skills and kicking. Section 3 covers the parameterization of the kick. Section 4 discusses the physical framework in which kicks were learned. Section 5 introduces the algorithms used for learning. Section 6 presents the primary results of optimizing kick distance while Section 7 covers additional results and the variable distance kick. Conclusions and future work are presented in Section 8.

2 Background and Related Work

Related work can be categorized into three main thrusts: that which has optimized Robocup skills, that which has modeled the effects of Aibo kicks, and that which has used simulation to learn more effective kicks.

Robocup skill learning has optimized the skills of walking quickly, walking with stability, ball grasping, and ball trapping. Quick walking has been optimized by several groups [3,8,9,10,13], with learned gaits routinely outperforming their hand-coded counterparts. Of the gait learners, Kohl and Stone [12] compared the performance of hill climbing, amoeba, genetic, and policy gradient algorithms on the task of optimizing Aibo walk speed. Results show the learned walked significantly outperforms the best hand-coded walks and rivals other learned walks of the time. Subsequently, camera stability was incorporated into the learning process, allowing Saggar et al. [14] to learn a walk which sacrifices a small amount of speed in exchange for superior image stability.

The ball-grasping skill was optimized by Fidelman and Stone [6] using a Layered Learning framework [15]. The final ball grasping behavior shows significant improvements over hand-coded equivalents. Similarly, Kobayashi et al. [11] learn how to trap the ball using Reinforcement Learning [16]. Trapping, as opposed to grasping, is the process of intercepting and capturing a moving ball rather than a stationary one. Results show learning stabilizes near 80% successful trapping accuracy after approximately 350 episodes.

One of the main appeals of using machine learning is the reduction of manual effort. Thus, in each of the skill optimization approaches mentioned above, a physical learning framework was created to minimize the amount of human interference necessary throughout the learning process. In the above examples, humans were only required to start the learning process and to replace the Aibo’s battery once drained. The robots were able to autonomously test and evaluate different variations of the skill being learned, keeping track of their own progress. Like previous work, we create a learning framework to minimize the human interference in the learning process. The main novelty in our work is that, to the best of our knowledge, we are the first to use machine learning to optimize

the kicking skill fully outside of simulation, using a much higher dimensional parameterization than those of the aforementioned skills.

A second class of related work has focussed on modeling the effects of kicks. Chernova and Veloso [4] examined the effects of an arm and a head kick by tracking the angle and displacement of the ball directly after the kick was performed. They subsequently use this knowledge to select the most appropriate kick for a given situation, effectively reducing the time taken for an Aibo to score a goal. Similarly, Ahmadi and Stone [1] investigate kick models for use in the action planning domain. Unlike the parameterized kick models used by Chernova and Veloso, Ahmadi employs instance based models, in which the resulting ball location of each kick is recorded. These more detailed models allow the robot to plan more thoroughly about the effects of a given kick in the presence of obstacles or other robots. Rather than modeling a number of discrete kicks, each approximating a desired effect, we create a parameterized kick capable of moving the ball any reasonable desired kick distance.

Third, the process of learning to kick has been explored primarily inside of the realm of simulation. Zagal and Ruiz-del-Solar [18] used a simulator to investigate reactive ball kicks. Decomposing each kick into four 25-parameter poses (configurations of joint angles for each limb), they successfully learned a falling kick – qualitatively similar to most powerful kicks of the time. Cherubini, Giannone, and Iocchi [5] learned two kicks as a part of a larger, Layered Learning approach focused on learning to attack the goal. Learning started on a simulator and was subsequently continued on the physical robot, where the best simulator policies became the robot's initial policies. Results show progress on the overall task of attacking the goal, but are not reported for the subtask of improving the kicks. Finally, Hester et al. [7] recently applied model-based reinforcement learning to the task of learning to score penalty goals. Using the RL-DT algorithm and a 2-features state space, learning was first applied in simulation and then validated on the real robots. Results show a success rate of 85% in simulation and 70% on the actual robot. Unlike previous work, our approach has no simulation component. To the best of the author's knowledge, this is the first time a kick has been learned fully on physical robots.

In order to survey state of the art kicks, eleven hand-coded kicking motions were examined from a team who has competed in Robocup for several years, regularly placing highly. Each kick consisted of 4-19 discrete poses (full-body joint angle specifications) and the amount of time to hold each pose. Of the eleven kicks examined, the "Power Kick" was the clearly the most powerful and accurate. This kick grasps the ball between the Aibo's front legs and slams the head down, knocking the ball straight ahead of the Aibo. For the remainder of this work, all learned kicks are benchmarked against the Power Kick.

3 Kick Parameterization

In any learning process, the choice of parameterization can dramatically affect results. Increasing the number of parameters increases the learning time and

complexity of the problem. On the other hand, decreasing the number of parameters limits the scope of possible learning, in some cases simplifying the problem to the point of triviality. Previously, Kohl used 12 parameters to characterize a walk and Kobayashi used only two in order to learn the ball trapping skill. In simulation Zagal learned kicking with 100 parameters.

We use 66 parameters to characterize a kick – an increase from previous non-simulation learning (Chernova's 54 parameter walk [3] comes closest). Since the Aibo has 18 total joints, a maximum parameterization for a 6 pose kick would use 108 joint parameters ($6 * 18$) and 6 pose timing parameters for a total of 114 parameters. We reduced the maximum parameter space in two ways: First, exploiting the Aibo's bilateral symmetry, the three joints in the front left leg were made to mirror those of the front right; the same was true of the back legs. Second, the robot's two tail joints were considered useless and eliminated. These changes reduced the number of parameters per pose from 18 to 10 – for a total of 66 parameters. By only exploiting the inherent symmetry of the robot, our parameterization retains as much flexibility as possible. Attempts were made to use an even more flexible parameterization – one not exploiting symmetry – but there were no obvious advantages. Figure 1 displays the 10 parameters used in each pose and the number of possible values each parameter could assume. Integer values were required by the ERS-7 hardware.

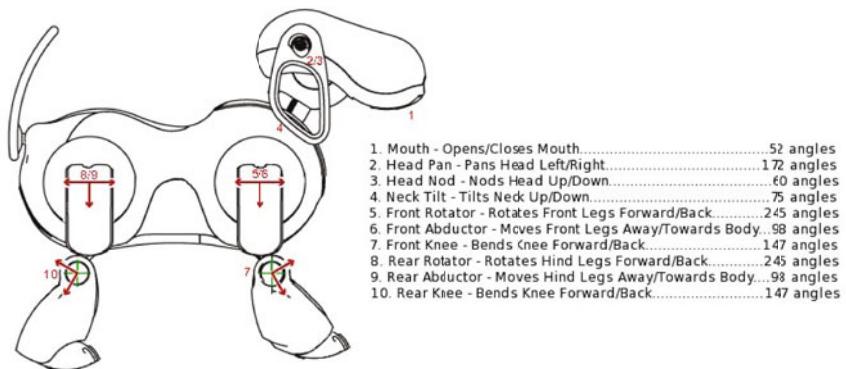


Fig. 1. The ten joints per pose to which learning was applied and the number of possible angles each joint could assume.

As a final note, this parameterization required very little ad hoc engineering or knowledge of the domain. To illustrate this point, consider the 12 parameters used by Kohl to optimize Aibo walk speed: Six of them specify length, height, and shape of the elliptical locus through which the end effector of the foot moves. Two more specify the height of the front and rear body while another determines the fraction of time each foot remains on the ground. Clearly learning has been abstracted to a higher level than specifying raw joint angles. Learning in such a manner often proves more time efficient but requires extensive engineering of

the parameter space. In this case it would be necessary, at the minimum, to engineer functions moving each foot through an adjustable elliptical locus while simultaneously raising and lowering front and rear body height. In contrast, our learning process operates on raw joint angles, requiring no intermediate functionality.

4 Learning to Kick

Figure 2 depicts the inclined ramp constructed to partially automate the kick learning process. Learning consisted of the Aibo kicking the ball up the ramp, receiving feedback on the distance of the kick, and being repositioned for the next kick. The slope of the ramp was adjustable by changing the height of the object the ramp was resting upon. A suitable slope was found through trial and error, guided by the principle that the ramp should be as low as possible so long as the Aibo was unable to kick the ball off the far end.

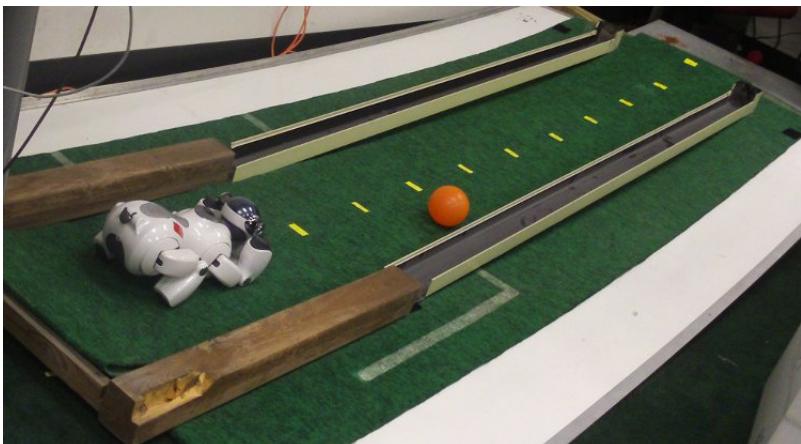


Fig. 2. Inclined Ramp for Optimizing Kick Distance

Despite attempts at full autonomy, a human was required to relay the distance of the kick and reposition the Aibo after each kick. Making this process fully autonomous proved difficult for several reasons: first, the kicking process proved violent and unpredictable; It was common for the Aibo to either fall over or fumble the ball backwards, making it hard to recover and reposition the ball for the next kick. Second, because of the carpet covering the ramp, the ball would occasionally get stuck at the apex of its path. This could potentially have been resolved with a steeper incline, but that would have led to shallower kicks and less informative distance feedback. Finally, it would have been problematic for the Aibo to autonomously estimate kick distance primarily because many kicks ended with the Aibo's head (and sometimes body) facing in a different direction

than that in which the ball was kicked. Estimating distance for these kicks would require extremely fast visual re-acquisition and distance estimation before the ball reached its apex. Despite the dependence on human assistance, the ramp system was quite efficient, yielding approximately 500 kicks over the duration of a single battery charge, approximately one hour – an average of one trial every seven or eight seconds.

5 Machine Learning Algorithms

In order to learn the kick within the framework described above, we tested two existing optimization algorithms. Specifically, we used exactly the Hill Climbing and Policy Gradient algorithms described in detail by Stone and Kohl in their work on quadruped locomotion [12]. The Policy Gradient algorithm was used with a step size η of 2.0 and run for 65 iterations, in each of which 5 new policies were generated and evaluated. Likewise, Hill Climbing used 5 policies per iteration for 27 iterations. The number of iterations used is a result of running each algorithm for the course of a single battery charge. In both algorithms, new policies were generated by incrementing each joint angle by a random number in the range of zero to one-tenth of the full output range of that joint.

The Hill Climbing algorithm had the advantages of simplicity and robustness in the presence of high numbers of parameters. Hill Climbing starts with an initial policy π consisting of initial values for each of the parameters and generates one or more additional policies $\{R_1, R_2, \dots, R_t\}$ in the parameter space around π . Each policy is then evaluated and the best is kept as the initial policy in the next iteration. Because Hill Climbing is little more than a guess and check algorithm, it can safely be used in the presence of high dimensional parameter spaces, making it ideal for both tuning and learning high dimensional kicks.

The Policy Gradient Algorithm was chosen because it had previously proven effective at learning Aibo walking and grabbing. The specific variant of general policy gradient learning techniques [2,17] seeks to estimate the gradient of the objective function by sampling policies in the vicinity of the initial policy and inferring the partial derivative for each parameter. After the gradient has been estimated, the new policy is shifted in the direction maximizing the objective function.

6 Results

The main result of this work is the creation of a kick significantly more powerful than a competitive team's most powerful hand-coded kick, the Power Kick. To achieve this result, Policy Gradient and Hill Climbing algorithms were applied in succession to create the final learned kick. Starting from the Power Kick, Policy Gradient was run for 65 iterations (650 kicks) over the course of a single battery charge. In each iteration, ten policies were evaluated with the score for each policy determined by allowing the policy to generate a single kick and timing how long it took for the ball to roll up the ramp and return the Aibo. Figure 3

plots the score of the first policy at each iteration with higher millisecond return times corresponding to more powerful kicks. Additionally, the sum of squared difference for each parameter is plotted between the current policy and the initial policy. This measure indicates whether or not the current policy is stuck inside of a local optimum around the initial policy or if it is successfully traversing the parameter space.

As Figure 3 indicates, Policy Gradient effectively explores the parameter space without getting stuck in a local optimum around the initial policy. Generally, while there were occasional spikes in score (one kick went off the far end of the ramp), there is little sustained improvement in overall score. Even though PG did not converge to a single kick, it did identify, in isolation, several promising motions sequences. The single most powerful kick (iteration 47) from the 65 iterations of PG was selected and further refined through 21 iterations of Hill Climbing, in each of which 5 policies were generated and scored by averaging the power of two kicks generated by that policy.

At this point, the Learned Kick was evaluated against the Power Kick on flat ground with ten different Aibos and five trials of each kick per Aibo. Figure 4 displays the resting locations of the 50 Power and Learned Kicks. On average, the Learned Kick moved the ball 373.66cm with a standard deviation of 105.51cm compared to the Power Kick which had an average kick distance of 322.4cm and a standard deviation of 130.21cm. This increase proves statistically significant with a two-tailed P value of 0.0330 on an unpaired t-test. Additionally, the learned kick was able to always move the ball at least 200cm and yielded one kick so powerful that it was stopped only after hitting the far wall of the room, a distance of 628cm, or over 20 feet!

Accuracy for both kicks was assessed by recording how many centimeters the kick deviated from the X-axis per meter of kick distance. According to this metric, the Power Kick is slightly, but not statistically significantly, more accurate than the Learned kick, with an average Y-deviation of 15.82cm per meter of kick distance (std. dev. 14.88) compared to the Learned Kick's 19.17cm Y-deviation (std. dev. 16.32). This effect is likely a result of accuracy not being emphasized during the learning process, aside from the slight loss of momentum when an inaccurate kick would hit a wall of the inclined ramp.

Figures 5 and 6 show the poses and action of the learned kick. Videos of both kicks can be found online at <http://www.youtube.com/user/aiboKick>.

7 Additional Results

The results in Section 6 demonstrate that learning a kick can yield a stronger kick than laborious hand-tuning. In this section, we further examine the effect of the starting kick on the learning process, as well as whether the kick can be modified for variable distances.

7.1 Effect of Starting Kick

Our final kick was based on learning from an already-tuned starting point. In this section we demonstrate that learning can also work from a weak starting point. Figure 7 shows the results of 27 iterations of Hill Climbing (270 kicks) from a motion sequence unable to propel the ball forwards.

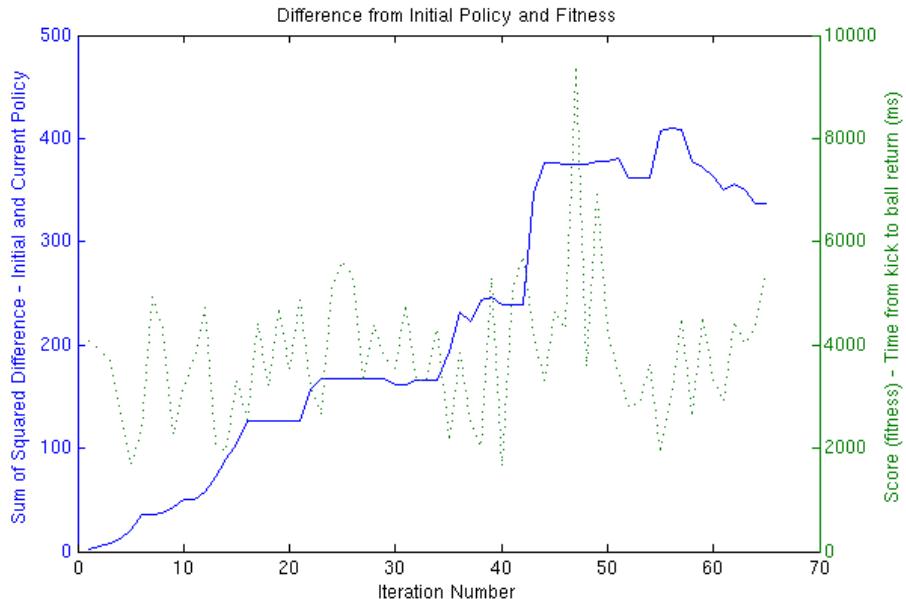


Fig. 3. Policy Gradient Learning from Power Kick

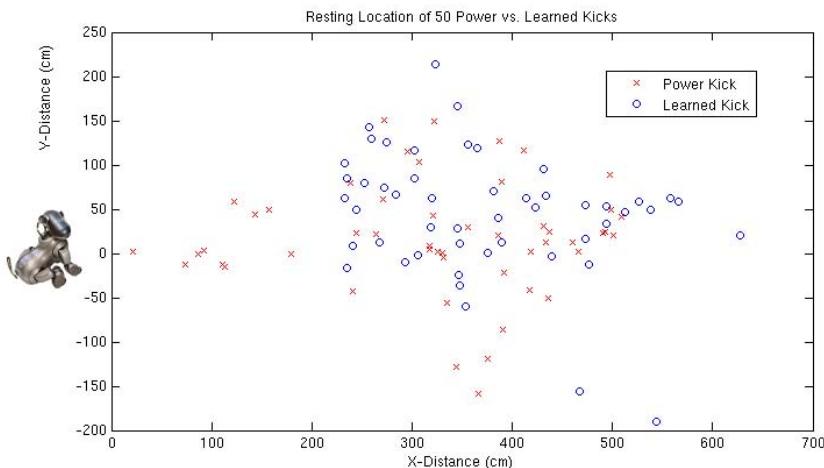


Fig. 4. Aibo Kicks from origin in positive X-direction



Fig. 5. The six poses comprising the final learned kick



Fig. 6. Learned Kick in motion

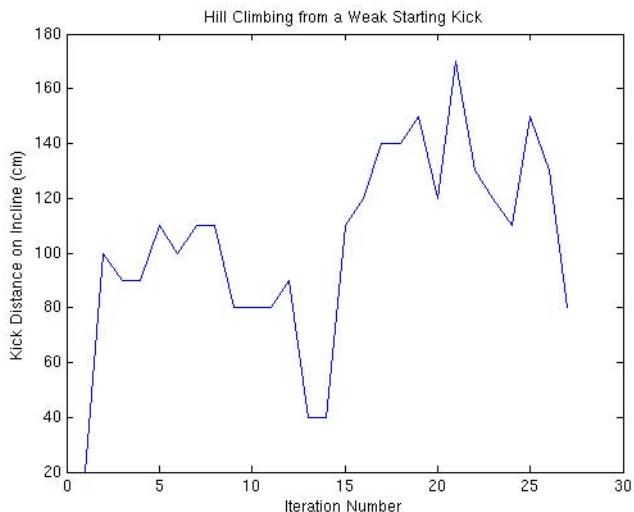


Fig. 7. Hill Climbing from a Weak Starting Kick

Five policies were evaluated in each of the 27 iterations of hill climbing, with the best becoming the initial policy for the next iteration. Each policy was scored by averaging two kicks generated by that policy. The power of the kick at each iteration was evaluated by noting how far up the 200cm ramp the ball traveled. The initial policy started with a distance score of zero, as it was unable to kick the ball forwards. In just a few iterations a working kick was discovered and generally optimized throughout the rest of the run. After only 21 iterations, the kick was powerful enough to near the end of the 200cm ramp. Admittedly, not any initial policy will lead to a working kick. Fully random initial policies were experimented with and generally had little potential for creating working kicks. In contrast, the starting motion sequence used here had the advantage of keeping the Aibo roughly in a standing position for all 6 poses.

7.2 Variable Distance Kick

While a powerful and accurate kick might be able to reliably reach a target Aibo, we speculate that overpowered passes are harder to receive than passes with a predictable, and more moderate, velocity. Accordingly, we seek to design a kick which gives the passing Aibo control over the power and distance of the hit.

Rather than learning a new kick, or possibly many new kicks – one for each desired distance, we investigate how changes in the timings between poses on an existing kick affect that kick’s power. Both the Power Kick and the Learned Kick (Section 6) consist of 6 poses with 6 pose transition times. In both, the fourth pose was identified as the “hit” pose – the pose in which the Aibo makes contact with and propels the ball forwards. It was observed that varying the amount of time allotted to this pose greatly affected the power of the kick.

Distances of Power and Learned kicks on flat ground were recorded for number of frames in “hit” pose ranging from 0 to 100, with the Figure 8 showing the interesting part of each range. Each data point represents the average distance of three kicks.

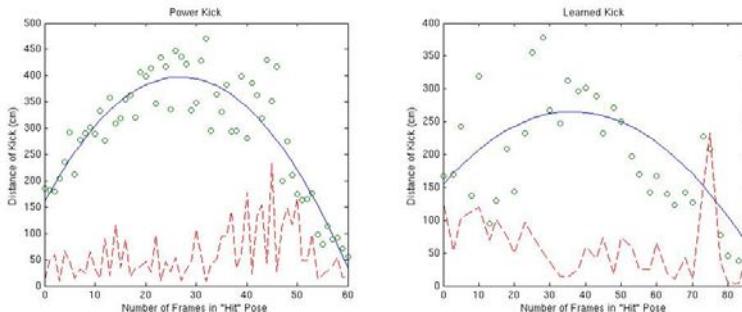


Fig. 8. Quadratics (solid line) fit to kick distance data points (circles) and standard deviations of the 3 kicks comprising each data point (dashed line)

After distance data was collected, a quadratic was fit to the points. In order to invert the model, it was necessary to solve for the number of frames as a function of the desired kick distance. Table 1 displays the two solutions found for each kick.

In the case of the Power Kick, the standard deviations shown in Figure 8 indicate that the left side of the quadratic has lower variance and should thus be

Table 1. Quadratic equations fit to kick distance data and their solutions. x is the number of frames in “hit” pose and y is the desired kick distance. Solution 1 corresponds to the left side of the quadratic while Solution 2 fits the right side.

	Power Kick	Learned Kick
Quad Eq.	$y = -3x^2 + 17.7x + 160.1$	$y = -0.08x^2 + 6.1x + 154.6$
Soln. 1	$x = -(\sqrt{15} * \sqrt{87437 - 220 * y} - 885)/33$	$x = -(5 * \sqrt{5} * \sqrt{44561 - 168 * y} - 1525)/42$
Soln. 2	$x = (\sqrt{15} * \sqrt{87437 - 220 * y} + 885)/33$	$x = (5 * \sqrt{5} * \sqrt{44561 - 168 * y} + 1525)/42$

preferred over the right side whenever possible. As a result, a simple variable distance kick was designed which uses Solution 1, corresponding to the left side of the quadratic, if the desired kick distance is between 160cm and 398cm (the maximum of the quadratic function) and Solution 2, corresponding to the right side of the quadratic, for all other distances. Because the Learned Kick generally showed smaller standard deviations on the right side of the quadratic, Solution 2 was preferred for all distances.

To evaluate the control of both kicks, 20 desired kick distances in the range of 60cm to 400cm were randomly generated. Both kicks were then evaluated based on how close to each desired distance they were able to propel the ball. On average the Power Kick was accurate to within 57.8cm of the requested distance (standard deviation of 41.1cm) and the Learned Kick was accurate to within 45.5cm (standard deviation 39.3cm). To phrase it differently, both kicks were able to place the ball within 1-2 Aibo lengths of their target. It seems likely that the methods applied here to convert a normal kick into a variable distance kick would be equally applicable to nearly any type of kick.

8 Future Work and Conclusions

In this work we demonstrated the learning of a kick more powerful than a strong hand-coded kick. More importantly, this kick was learned in a way that required minimal task-specific engineering of the parameter space. In addition, we created a parameterized kick capable of propelling the ball a requested distance.

This paper opens up several interesting directions for future work. First, a more complex inclined ramp could be created to increase the autonomy of the learning process. Additionally, it would have been desirable to benchmark the Learned Kick against the best hand coded kicks of a variety of different teams. Furthermore, accuracy has yet to be incorporated into the learning process. This could potentially be accomplished by recording the distance from the kicking Aibo to either the apex of the kick or the first collision with one of the incline walls. It would be meaningful to evaluate the Variable Distance Kick in the context of the Robocup passing challenge, demonstrating the connection between controllable kicks and successful passes. Finally, it would be interesting to see if a similar learning process could be used for learning to kick on humanoid robots.

Acknowledgements

The authors would like to thank Michael Quinlan, Shivaram Kalyanakrishnan, and Todd Hester for useful discussions, as well as the entire UT Austin Villa robot soccer team for development of their code base. Additional thanks to Julian Bishop for help constructing the inclined ramp. This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-0615104 and IIS-0917122), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), and the Federal Highway Administration (DTFH61-07-H-00030).

References

1. Ahmadi, M., Stone, P.: Instance-based action models for fast action planning. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 1–16. Springer, Heidelberg (2008)
2. Baxter, J., Bartlett, P.L.: Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res (JAIR)* 15, 319–350 (2001)
3. Chernova, S., Veloso, M.: An evolutionary approach to gait learning for four-legged robots. In: Proceedings of IROS 2004 (September 2004)
4. Chernova, S., Veloso, M.: Learning and using models of kicking motions for legged robots. In: ICRA (May 2004)
5. Cherubini, A., Giannone, F., Iocchi, L.: Layered learning for a soccer legged robot helped with a 3d simulator, pp. 385–392 (2008)
6. Fidelman, P., Stone, P.: The chin pinch: A case study in skill learning on a legged robot. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 59–71. Springer, Heidelberg (2007)
7. Hester, T., Quinlan, M., Stone, P.: Generalized model learning for reinforcement learning on a humanoid robot. In: ICRA (May 2010)
8. Hornby, G., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., Fujita, M.: Evolving robust gaits with aibo. In: IEEE International Conference on Robotics and Automation, pp. 3040–3045 (2000)
9. Hornby, G.S., Fujita, M., Takamura, S.: Autonomous evolution of gaits with the sony quadruped robot. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1297–1304. Morgan Kaufmann, San Francisco (1999)
10. Kim, M.S., Uther, W.: Automatic gait optimisation for quadruped robots. In: Australasian Conference on Robotics and Automation, Brisbane (December 2003)
11. Kobayashi, H., Osaki, T., Williams, E., Ishino, A., Shinohara, A.: Autonomous learning of ball trapping in the four-legged robot league, pp. 86–97 (2007)
12. Kohl, N., Stone, P.: Machine learning for fast quadrupedal locomotion. In: The Nineteenth National Conference on Artificial Intelligence (July 2004)
13. Quinlan, M.J., Chalup, S.K., Middleton, R.H.: Techniques for improving vision and locomotion on the sony aibo robot. In: Proceedings of the 2003 Australasian Conference on Robotics and Automation (2003)
14. Saggar, M., D'Silva, T., Kohl, N., Stone, P.: Autonomous learning of stable quadruped locomotion. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 98–109. Springer, Heidelberg (2007)
15. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press, Cambridge (2000)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning), May 1998. Mit Pr (1998)
17. Sutton, R.S., Mcallester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: Advances in Neural Information Processing Systems 12, pp. 1057–1063. MIT Press, Cambridge (1999)
18. Zagal, J.C., Ruiz-del-Solar, J.: Learning to kick the ball using Back-to-Reality. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 335–346. Springer, Heidelberg (2005)

Real-Time Active Vision by Entropy Minimization Applied to Localization

Stefan Czarnetzki, Sören Kerner, and Michael Kruse

Robotics Research Institute
Section Information Technology
TU Dortmund University
44221 Dortmund, Germany

Abstract. This paper presents an active vision approach to enhance mobile robot localization. A particle filter localization is extended with a module to find active vision decisions that are optimal based on the current localization and its uncertainty. Optimality is expressed as a criterion of entropy minimization. Further approximations are introduced to enable real-time computation. Both the usefulness of the presented approach in a RoboCup scenario and the performance and quality of the approximations are evaluated in different static and dynamic situations.

1 Introduction

Knowing their own position relative to the environment and certain points of interest is essential for mobile robots to autonomously achieve their tasks. Approaches focusing on localization exclusively are called *passive localization*, i.e. they address the position estimation based on an incoming stream of sensor data that can not be influenced. Solutions based on Kalman filters, particle filters and combinations and extensions thereof are numerous and common in this field. A comprehensive overview is given in [1]. *Active localization* instead implies the interaction of the localization process with the control of the robot. Examples for control or influence of the whole navigation process are coastal navigation algorithms where the localization uncertainty is anticipated and taken into account for path planning [2]. The more complex the robot's task is, however, the more problematic to include localization quality as an additional criterion into the general planning processes for achieving the given task. A common trade-off is to allow (partial) control over the information gathering process, e.g. viewpoint and orientation of directed sensing devices, while leaving the navigation decisions unaffected. In case of imaging sensors this belongs to the field of *active vision*.

Active vision in general refers to actuated camera systems with a task-oriented control of the camera movement. This might be staying on target or automatically choosing regions of interest for tracking or surveillance applications [3] or fixating on particular objects in remote collaboration scenarios [4].

Active vision for localization tasks means to move the camera with regard to the expected information gain. Therefore it is necessary to specify such information gain, to infer which observations would improve knowledge the most,

and estimate the result of different actions based on possibly multi-modal localization belief states. Previous work exists for different fields of application. The approaches depend on the range of possible actions, the types of observations and the representations of the localization belief [5,6,7].

This paper presents the application of an approach similar to [8] to a humanoid robot with directed vision in the context of a known environment with mainly ambiguous landmarks. First, the underlying localization is described briefly together with concepts for the measurement and estimation of localization quality to infer expected information gains. Modeling of the environment and sensors is necessary to reason about expected results of selected actions. Furthermore the algorithm to find optimal active vision decisions is described in detail including necessary discretization and adaptations for real-time processing on platforms with restrictive resources. Our approach differs from [8] in the underlying localization approach of a particle filter instead of a grid localization and the use of directed vision instead of an array of sonar sensors, which therefore results in different adaptations and approximations for an efficient implementation. Finally the performance both in static and dynamic situations is evaluated and a conclusion is given.

2 Modeling

The computation of an optimal active vision decision depends on the localization task and its belief representation, the actions to choose from and the estimation of their outcomes. The latter involves using knowledge about environment and sensors to first predict observations given a localization belief state and then predict their influence on said belief state. Those predictions obviously depend on the underlying localization approach and models of environment and sensors. This will be described briefly in the following sections.

2.1 Localization

The task of (passive) localization consists of the estimation of the robot's location relative to a known map given a stream of sensor and control information, y_1, \dots, y_t and u_1, \dots, u_t , respectively. Global in contrast to local localization assumes no a-priori knowledge about the initial position x_0 . The current belief state of a robot's localization is therefore given by

$$\text{bel}(x_t) = p(x_t | u_{1:t}, y_{1:t}) \quad (1)$$

and the posterior prior to incorporating the latest observation y_t , the so called prediction, is

$$\overline{\text{bel}}(x_t) = p(x_t | u_{1:t}, y_{1:t-1}). \quad (2)$$

Having to work on the whole collection of sensor and control data from all previous time steps results in complexity and needed memory increasing with time. The Markov assumption of a complete representation of the state x_{t-1} at

the last time step $t - 1$ allows the estimation of the current state x_t based on this last state and the current control input u_t only. The recursive state estimation of the Bayes filter is then given by

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \quad (3)$$

$$bel(x_t) = \frac{p(y_t|x_t) \overline{bel}(x_t)}{p(y_{t+1}|u_t)} = \eta p(y_t|x_t) \overline{bel}(x_t) \quad (4)$$

with η normalizing the belief to add up to 1. The state transition $p(x_t|u_t, x_{t-1})$ and the measurement probability $p(y_t|x_t)$ depend on the given task, in case of localization on the robot's odometry and kinematic model and its sensor model. The actual implementation of equations 3 and 4 depends on the representation of $bel(x_t)$.

In case of particle filters the belief is given as a set of particles or hypotheses $X_t = x_t^1, x_t^2, \dots, x_t^n$ representing the distribution

$$x_t^i \propto p(x_t|u_{1:t}, y_{1:t}). \quad (5)$$

2.2 Entropy of Belief States

To evaluate effects of possible actions it is necessary to specify criteria for the usefulness of their result. With respect to localization this refers to the quality of the belief state. Without knowledge about the robot's real position it is only possible to judge the belief state's uncertainty, i.e. the covariance matrix in case of an unimodal Gaussian distribution for Kalman filters. A common measure for this is the entropy.

Entropy $H(X)$ is defined as the expected information content

$$H(X) = - \int p(x) \log p(x) dx. \quad (6)$$

with $p(x) \log p(x) = 0$ for $p(x) = 0$. This allows to compute a single scalar value as a measure for the uncertainty of a given probability distribution. Obviously for a distribution which is non-zero only at a single point, i.e. a gaussian with zero variance or a Dirac delta function, the entropy $H(X) = 0$. For a uniform distribution on the other hand the entropy is maximal.

In case of a probability distribution given as a particle set the analogy of this concept to thermodynamic entropy is obvious. For computing it using equation 6 the value of $p(x)$ can be extracted from the local particle density relative to the overall number of particles.

3 Active Vision Decision Finding

Choosing an action in active vision means to choose where to point the camera. This is comparatively easy as long as the robot's (or the camera's) position is

known. A naive approach would be to choose a feature from the map, e.g. the most descriptive or the closest, and to point the camera in the direction of the expected position. While this might still work as long as the belief distribution is concentrated close to the true robot pose, it becomes less effective once the distribution is less certain or even multi-modal.

The following section describes the computation of an optimal active vision decision under uncertain belief distributions.

3.1 The Optimal Action

To choose an optimal action for a given localization belief state, it is necessary to evaluate the profit of every possible candidate. For every action the probable observations need to be computed, their influence on the belief state inferred and the result evaluated. The presented approach is based on and follows the outline of [8].

In general possible actions might be simple motor commands to position sensors, short motion sequences or complex actions like navigation commands to move the robot to certain positions. In the following every action u_i from the set of possible actions $U = u_1, u_2, \dots, u_m$ will be considered atomic and consequences will only be considered for the complete execution of an action. The notation of u is identical to the control data in section 2.1 since the following holds for robot actions in general. The specific application to active vision is referred to in sections 3.2 to 3.4. The presented approach always chooses the best action based on the immediate benefit and can thus be characterized as greedy.

Section 2.2 illustrates the relationship between entropy and uncertainty. The optimal action u_{opt} for a given belief X_t is the one minimizing the entropy $H(X_{t+1}|u_{opt}, y_{t+1})$ of the belief X_{t+1} after execution of u_{opt} and incorporation of the observation y_{t+1} resulting from this action (see equation 7).

$$u_{opt} = \operatorname{argmin}_{u \in U} H(X_{t+1}|u, y_{t+1}) \quad (7)$$

The terminology here is that only one observation is made at any time, but this single observation might include measurements of a various number of landmarks or features. This observation y_{t+1} , however, can only be predicted with certainty if the true position is known. An uncertain position given as a probability distribution only allows a decision based on the expected entropy of the next belief, as shown in equation 8.

$$\hat{u}_{opt} = \operatorname{argmin}_{u \in U} E[H(X_{t+1}|u)] \quad (8)$$

Computing this expected entropy necessitates the simulation of all possible observations y_{t+1} for a given action. Depending on the complexity of the observation's representation and the sensing process, generating the observations can be based on recorded data as in [5] or [6]. For laser or sonar sensors this might be done using the known environment model and ray-casting operations [7] or a simple visibility simulation for cameras.

Let $p(x)$ from equation 6 be represented by $bel(x)$ and let $p(y_{t+1}|u)$ be the probability of observing y_{t+1} after executing u . Then the expected entropy can be calculated as follows using equations 6 and 4:

$$\begin{aligned} E[H(X_{t+1}|u)] \\ = \int p(y_{t+1}|u) H(X_{t+1}|u, y_{t+1}) dy_{t+1} \end{aligned} \quad (9)$$

$$= - \iint p(y_{t+1}|u) bel(x_{t+1}) \log bel(x_{t+1}) dx_{t+1} dy_{t+1} \quad (10)$$

$$\begin{aligned} &= - \iint p(y_{t+1}|u) \frac{p(y_{t+1}|x_{t+1}) \overline{bel}(x_{t+1})}{p(y_{t+1}|u)} \\ &\quad \log \frac{p(y_{t+1}|x_{t+1}) \overline{bel}(x_{t+1})}{p(y_{t+1}|u)} dx_{t+1} dy_{t+1} \end{aligned} \quad (11)$$

$$= - \iint p(y_{t+1}|x_{t+1}) \overline{bel}(x_{t+1}) \log \frac{p(y_{t+1}|x_{t+1}) \overline{bel}(x_{t+1})}{p(y_{t+1}|u)} dx_{t+1} dy_{t+1}. \quad (12)$$

By inclusion of $\overline{bel}(x_{t+1})$ possible movements of the robot are already considered. If u includes motion controls then choosing those according to equation 12 results in navigation optimizing the robot's localization. If u includes only such commands without influence on the state x_t then those can be controlled to optimize localization without interfering with the robot's task-oriented navigation. When u has no influence on x_{t+1} as in active vision then $\overline{bel}(x_{t+1})$ is the same for all u and needs to be computed only once. In the case that the difference between x_t and x_{t+1} can be neglected, i.e. the robot's motion during this time is small, further simplification is possible resulting in equation 13 omitting calculation of the process model.

$$E[H(X_{t+1}|u)] = - \iint p(y_{t+1}|x_t) bel(x_t) \log \frac{p(y_{t+1}|x_t) bel(x_t)}{p(y_{t+1}|u)} dx_t dy_{t+1} \quad (13)$$

Actions may also vary in necessary effort so that their costs need to be considered. This is done by weighted addition to the term to be minimized in equation 8.

3.2 Discretization

The integrals in equations 12 or 13 can not be computed in closed form for any non-trivial applications. Discretization is necessary at certain points to allow the computation of those terms.

A first step is obvious for the application of a particle filter. Let $\bar{x}_{t+1}^i \in \overline{X}_{t+1}$ be a particle from the set X_t after applying u , then equation 12 becomes

$$\begin{aligned} E[H(X_{t+1}|u)] \\ = - \int \sum_{i=1}^n p(y_{t+1}|\bar{x}_{t+1}^i) p(\bar{x}_{t+1}^i|u) \log \frac{p(y_{t+1}|\bar{x}_{t+1}^i) p(\bar{x}_{t+1}^i|u)}{p(y_{t+1}|u)} dy_{t+1}. \end{aligned} \quad (14)$$

In this case $p(\bar{x}_{t+1}^i|u)$ can not be inferred from a single particle but from the local particle density around it. A working particle localization normally tends to result in few areas with high probability after a certain time. A further approximation might concentrate on those particle clusters and use average positions computed from these subsets instead of all particles separately which lowers the computational cost at least by an order of magnitude and provides the needed local particle density at the same time.

Discretization of the observations can not be motivated by the particle filter concept but depends on the sensor type and the possible observations to be made. For vision based perception those observations are commonly recognized landmarks. A complete enumeration of all possible observations would need to account for all landmarks with all possible distances and bearings which is clearly not practical. To avoid the generation of many observations whose measurement probability is near zero for the current belief an alternative is the simulation of observations for all position hypotheses given by the particle set or the chosen position discretization. This generates a number of observations in the order of the number of landmarks times the amount of position hypotheses. This discretization step results in

$$E[H(X_{t+1}|u)] = - \sum_{j=1}^m \sum_{i=1}^n p(y_{t+1}^j|\bar{x}_{t+1}^i) p(\bar{x}_{t+1}^i|u) \log \frac{p(y_{t+1}^j|\bar{x}_{t+1}^i) p(\bar{x}_{t+1}^i|u)}{p(y_{t+1}^j|u)} \quad (15)$$

or

$$E[H(X_{t+1}|u)] = - \sum_{j=1}^m \sum_{i=1}^n p(y_{t+1}^j|x_t^i) p(x_t^i) \log \frac{p(y_{t+1}^j|x_t^i) p(x_t^i)}{p(y_{t+1}^j|u)} \quad (16)$$

for the case that the robot's motion can be neglected.

Finally the actions to be evaluated in equation 8 need to be enumerated. In case of active vision these are motor commands for all possible pan and tilt angles to control the camera's field of view. Since it is not possible to process this search space in any more efficient way than to do a complete evaluation the chosen discretization is essential for real-time computation.

3.3 Decision Computation

Based on the considerations of the previous sections it is possible to compute the optimal action according to equation 8. In algorithm 1 the belief $bel(x_t)$ is given as a set of particles X_t and the optimal action \hat{u}_{opt} out of the set of possible actions U is calculated under the assumption of negligible robot motion according to equation 16.

Note that the expression of $\|\mathcal{X}_t^i\|/\|X_t\|$ for the term $p(\tilde{x}_t^i)$ is only valid for clusters representing equally sized volumes of state space, e.g. in case of a regular grid or similar clustering techniques, and when all particles are represented exactly once. The probability $p(y_{t+1}^k|\tilde{x}_t^i)$ in line 14 is the measurement probability in section 2.1 and $p(y_{t+1}^k|u)$ from line 15 is equivalent to the normalizing factor η in equation 4 and can be determined by

Algorithm 1. Active localization by entropy minimization

```

Input: particle set  $X_t$ , set of possible actions  $U$ 
1  $H_{min} = \infty$ 
2 Generate a set of clusters  $\mathcal{X}_t$  from  $X_t$ 
3 foreach  $\mathcal{X}_t^i \in \mathcal{X}_t$  do
4     Calculate average  $\tilde{x}_t^i$  of all  $x_t^j \in \mathcal{X}_t^i$ 
5 end
6 foreach  $u \in U$  do
7     foreach  $\mathcal{X}_t^i \in \mathcal{X}_t$  do
8         Generate observation  $y_{t+1}$  based on  $\tilde{x}_t^i$  and  $u$ 
9          $Y_u = Y_u \cup y_{t+1}$ 
10    end
11     $h = 0$ 
12    foreach  $y_{t+1}^k \in Y_u$  do
13        foreach  $\mathcal{X}_t^i \in \mathcal{X}_t$  do
14             $g = p(y_{t+1}^k | \tilde{x}_t^i) \cdot |\mathcal{X}_t^i| / \|X_t\|$ 
15             $h = h - g \log(g/p(y_{t+1}^k | u))$ 
16        end
17    end
18    if  $h < H_{min}$  then
19         $H_{min} = h$ 
20         $u_{opt} = u$ 
21    end
22 end
23 return  $u_{opt}$ 

```

$$p(y_{t+1}^k | u) = \sum_{i=1}^n p(y_{t+1}^k | \tilde{x}_t^i) bel(\tilde{x}_t^i). \quad (17)$$

This implies the advantage of precomputing all probabilities and their sum before computing the entropy, which was omitted here for clarity.

3.4 Adaptations for Real-Time Processing

The use of clusters instead of separate particles both for the generation of observations and the evaluation of their effect as introduced in the previous section already represents a first approximation. While this allows a computation an order of magnitude more efficient than without this approximation, several other adaptations and simplifications are possible to allow the application on robotic platforms with limited processing power such as the humanoid robot Nao (see section 4).

As long as relevant observations are only caused by static elements of the environment it is possible to pre-compute them to be stored in a lookup table for all robot states and relevant control commands. For 2D localization and pan and tilt controls for a camera this table has a dimension of 5 assuming an approximately constant height of the camera. Of course the limited resolution

in each dimension and the constant height approximation cause further loss of precision which might influence the result. This will be evaluated in section 4.

Similarly the measurement probability might also be pre-computed and stored in a lookup table. Additionally to the 5 dimensions mentioned above this table would need to consider the distance and bearing of the observation and in case of certain features like line crossings on a soccer field the feature's orientation is relevant, too. The resulting 8 dimensions would cause the table to exceed several 100 MB for useful resolutions though, so for embedded platforms with limited memory this is no practical solution.

As can be seen in figure 1 for a common situation the localization belief state often focuses on few areas of high probability. The clustering step in line 2 of algorithm 1 already causes the areas without particles to be neglected. This is valid since those areas correspond to quasi-zero probabilities. Omitting further areas with low but non-zero probability represents an approximation to equation 16 but considerably decreases the computational costs.



Fig. 1. Identification of regions with high particle density

4 Evaluation

The usefulness of this active vision approach to aid localization and the validity of the approximations proposed in section 3.4 are evaluated in several experiments presented in this section. The system used for those experiments is the humanoid robot Nao by Aldebaran Robotics which is used in the RoboCup Standard Platform League. This robot contains a x86 AMD Geode 500 MHz CPU and 256 MB SDRAM. The runtime necessary for algorithm 1 with and without further approximations is presented in table 1. The significant range in those measurements is a direct result of the dependency of computational complexity on localization uncertainty. If a localization belief is less certain more positions and consequently more possible observations need to be evaluated.

The following experiments are performed in a 3D simulation for easy access to ground truth information of the robot position and the repeatability of experiments based on the exact same conditions. The environment is a robot soccer

Table 1. Runtime of active vision module with and without approximations

	Maximum	Average	Minimum
Without Approximations	331.2 ms	38.5 ms	11.5 ms
Pre-calculated Observations	294.3 ms	35.6 ms	9.3 ms
Selective Computing	45.8 ms	4.2 ms	1.2 ms

field as used in Standard Platform League competitions. All processes running on the real robot are also executed in simulation including motion, image processing and cognition. Since all cognition processes run at 15 Hz the algorithm with pre-calculated observations and selective computing is suitable even for platforms with restricted resources like the Nao. Active vision is implemented to decide the target viewing direction, move the head and stare in that direction for one second, then continue to choose the next direction.

4.1 Static Situations

The discretization of possible actions used in all following experiments is 7 different pan angles and 2 tilt angles, one to point the camera at features close to the robot and one for far away observations. The choice is based on the camera's opening angles and slightly overlapping fields of view. This represents a minimal number of distinct actions. A more detailed resolution could provide better decisions, but this would imply higher computational costs.

The first experimental setup shows the validity of the active vision decision process and the applied approximations. Situations are generated which allow decisions that are easy to comprehend. First a robot is placed on the field near a penalty kick mark looking at the center circle until this repetitive exclusive perception generates two symmetrical clusters of position hypotheses. This situation might occur for a robot staring at a ball lying at the kick-off position. Figure 2(a) shows the expected entropy for different action choices with and without approximations. Figure 2(b) illustrates the average position error of all particles after executing each action and processing the resulting observations. The quantitative correlation of the values is clearly visible.

A similar situation occurs frequently in soccer games: When penalized players come back into the game they are placed on the side line in the middle of the field. The side is not known to the robot in advance. In the given particle filter localization this penalty information is handled by placing 40% of the particles near each re-entry spot and distributing the rest randomly to cope with wrong game controller handling. The results for this setup are shown in figure 3.

Additional experiments not described here in detail are set up to evaluate the localization performance starting from total uncertainty in different situations focusing on a comparison between active vision guided localization and the simple approach to move the head from side to side to continuously scan the environment. Those show that the passive approach of fast scanning from side to side is more effective when uncertainty is high. This is reasonable since high

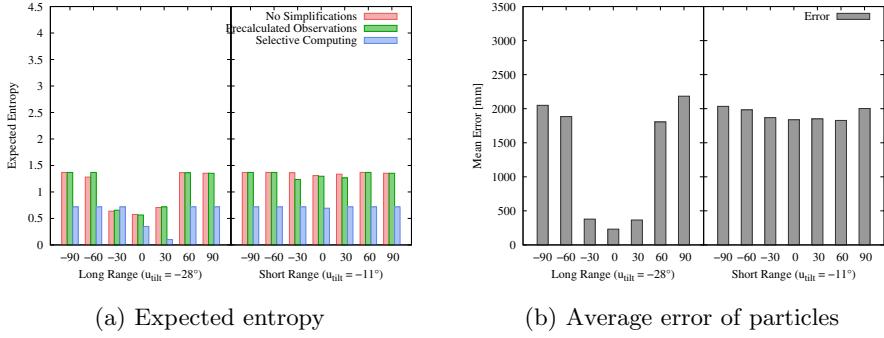


Fig. 2. The expected entropy and the real error of particles after executing an action on a penalty kick position facing the center circle

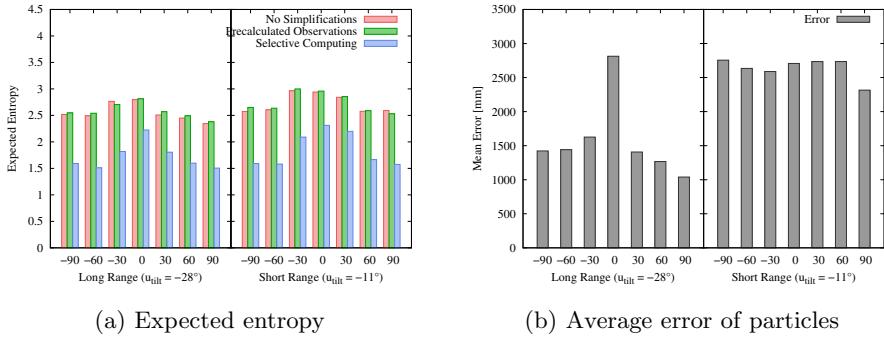


Fig. 3. The expected entropy and the real error of particles after executing an action on the removal-penalty's return position, i.e. on the side line facing the center circle

uncertainty deprives the reasoning attempt of its information to make useful decisions while scanning around for several different observations resolves the uniform distribution fastest. After different distinct position hypotheses can be deduced from the belief state the active approach tends to concentrate on the features providing most information optimally reducing the remaining uncertainty to result in a more precise position estimate.

4.2 Dynamic Situations

A final experiment places the robot on the sideline of the field with uncertain prior knowledge of the starting position to be on one of the side lines and the task to walk to a series of target positions (see fig. 4 and 5). Prior localization knowledge is provided since random particle distributions result in random active vision decisions as shown before. The benefit of the active approach is clearly visible especially when only few useful features are in front of the robot. In such

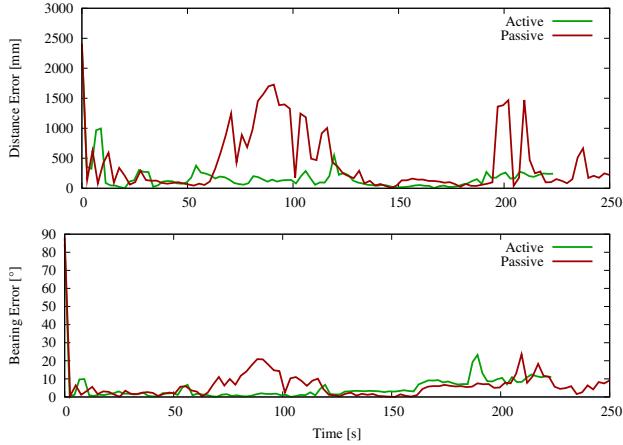


Fig. 4. Position errors (in average 172 mm and 418 mm) and orientation errors (in average 4.97° and 6.14°) of localization while walking to target positions on the field for the active and passive approach, respectively

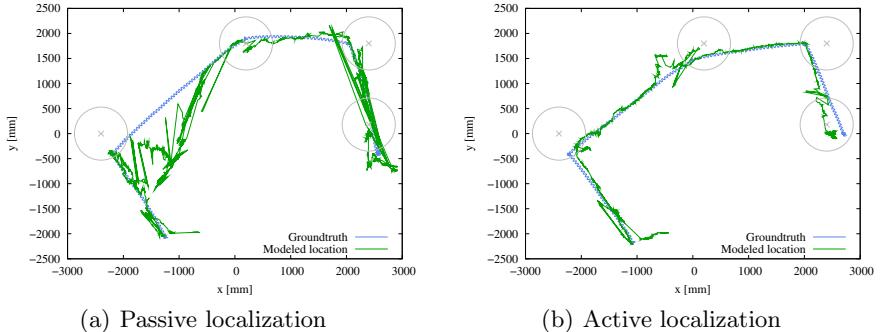


Fig. 5. Localization (green) versus true position (blue) of the robot while walking to a series of target positions

situations most of the scanning motion of the passive approach is wasted on featureless areas.

5 Conclusion

In summary active vision provides the possibility to improve localization in most common situations where the localization algorithm itself already provides some result. For situations of random particle distributions which can be detected with the same entropy criterion a fast scanning motion should be used instead to resolve the high uncertainty by many different observations instead of a few specific ones. Most importantly active vision is also possible and useful even

for platforms with low processing power like the Nao and in environments where relevant features are distributed uniformly as is the case for a Standard Platform League soccer field.

In future work this approach can be extended to also consider dynamic elements of interest like the ball or other robots. Since those are commonly not estimated together with the localization in a single particle filter a multi-modal optimization has to be applied to the decision problem instead of minimization of the entropy of a single particle set.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), 3rd edn. The MIT Press, Cambridge (2005)
2. Burgard, W., Fox, D., Thrun, S.: Active mobile robot localization. In: Proceedings of IJCAI 1997, Morgan Kaufmann, San Francisco (1997)
3. Hernández, M., Cabrera, J., Domínguez, A., Castrillón, M., Guerra, C., Hernández, D., Isern, J.: DESEO: An active vision system for detection, tracking and recognition. In: Christensen, H.I. (ed.) ICVS 1999. LNCS, vol. 1542, pp. 376–391. Springer, Heidelberg (1999)
4. Davison, A.J., Mayol, W.W., Murray, D.W.: Real-time localisation and mapping with wearable active vision. In: ISMAR 2003: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality, p. 18. IEEE Computer Society Press, Washington (2003)
5. Porta, J.M., Terwijn, B., Kröse, B.: Efficient entropy-based action selection for appearance-based robot localization. In: Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2842–2847 (2003)
6. Arbel, T., Ferrie, F.: Entropy-based gaze planning. In: Proceedings of the Second IEEE Workshop on Perception for Mobile Agents (1999)
7. Fairfield, N., Wettergreen, D.: Active localization on the ocean floor with multibeam sonar. In: Proceedings of MTS/IEEE OCEANS (2008)
8. Fox, D., Burgard, W., Thrun, S.: Active markov localization for mobile robots. *Robotics and Autonomous Systems* 25, 195–207 (1998)

Multi-agent Behavior Composition through Adaptable Software Architectures and Tangible Interfaces

Gabriele Randelli¹, Luca Marchetti¹,
Francesco Antonio Marino², and Luca Iocchi¹

¹ Department of Computer and System Sciences,
Sapienza University of Rome,
Via Ariosto 25, 00185 Rome, Italy

`name.lastname@dis.uniroma1.it`

² Mathematics Department,
Tor Vergata University in Rome,
Via della Ricerca Scientifica snc, 00133 Rome, Italy
`marino@mat.uniroma2.it`

Abstract. Cooperative behavior realization is an important aspect of Multi-Agent Systems, and it has been widely addressed by the robotics community. However, the interaction among multiple robots and a human operator still requires a non-negligible effort to be effective. The availability of modern input devices can ease the behavior composition process, allowing a user to train Multi-Agent Systems by using alternative methods. In this paper, we introduce a novel approach to integrate Tangible User Interfaces within a reconfigurable software architecture, for cooperative Multi-Robot Systems. To address a real test case, we implemented a strategy training system for a humanoid RoboCup soccer team. Such a system allows a human coach to train several robotic players by using multiple input interfaces, directly onto the application field.

Keywords: tangible user interfaces, multi agent systems, humanoid robots, behavior composition, system architectures.

1 Introduction

Behavior composition is a key task for multi-robot systems in several applications, such as RoboCup soccer or *Urban Search And Rescue*. Because of the intrinsic complexity of this problem, the role of human operators in defining complex behaviors is often fundamental, and it is cognitive demanding. In fact, managing several robots through traditional user interfaces is neither comfortable nor scalable. On the one hand, interfaces are typically overwhelmed by a huge amount of data, thus preventing from an easy assessment. As well as innovative interfaces, another benefit consists of distributing this process among different operators. This requires a novel class of robotic software architectures

for supporting a multi-operator/multi-robot paradigm. The main requirement of such a system is adaptability, in order to manage multiple operators with different input devices, acting in different scenarios with different robotic platforms. Furthermore, flexibility is also crucial, to dynamically tune the system according to run-time changes in the robotic team, in the environment, or in the task to accomplish.

In this paper, we propose a new methodology for the problem of behavior composition in Multi-Agent Systems, through the use of Tangible User Interfaces and designing an adaptable and flexible software architecture to integrate their usage with different robotic platforms and other input devices. Furthermore, we provide a very preliminary evaluation of our system, which will be properly discussed and extended and in some future work.

A *Tangible User Interface* (TUI), is a user interface in which a person interacts with digital information through the physical environment. TUIs provide a high-level interaction paradigm, for example, through natural gestures. Furthermore, they are more comfortable with respect to traditional GUIs, as do not force the operator to switch her attention from the ongoing mission to the interface. Our objective is to develop a methodology to define behaviors for multi-agent systems through a Wiimote device, which allows for a direct interaction with the robots on the field.

We do aim at integrating TUIs with several other input devices, and interacting with heterogeneous platforms in heterogeneous environments (both real and simulated), without reconfiguring the whole system when a change occurs. We designed a flexible architecture that supports classic input methods, as well as innovative ones, and that codifies, through a well defined semantics, every input command into actions for different platforms: real wheeled robots, real humanoid robots, as well as simulated environments, such as Player/Stage¹, USARSim² and Webots³.

The remainder of the paper is structured as follows. In Section 2, we address the problem of behavior composition for multi-agent systems and present some related works about tangible user interfaces. Then, in Section 4 we introduce the Wiimote device and our interaction system, while Section 5 describes our high-level command architecture. Finally, in Section 6 we evaluate the overall system implementing a strategy virtual coaching application for a humanoid RoboCup soccer team, and we conclude with some discussion in Section 7.

2 Related Work

Tangible user interfaces take advantage from human spatial orientation and interaction with tangible objects, two fundamental skills for human interaction in everyday activities. In fact, cognitive studies [5] have assessed how gesturing, as well as speeching, is one of the first learnt communication means (at around 10

¹ <http://playerstage.sourceforge.net>

² <http://usarsim.sourceforge.net>

³ <http://www.cyberbotics.com>

months of age). Concerning a comparison of TUIs with respect to classic input methods, one interesting study is by Guo and Sharlin [3], who evaluated the performance of a Wiimote TUI and a classic keypad through navigation and posture tasks, using a Sony AIBO robot. The results evinced that a tangible device reduces the user thinking time for the correct movement, since they correspond to innate actions performed everyday. However, this does not imply that a TUI always guarantees a better performance with respect to other input devices. Analogous results are presented in the work by Song et al. [8], where they compared four types of devices: a pad-like device, a joystick, a driving device, and a Wiimote equipped with motion sensors. Varcholik et al. [9] address how TUIs provide two novel interaction methods: motion steering-wheel style control or gestural control. The former consists of using the sensors feedback to directly control the locomotion of the robot, while the latter recognizes gestures and associates them to high-level commands through machine learning and pattern recognition techniques. As for this latter, Mlích [6] defined a gesture recognition algorithm based on hidden Markov models that detects basic gestures, such as circles, squares, and so on. However, all these works present a single human-single robot interaction paradigm (SHSR) and, to our knowledge, TUIs have never been adopted with multi-agents systems.

3 Problem Space

A critical component for the success of a Multi-Agent System is the ability to design and implement complex behaviors that involve all the robotic agents.

To achieve an effective result for a team of agents, it is important to identify a global target that should be pursued by the team itself. The global goal should be decomposable in sub-problems, each one carried on by a single agent. The role of the human designer is to identify the intrinsic characteristics of a global task, and then to develop enough sub-tasks to be assigned to the team player.

We assume that such a process of problem analysis is already done, and it is available, to the human operator, as a set of simple behaviors that can be assigned to the agents.

Giving a real example, let us consider the scenario of a robotic soccer team. The global goal of the team is to win the match. If the rules of the match are similar to the real soccer ones, we can identify distinct roles for the players. There would be a *goalie*, some *defenders* and some *attackers*. A robotic coach should, then, dynamically assign a role to each robot, depending on the situation. In autonomous robotic teams the role assignment could be done by the robot itself, directly on board.

How robots interact each other, which areas have to be covered by which robot, are complex tasks that require the interaction with the human operator. In most cases, this is done by trying different configurations of role assignment and covered areas. At the end of several trials, the user can encode many tactics inside some sort of structure (robot-friendly) and the robot-agent will select the most appropriate one.

The problem of defining the interaction between human operator and a team agent is the main focus of this paper. We believe it is possible to relieve the burden of the user to interact with complex languages for defining tactics.

Therefore, the assumptions we made on this paper are:

- a number of predefined and simple behaviors for single agent are already available;
- it exists some kind of languages that can be used for composing “atomic” behaviors;
- the system can select multiple behaviors and check if there is any consistency problem (it should verify if there is any incompatibility among selected behaviors);
- the user can execute a single configuration of the composed behaviors and be able to compare the results.

4 Wiimote Interaction System

4.1 Wiimote Technical Characteristics

The TUI we adopted in this paper is a Wiimote [11], reported in Figure 1. It is a commercial-off-the-shelf (COTS) inexpensive remote input device, which provides innovative sensing and feedback functionality, and communicates through the Bluetooth protocol architecture. The Wiimote is equipped with eleven buttons, two types of sensors, and three feedback controllers. As for the sensors, there is a three-axis ADXL330 linear accelerometer that measures the acceleration in a free fall frame of reference, providing the device with motion sensing capabilities. As well as the accelerometer, the device has an IR camera that acquires beacons coming from a Sensor Bar with ten infrared LEDs, five at each end of the bar. The Sensor Bar allows to use the Wiimote as a pointing device, since through triangulation methods it is possible to retrieve the distance between the Sensor Bar and the Wiimote, and to track the position onto the screen where the device is pointing. Concerning the feedback functionality, the Wiimote incorporates four blue LEDs, a rumble feature, and a small low quality speaker which emits short sounds. It is also provided an expansion port at the bottom, to plug in extension controllers, such as the Wii Motion Plus.

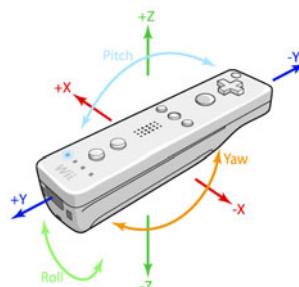


Fig. 1. The Wiimote device with its body-fixed coordinate system

4.2 Wiimote Configuration for Behavior Composition

Our proposed methodology matches the objectives evinced in Section 3 with the Wiimote functionality and limits. We aim at using the Wiimote to create behaviors by directly interacting with robots, without any robot interface in the middle. Such an approach allows the operator to use natural gestures, and, in case of real platforms, to move within the environment where the robots are deployed. Furthermore, through multi-modal feedback, such as rumble or sounds, the operator can look only at the environment, and not at the Wiimote, hence lowering her cognitive fatigue.

The Wiimote offers several types of interaction. For example, thanks to its particular design, it can be used as a pointer to focus on a specific point in the space, or just to select a robot. Moreover, through its motion sensing capabilities, it is possible to associate gestures to high-level commands to control the robotic team. However, the Wiimote embedded sensors give rise to several challenges in localizing precisely the device itself. As an example, accelerometers can sense rough motion, but they are not sufficient to retrieve the full attitude of the device within the space (in particular, with respect to a free fall coordinate system, it is not possible to detect the yaw of the device). A possible solution would be to use the IR camera. However, this must be used in combination with a Sensor Bar, hence forcing the operator to point the device against a specific part of the environment. Sko and Gardner [7] overcome this problem by installing and managing multiple Sensor Bars in a smart environment. However, this approach requires to cable the whole environment, which is something that does not favor the adaptability or the pervasiveness of the system. Our solution extends the Wiimote functionality using the Motion Plus controller, which contains two gyro sensors that report the three angular rates of the device. This allows to track the device attitude over time, without constraining the operator movements, as in the case of the IR camera, and without any change in the environment. This configuration does not use neither accelerometers nor the IR camera, hence its drawback is to track only the Wiimote attitude, but not its position within the environment. Thereby, the operator cannot perform gestures while walking, but she must stand in a fixed spot of the space.

4.3 Wiimote Interaction System

In order to use the Wiimote and Motion Plus extension for behavior composition, we designed an interaction system with the following objectives:

- supporting multiple Wiimote devices;
- integrating the Wiimote to our robotic system and, in particular, to the adaptable command architecture;
- extracting and tracking high-level information from the raw data coming from the Wiimote sensors.

The interaction system component diagram is reported in Figure 2. Our first component is a library to manage the Bluetooth communications with Wiimotes,

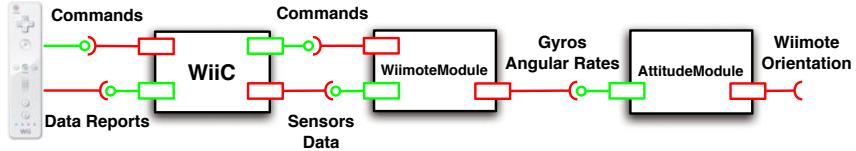


Fig. 2. The Wiimote interaction system UML component diagram

and their input/output messages. We extended Wiiuse, an open-source C library, realizing our own library, *WiiC*⁴. To our knowledge, *WiiC* is the only C library fully portable on Linux and Mac platforms, which supports Motion Plus and Balance Board extensions. As well as using *WiiC* for standalone application, it is also supported by the *OpenRDK*⁵ framework for robotic applications [1], through the wrapper module *WiimoteModule*. This allows to easily integrate the Wiimote support to our robotic system, which is fully designed via such a framework. Moreover, *WiimoteModule* exhibits high-level functionality through a user-friendly graphical interface.

Once connected, the Wiimote periodically reports the status of its sensors. More formally, gyros transmit roll, pitch and yaw rates with respect to the Wiimote-fixed coordinate system W , represented at time t with the vector $\omega^W(t) = [\omega_{pitch}(t), \omega_{roll}(t), \omega_{yaw}(t)]^T$. It is worth mentioning that pitch and roll are inverted because, according to the Wiimote reference system, roll is a rotation around the y-axis, while pitch represents a rotation around the x-axis. Our purpose is to track the device orientation with respect to an absolute reference system O , hence we must solve an inverse differential kinematics problem. This is accomplished through the *AttitudeModule* component.

The attitude of the Wiimote is given by the quaternion vector $q(t) = [q_0(t), q_1(t), q_2(t), q_3(t)]^T$. We adopted quaternions to deal with singularities, since they are a more robust representation than Euler angles. Given the quaternion vector, $q(t)$, the following relationship represents the time derivative with respect to the angular velocity given in the body frame $\omega^W(t) = [\omega_{pitch}(t), \omega_{roll}(t), \omega_{yaw}(t)]^T$.

$$\dot{q}(t) = \frac{1}{2}\Omega(t)q(t), \quad \Omega(t) = \begin{bmatrix} 0 & -\omega_{pitch}(t) & -\omega_{roll}(t) & -\omega_{yaw}(t) \\ \omega_{pitch}(t) & 0 & \omega_{yaw}(t) & -\omega_{roll}(t) \\ \omega_{roll}(t) & -\omega_{yaw}(t) & 0 & \omega_{pitch}(t) \\ \omega_{yaw}(t) & \omega_{roll}(t) & -\omega_{pitch}(t) & 0 \end{bmatrix} \quad (1)$$

The integration process has been implemented using the *classical Runge-Kutta method* (RK4). Finally, in order to have an easier interpretation of the attitude we convert back from the quaternion representation to Euler angles.

Let the Euler angle vector $\Theta^O = [\phi, \theta, \psi]^T$ be the attitude of the Wiimote at time t , with respect to the coordinate system O . Then, using the following equations, pitch, roll and yaw angles can be obtained in the absolute frame:

⁴ *WiiC* is available at <http://wiic.sf.net>

⁵ <http://openrdk.sourceforge.net>

$$\begin{cases} \phi = \arctan\left(\frac{2(q_3q_3+q_1q_1)}{q_0^2-q_1^2-q_2^2+q_3^2}\right) \\ \theta = \arcsin(-2(q_1q_3 - q_1q_2)) \\ \psi = \arctan\left(\frac{2(q_2q_2+q_1q_3)}{q_0^2+q_1^2-q_2^2-q_3^2}\right) \end{cases} \quad (2)$$

Since gyros are quite noisy, the *AttitudeModule* filters the incoming angular rates, before integrating them, using a *moving average filter* [10]. However, in case of orientation tracking over long time intervals, such a filtering method is not enough efficient and it is necessary to adopt some control technique, such as a Kalman Filter, as proposed by Luinge and Veltink in [4]. The Wiimote attitude tracking can be useful in several different applications, and we will provide a practical example in Section 6, using it as a pointing device.

5 High-Level Command Architecture

The next step towards a more user-friendly experience in MAS behavior composition requires an architecture to support it. In this Section, and in the following ones, the term “behavior composition” indicates the result of user interaction with a library of simple robotic behavior. The TUI device can select a sequence of simple behaviors, or select a predefined complex one (designed off-line by the user). In Section 6, we defined a case study for behavior composition.

Let us start by analyzing the requirements for building a general-purpose intercommunications architecture. We can identify the following features:

- support for multi communications protocols;
- support for interactive input devices (joypad, keyboard, speech recognition, and so on);
- integration with multiple output devices (robotic platforms, sound synthesis, automatic signaling devices);
- capabilities to adapt at runtime to different devices.

Given these specifications, we designed a complex framework that allows human operators to interact with Multi-Agent Systems. The software framework implemented is built upon *OpenRDK*, an open-source framework for robotic applications development [1]. OpenRDK is a complete framework for realizing effective robotic applications. It focuses on modularity and re-usability, defining some tools for designing software agent. Each *RAgent* is built combining *modules* that express the functionality of the agent itself. The bear of communication methods, properties sharing, as well as low-level integration with robotic platforms are relieved by the developer: the focus of robot programming is on algorithm designing for real problem solving. This framework is successfully used in different contexts, such as robotic platforms for *Urban Search And Rescue (USAR)*, robotic soccer (*Nao Humanoid*), UAVs.

Within such framework, we designed an adaptable architecture as shown in Figure 3.

The *CmdInterpreter* component is an OpenRDK Module. It declares some basic properties for adapting its behavior to several input and output platforms.

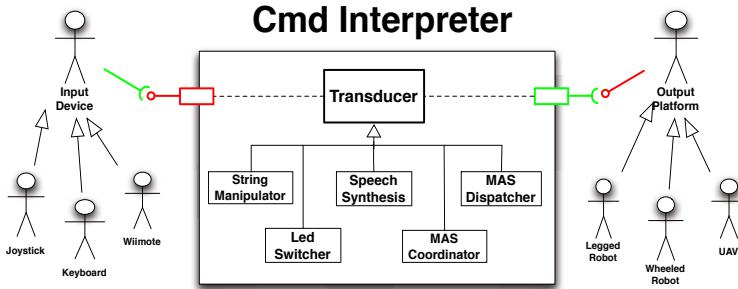


Fig. 3. A UML representation of our adaptable architecture

The *Input* module is the category wherein multiple input devices are declared: it includes, but is not limited to, TUIs (such as Wiimote), joypad and joystick, keyboard interfaces. The *Output* module includes all our supported categories of robotic platforms, real or simulated.

The interesting part is the block named *Transducer*. Inside the module, the *Transducer* component defines a hierarchy of classes, used to interface with input devices. Each class is responsible of creating support properties for handling a specific input device. Moreover, at runtime, it exposes itself within the system architecture.

A typical execution of the system is the following:

1. The *ragent* program loads all the modules defined as main components of the application.
2. The *CmdInterpreter* module loads all the transducers within a prototype database.

This will be used to instantiate at runtime the appropriate handler related to the application. In fact, inside this module it is possible to select, either before or during the execution of *RAgent*, which transducer to be loaded: the module selects the right class prototype and instantiate the relative transducer.

3. The *CmdInterpreter* module calls the init for a selected transducer.

At this stage, all input and output variables are created and the *EventHandler* is declared.

4. The full *RAgent* is up and running and it is ready to accept input from the user.

The behavior of *CmdInterpreter* module is dependent on the loaded transducer: the logic of command interpretation is delegated to this minimal component. Currently, we implemented transducer for string manipulation (*StringManipulator*), audio and visual feedback (*LEDSwitch* and *SpeechSynthesis*), MAS interaction (*MASCoordinator*, *MASDispatcher*).

Given to the modularity characteristic of OpenRDK framework, it is possible to combine several *CmdInterpreter* modules, to implement a complex command

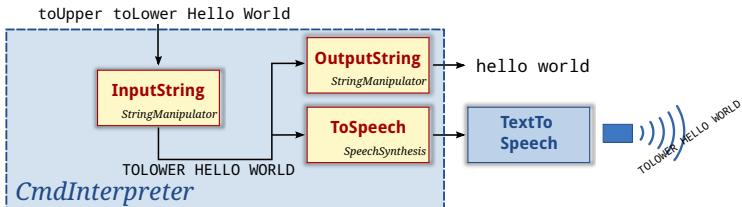


Fig. 4. An example of the polymorphic characteristics of our architecture: two *StringManipulator* and a *SpeechSynthesis*

chain. In Figure 4, it is illustrated an example that combines three different instances of *CmdInterpreter* module.

The *InputString* has an input property that takes a complex sentence as input. The user can insert basic commands for string manipulation (for example, “toLowerCase” and “toUpperCase”). Let assume she inserts “toUpperCase toLowerCase Hello World”. The output string property will be “TOLOWER HELLO WORLD”.

This is propagated to *OutputString* and to *ToSpeech*. The *OutputString* transducer will decode the input string in the same way *InputString* already did: the resulting output will be “hello world”. The *ToSpeech* transducer, instead, will propagate the sentence directly to the *TextToSpeech* module that simply plays the whole sentence. The user should hear: “tolower hello world”.

Despite this simple example shows a very trivial case, it presents some of the key features of the designed software architecture:

1. different transducers can be implemented limiting the programming effort;
2. given their input and output are compatible, they can be used in cascade;
3. they can share some input data each other;
4. multiple inputs and outputs can be defined.

Then, providing a transducer dispatcher (not shown here for simplicity sake), a chain of transducer can handle multiple input devices towards multiple output devices.

6 A Practical Example: Virtual Coaching

To prove the effectiveness of our software architecture, we developed a prototype application for Virtual Coaching. The objective of the system is to compose complex behavior for Multi-Agent Systems, such as game tactics, by using a user-friendly interface. In this scenario, a human operator combines tactics for a soccer team of robotic players. The soccer field is divided into several zones and each one is assigned to a single robot. The operator selects one area for each robot, simulating the placement onto the field. The tactic is, then, built by imposing a distinct role for each player. Since we are using real soccer rules, simple role behaviors are distinguished among: GoalKeeper, Defender, Supporter and Attacker.

6.1 System Overview

The overall system architecture is depicted in Figure 5.

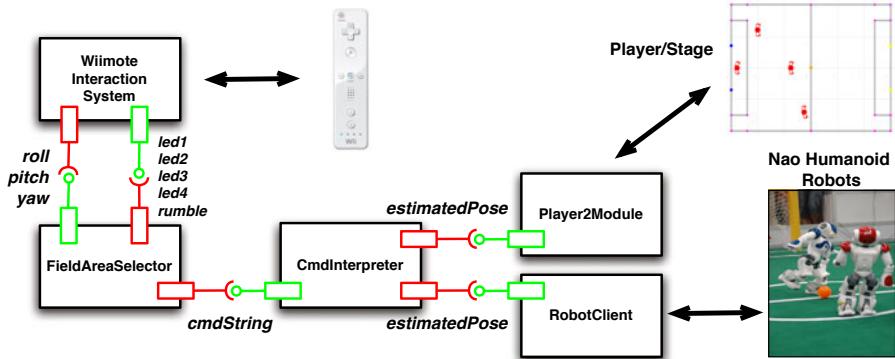


Fig. 5. The proposed architecture for Virtual Coaching

The core of the architecture is the *CmdInterpreter* module that uses a *MAS-SoccerPlayer* transducer. The *MASSoccerPlayer* component is defined by:

Input. a single string command, where the areas selected by the user are encoded;

Output. a *TargetPose* for each robot player.

A coordination routine is responsible for selecting the appropriate player for a selected area.

As input for the system, we have utilized the Wiimote, configured with the Motion Plus extension, as explained in Section 4.

For the output, we have used the Nao humanoid robot, currently adopted in the RoboCup Standard Platform League (SPL). To speed up the experimental results, we designed a Player/Stage simulated world [2], modeled upon the SPL rules. Commands, decoded by the *CmdInterpreter* module, are propagated both to the real robots and to the simulated one.

The complete data flow for the described application is shown in Figure 6(a).

First of all, the user calibrates the Wiimote. Then, she selects four areas. The positions of the Wiimote are translated into field area selections, and then a message to the *CmdInterpreter* module is sent. At this time, the areas are translated into field coordinates and the coordination procedure is called. It is worth mentioning that the system also validates whether the selected target points correspond to free areas within the field, hence no outer parts of the field nor obstacles can be selected. The last step involves sending the target positions to the corresponding robots. A rumble feedback is sent to the Wiimote to warn the user of successful execution of the command. The complete Finite State Machine of the user interaction is given in Figure 6(b).

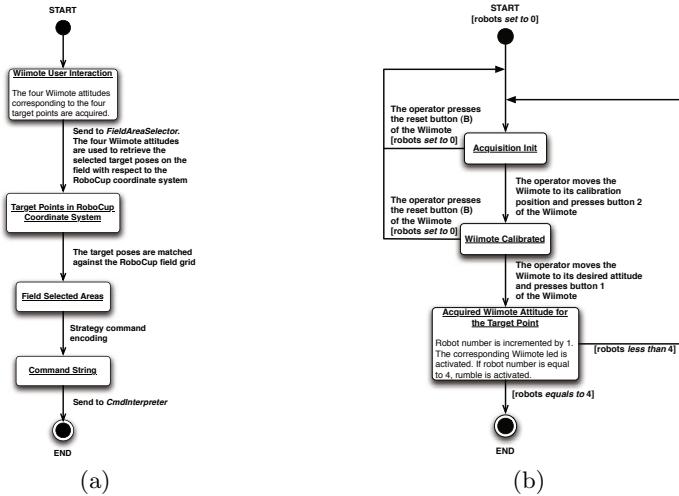


Fig. 6. UML state diagrams for the Virtual Coaching application 6(a) and the Wiimote user interaction 6(b)

6.2 Experimental Results

Experiments have been conducted to establish the correctness of the proposed architecture. They are not intended to be exhaustive, since they lack a strong validation formalism, which will be considered and evaluated in a future work.

Given the limitation of the devices, we defined the following assumptions. In order to correctly determine selected areas, the user should define the approximate height of the Wiimote from the ground. Besides that, we assume that the user selects the points by standing at the center sideline of the field. This information can be changed at runtime, and does not affect the accuracy of the system. Moreover, we discretized the field into nine distinct areas. We have determined that the system has an accuracy of $0.4m$. This means, the architecture is not able to distinguish areas whose size is less than $0.16m^2$. This is due to the noise in the sensor readings, and it determines a lower bound for the accuracy that can be achieved. However, this value is still acceptable to be used in real applications, and we proved that by developing the Virtual Coaching example.

Preliminary experiments have been conducted by using a real RoboCup SPL field. We compared the execution of the Virtual Coaching by using three different environment: text-based, with a visual interface, and using a TUI. In each scenario, we used the adaptable architecture developed so far, but adopting different input interactions. In the *TextOnly* scenario, the user has to insert the command by sending a string using a telnet-like program. In the *GUI* scenario, the operator interacts with the *RConsoleQT* program, provided by the *Open-RDK* distribution. Finally, in the *TUI* scenario, the operator uses a Wiimote connected to the system.

In these experiments, we also evaluated the characteristics of the scenario and the constraints the users have to perform same actions with different input. The Table 1 resumes the expected features emerged by performing such qualitative experiments.

Table 1. Qualitative results for Virtual Coaching experiments

Scenario	User Interaction	Command Structure	Feedback Visual	Feedback Physical	Access to the Environment
TextOnly	text	string	✗	✗	✗
GUI	point&click	button/mouse	✓	✗	✗
TUI	tangible	button/gesture recognition/ sensor motion	✓	✓	✓

7 Conclusions

In this paper, we analyzed the problem of human-robot interaction for Multi-Agent System behavior composition. We focus our attention on how to ease the operator effort to control a team of robotic player. The objective of the operator is to give high-level commands to the robots, by assigning specific tasks to them. We proposed the integration of Tangible User Interfaces, as means for giving such commands. Therefore, we presented a system software architecture for integrating TUIs in a MAS environment. The developed architecture is able to adapt to multiple and different input devices. Moreover, it is able to control several robotic platforms, as well as software simulators.

As practical test case, we presented a Virtual Coaching application, that integrates a Wiimote device and SPL humanoid robots. The conducted experiments validate the effectiveness of our developed architecture, even if they showed some limitations and the evaluation process is still at a preliminary stage. As future improvements, we need to increase the robustness of the system, by relaxing the starting point assumptions. A more robust integration over time of the acquired data can improve the self-localizing abilities of the Wiimote. More in depth experiments on real robots have to be done. Moreover, the system needs a more formal validation of the human-robot interaction, by performing quantitative experiments through user testing and validation.

References

1. Calisi, D., Censi, A., Iocchi, L., Nardi, D.: OpenRDk: a modular framework for robotic software development. In: Proc. of Int. Conf. on Intelligent Robots and Systems (IROS), pp. 1872–1877 (September 2008)
2. Gerkey, B.P., Vaughan, R.T., Howard, A.: The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proc. of the Int. Conf. on Advanced Robotics (ICAR 2003), Coimbra, Portugal, June 30-July 3, pp. 317–323 (2003)

3. Guo, C., Sharlin, E.: Exploring the use of tangible user interfaces for human-robot interaction: a comparative study. In: CHI 2008: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, pp. 121–130. ACM, New York (2008)
4. Luinge, H., Veltink, P.: Measuring orientation of human body segments using miniature gyroscopes and accelerometers. Medical and Biological Engineering and Computing 43(2), 273–282 (2005), <http://doc.utwente.nl/61405/>
5. Messing, L., Campbell, R.: Gesture, speech, and sign. Oxford University Press, Oxford (1999)
6. Mich, J.: Wiimote gesture recognition. In: Proceedings of the 15th Conference and Competition STUDENT EEICT 2009. Faculty of Electrical Engineering and Communication BUT, vol. 4, pp. 344–349 (2009), http://www.fit.vutbr.cz/research/view_pub.php?id=8933
7. Sko, T., Gardner, H.: The wiimote with multiple sensor bars: creating an affordable, virtual reality controller. In: CHINZ 2009: Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction, pp. 41–44. ACM, New York (2009)
8. Song, T.H., Park, J.H., Chung, S.M., Hong, S.H., Kwon, K.H., Lee, S., Jeon, J.W.: A study on usability of human-robot interaction using a mobile computer and a human interface device. In: MobileHCI 2007: Proceedings of the 9th International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 462–466. ACM, New York (2007)
9. Varcholik, P., Barber, D., Nicholson, D.: Interactions and training with unmanned systems and the nintendo wiimote. In: Proceedings of the 2008 Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) (2008)
10. Wei, W.W.: Time Series Analysis. Addison-Wesley, Reading (2006)
11. WiiBrew (2009), http://wiibrew.org/wiki/Main_Page

A Novel Real-Time Local Visual Feature for Omnidirectional Vision Based on FAST and LBP

Huimin Lu, Hui Zhang, and Zhiqiang Zheng

College of Mechatronics Engineering and Automation,
National University of Defense Technology, Changsha, China
`{lhmnew, hui_zhang_nudt, zqzheng}@nudt.edu.cn`

Abstract. A novel real-time local visual feature, namely FAST+LBP, is proposed in this paper for omnidirectional vision. It combines the advantages of two computationally simple operators by using Features from Accelerated Segment Test (FAST) as the feature detector and Local Binary Patterns (LBP) operator as the feature descriptor. The matching experiments of the panoramic images from the COLD database are performed to determine its best parameters, and to evaluate and compare its performance with SIFT. The experimental results show that our algorithm performs better, and features can be extracted in real-time.

1 Introduction

In comparison with global visual features, local visual features have better discriminative power, and are more robust to occlusion. Furthermore, good local visual features can be invariant to image rotation, image translation, image scale, changes of view, and even changes of illumination. Thus local visual features have become increasingly popular in recent years, and they have been applied very well in many computer/robot vision problems, such as image retrieval, image stitching, wide baseline matching, object recognition, place recognition, texture recognition, robot localization, and robot navigation. Local visual feature algorithm consists of feature detector and feature descriptor, and lots of algorithms have been proposed. In feature detector, Harris [1], Susan [2], DOG [3], MSER [4], Harris-Laplace [5], Harris-Affine [5], Hessian-Affine [5], Features from Accelerated Segment Test (FAST) [6], etc. have been designed; in feature descriptor, SIFT [3], SURF [7], GLOH [8], Local Binary Patterns (LBP) [9], etc. have been used to calculate the feature vectors over the feature region. Many researchers have done lots of work on evaluation and comparison of these algorithms [8][10][11][12], while relative source codes (or binaries), and many image databases have been released for further evaluation with new algorithms.

Although local visual features have so many advantages, a common deficiency for most of the existing algorithms is that their computation costs are usually high. This deficiency limits the actual application of local visual features, especially in those situations with high real-time requirements. Therefore several improved versions of the above algorithms have been proposed to accelerate the

feature detection or description. Fast approximated SIFT is presented in Ref. [13]. Compared to standard SIFT, it uses a box filter to calculate the DoM (Difference-of-Mean) images efficiently based on integral images. The key-points can then be detected, and the descriptor is also accelerated by using an integral orientation histogram. The experiments show speed increases by a factor of eight while the performance is only slightly decreased. In the iterative SIFT [14], the number of features can be defined in advance, so the process of searching the key-points continues iteratively without the need for sequentially going through the whole scale space. When it is applied in robot's localization, the computation effort of the feature extraction and matching process can be reduced as much as possible while high localization accuracy can be maintained. SURF [7] also takes the advantage of integral images. In feature detector, SURF approximates second order Gaussian derivatives with box filters, and image convolutions with these box filters can be computed rapidly by using integral images. In feature descriptor, Haar wavelet responses, which also can be quickly computed via integral images, are used to construct the descriptor vector. The experimental results show that the performance of SURF outperforms its competitors.

Because omnidirectional vision can provide a 360° view of the robot's environment in an image, it has become more and more popular as a visual sensor for robots. In this paper, we will propose a novel real-time local visual feature for omnidirectional vision which can be applied in the actual engineering problems with high real-time requirements. Features from Accelerated Segment Test (FAST) [6] will be used as the feature detector, and Local Binary Patterns (LBP) [9] as the feature descriptor, so an algorithm named FAST+LBP will be designed. The panoramic images in the COLD database [15] will be used to test our algorithm. The following sections are organized as follows: FAST and LBP are introduced in section 2; our FAST+LBP is proposed in section 3; the best algorithm parameters are determined by experiments, and the performance of FAST+LBP is evaluated and compared with standard SIFT in section 4; section 5 is the conclusion of this paper.

2 FAST and LBP

2.1 FAST

The corner feature is defined in FAST detector by the following Segment-Test algorithm: If more than N contiguous pixels in a Bresenham circle of radius r around a center pixel p are all brighter than p by some threshold or all darker than p by some threshold, there is a corner feature at p . Machine learning is utilized to speed up this corner detection process. Every pixel has 16 attributes corresponding to the 16 pixels in the Bresenham circle (for $r = 3$), and each attribute can be 0, 1 or -1. If a pixel with position x on the circle of p is brighter(darker) than p , the corresponding attribute is 1(-1). Otherwise, the attribute is 0. A decision tree can be learned by using ID3 to select the pixels in the circle which yield the most information about whether the center pixel is a corner. So a pixel can be classified as a corner feature or not more efficiently,

which means the Segment-Test algorithm is accelerated. This decision tree is then converted into C-code, creating a long string of nested if-then-else statements which is compiled and used as a corner detector. Finally non-maximal suppression is applied to remove corners which have an adjacent corner with higher value of the sum of the absolute difference between the pixels in the circle and the center pixel.

For N values of 9, 10, 11, and 12, the corresponding FAST algorithms are named FAST 9, FAST 10, FAST 11, and FAST 12. According to the experiments in Ref. [6], FAST 9 seems to be the best FAST detector, and it is over five times faster than the quickest non-FAST detector. The FAST algorithm also significantly outperforms Harris, DoG, Harris-Laplace, SUSAN, etc. in repeatability, except in cases with large amounts of added image noise.

2.2 LBP Method

LBP is firstly proposed as texture operator [16][17]. It describes each pixel by the relative gray values of its neighboring pixels. If the gray value of the neighboring pixel is higher or equal to that of the center pixel, the binary value is set to be one, otherwise to be zero. The LBP value of a center pixel in (x, y) position can be calculated over the neighborhood as follows:

$$LBP_{R,N}(x, y) = \sum_{i=0}^{N-1} s(n_i - n_c)2^i, \quad s(t) = \begin{cases} 1, & t \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where n_c is the gray value of the center pixel, and n_i the gray value of N equally spaced pixels on a circle of radius R . According to Eq.(1), the $LBP_{R,N}$ value may be any integer between 0 and $2^N - 1$. The histogram of the $LBP_{R,N}$ values computed over an image region (the histogram dimension will be 2^N) can be used for texture description, and it has been proven to be robust against illumination changes. It is also very fast to compute.

Several modified versions of LBP method have been described in Ref. [17] for achieving rotation invariance and reducing the histogram dimension of LBP. When the image is rotated, the gray value n_i will correspondingly move along the perimeter of the circle around n_c , so different $LBP_{R,N}$ may be calculated. To remove the effect of rotation, the first modified version with rotation invariance is defined as follows:

$$LBP_{R,N}^{ri}(x, y) = \min\{ROR(LBP_{R,N}, i) \mid i = 0, 1, \dots, N-1\} \quad (2)$$

where $ROR(LBP_{R,N}, i)$ performs a circular bit-wise right shift on the N -bit number $LBP_{R,N}$ i times. $LBP_{R,N}^{ri}$ can have 36 different values when $N = 8$, and the histogram dimension of $LBP_{R,N}^{ri}$ over an image region is 36.

In the second version named as *uniform* LBP, at most two one-to-zero or zero-to-one transitions in the circular binary code are allowed, so whether a LBP is uniform can be judged by the following definition:

$$U(LBP_{R,N}) = |s(n_{N-1} - n_c) - s(n_0 - n_c)| + \sum_{i=1}^{N-1} |s(n_i - n_c) - s(n_{i-1} - n_c)| \quad (3)$$

If $U(LBP_{R,N}) \leq 2$, the LBP is uniform. The *uniform* LBP, expressed as $LBP_{R,N}^{u2}$, can have $N(N - 1) + 2$ different values, so the histogram dimension of $LBP_{R,N}^{u2}$ over an image region is $N(N - 1) + 2 + 1$ (the final 1 corresponds to those *non-uniform* LBP).

The third version is the *uniform* LBP with rotation invariance which combines the above two modifications. So $LBP_{R,N}^{riu2}$ value is computed as follows:

$$LBP_{R,N}^{riu2}(x, y) = \begin{cases} \sum_{i=0}^{N-1} s(n_i - n_c), & U(LBP_{R,N}) \leq 2 \\ N + 1, & \text{otherwise} \end{cases} \quad (4)$$

$LBP_{R,N}^{riu2}$ value can have $N + 1 + 1$ different values, so the histogram dimension of $LBP_{R,N}^{riu2}$ over an image region is $N + 1 + 1$.

All the three modified LBP versions can be considered to be a mapping from the original LBP with wide value range to the corresponding modified LBP with narrow value range, so the histogram dimension can be reduced with different extents. In practice, the mapping process is implemented by a look-up table which can be created in advance according to the different mapping mode.

3 Our Novel Real-Time Local Visual Feature

Our novel real-time local visual feature, namely FAST+LBP, is divided into three steps: feature detector, feature region determination, and feature descriptor.

3.1 FAST Feature Detector

Because the FAST 9 algorithm has a low computation cost and excellent performance in repeatability, it was chosen as the feature detector. The typical panoramic images and the corner features detected by FAST 9 are demonstrated in Fig.1. The images are from the COLD database [15]. The two images are captured by the robot's omnidirectional vision in two different positions. The robot's translation is 0.7561 m, and the rotation is 0.9053 rad.

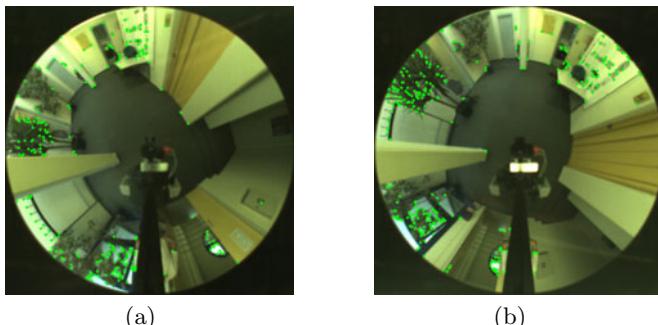


Fig. 1. A pair of panoramic images from the COLD database and the feature detecting results by FAST 9. The green points are the detected corner features.

3.2 Feature Region Determination

After a corner feature has been detected, a surrounding image region should be determined, and then a descriptor can be extracted from the image region. Some affine invariant feature detectors [5] have been proposed to adapt the feature region to affine transformations by iterative algorithms. Although they can provide better performance, the computation complexity increases significantly [5]. Therefore we do not consider affine invariance for our real-time local visual feature algorithm. We adopt the feature region determining method proposed in Ref. [18]. Rectangular image regions surrounding corner features are firstly determined in the radial direction, and then rotated to a fixed orientation, as shown in Fig.2. Fig.2(a) shows how a determined feature region is rotated to the fixed orientation. During the rotation process, bilinear interpolation is used.

In the next section we will compare this feature region determining method with the one which determines feature regions directly in horizontal and vertical directions through experimentation. The size of each feature region is also an important parameter, and the best size will be determined in the next section.

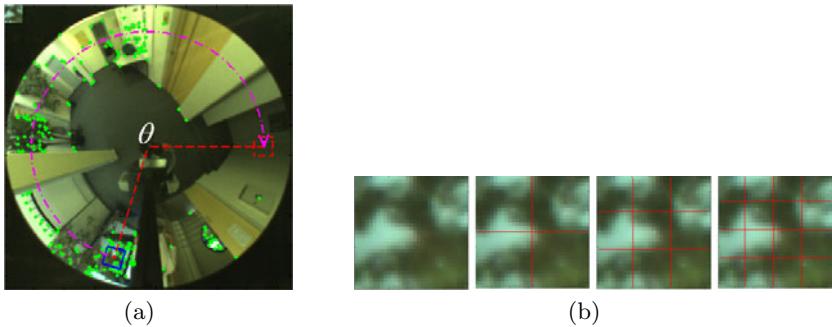


Fig. 2. (a) The blue rectangle is a feature region, and it is rotated by angle θ to a fixed orientation. The small region on the top left of the image is the rotated feature region. (b) The different grids that the feature region can be divided into. From left to right: 1×1 cell, 2×2 cells, 3×3 cells, 4×4 cells.

3.3 Feature Descriptor with LBP

The final step of local visual feature algorithm is to describe the features by calculating vectors according to the information of feature regions. Recently, LBP has been used as feature descriptor in Ref. [9], and the strength of SIFT descriptor is also combined. The SIFT-like grid is used, but SIFT gradient features are replaced by LBP-based features. In this paper, we use the same approach to extract descriptors for the detected features by FAST.

A LBP value for each pixel of the feature region can be calculated according to the introduction in section 2.2. In order to incorporate spatial information into the descriptor, the feature region can be divided into different grids such as 1×1 cell, 2×2 cells, 3×3 cells, 4×4 cells, as shown in Fig.2(b). For each cell,

the histogram of LBP values is created, and then all the histograms are concatenated into a vector as the descriptor. Finally, the descriptor is normalized to unit length. The descriptor dimension is $M \times M \times \text{the histogram dimension}$ for $M \times M$ cells. Therefore, the resulting descriptor is a 3D histogram of LBP feature locations and LBP values. In calculating the histogram, the LBP values can be weighted with a Gaussian window overlaid over the whole feature region, or with uniform weights over the whole region. The latter means that the feature weighting is omitted.

The performance and dimension of LBP descriptor will be affected greatly by different algorithm parameters such as the number of cells, different R and N , Gaussian or uniform weighting, LBP mode including the original LBP, LBP^{ri^2} , LBP^{u^2} , and LBP^{riu^2} as introduced in section 2.2. The best parameters will be determined by experiments in the next section.

4 Experimental Evaluation and Discussion

In this section, we will introduce the experimental setup firstly, and then determine the best parameters for FAST+LBP by experiments. The performance and the needed computation time of our algorithm will be compared with SIFT.

4.1 Experimental Setup

COLD is a freely available database which provides a large-scale, flexible testing environment for robot's vision-based topological localization [15]. The panoramic images are captured by the same omnidirectional vision in different rooms and under various light conditions. We will use the typical panoramic images and image series to perform our experiments.

When local visual features are applied in robot's localization, robot's SLAM, etc., the features should be matched between the image pairs captured in different imaging conditions, such as different robot positions and various lighting conditions. Therefore we evaluate the performance of local visual feature according to the feature matching results. For each feature descriptor in an image, We calculate its Euclidean distances with all the feature descriptors in another image needing to be matched. We consider that a match is found between the feature pair with the closest distance if the ratio of the closest to second-closest distance is smaller than threshold T_{ratio} [3] as follows:

$$\text{ratio} = \frac{\text{the closest distance}}{\text{the second - closest distance}} \leq T_{ratio} \quad (5)$$

In this paper, we will evaluate the overall performance of local visual feature as a whole, but not evaluate the detector and descriptor independently as in Ref. [5][6][8][9]. Therefore we use *matching score versus 1-precision* as the criterion for performance evaluation, instead of *recall versus 1-precision* which is used

to evaluate the descriptor's performance in Ref. [8][9]. We define *matching score* in the same way as Ref. [10]:

$$\text{matching score} = \frac{\text{the number of correct matches}}{\text{the smaller number of features in the pair of images}} \quad (6)$$

We define $1 - \text{precision}$ as follows in the same way as Ref. [8][9]:

$$1 - \text{precision} = \frac{\text{the number of false matches}}{\text{the number of correct matches} + \text{the number of false matches}} \quad (7)$$

After the feature matching is finished, an 18 bin histogram is created from $\Delta\theta_i = \text{normalize}(\theta_i - \theta'_i)$ using all the matched features, where θ_i and θ'_i are the rotated angles of the i th pair of matched features relative to the fixed orientation in section 3.2, and $\text{normalize}(.)$ means normalizing the angle to $[0, 2\pi]$. According to the character of omnidirectional vision, the relative angle of each pair of correctly matched features, namely φ , should be almost the same, so it can be estimated by computing the mean value of those $\Delta\theta_i$ falling into the highest bin, and φ is the rotation angle of robot approximately. If $|\Delta\theta_i - \varphi| < T_{angle}$, where T_{angle} is the threshold determined by experiments, the match related to $\Delta\theta_i$ is a correct match. Otherwise, it is a false match.

As we change the threshold T_{ratio} , the curve of *matching score versus* $1 - \text{precision}$ can be acquired to evaluate the performance of algorithms.

4.2 Parameter Evaluation for FAST+LBP

The evaluation of different parameter settings for FAST+LBP is carried out in this experiment to determine the best parameters. As presented in section 3, six parameters will affect the performance of FAST+LBP. We will test their different settings as follows:

The size of the feature region: 15×15 , 19×19 , 23×23 , 27×27 , 31×31 , 35×35 , 39×39 , 43×43 pixels; the feature region determining method: method 1—determining the feature's rectangular region directly in horizontal and vertical directions, method 2—determining the rectangular region in the radial direction and then rotating it to a fixed orientation as proposed in section 3.2; the number of grids: 1×1 cell, 2×2 cells, 3×3 cells, 4×4 cells; the N and R : $N = 8$ and $R = 1$, $N = 16$ and $R = 2$, $N = 24$ and $R = 3$; the LBP mode: the original LBP, LBP^{ri} , LBP^{u2} , LBP^{riu2} ; the weighting strategy: Gaussian weighting, uniform weighting.

Because of a huge amount of different combinations of above parameters, only one parameter is varied at a time while the others are kept fixed. The pair of images in Fig.1 are used to perform the feature matching, and the curves of *matching score versus* $1 - \text{precision}$ with different parameters are shown in Fig.3. The red curves in Fig.3 represent the best parameters. From the matching results, we see that 27×27 pixels for the feature region, region determining method 2, 2×2 cells, $N = 8$, $R = 1$, LBP^{u2} , and Gaussian weighting provide the best performance for FAST+LBP. The descriptor dimension of our final FAST+LBP is $2 \times 2 \times (8 \times 7 + 2 + 1) = 236$, as shown in Fig.4.

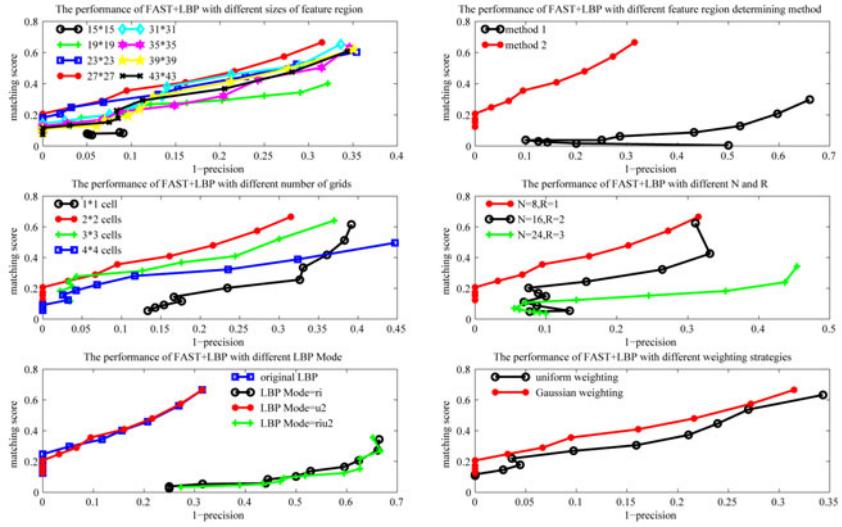


Fig. 3. Parameter evaluation results for FAST+LBP. Only one parameter is varied at a time while the others are kept fixed with the best parameters.

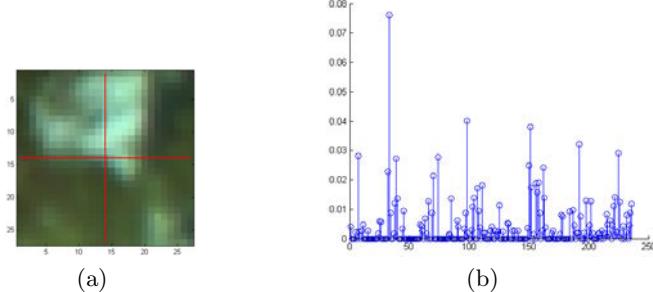


Fig. 4. Our final FAST+LBP. (a) The scale-up feature region which is divided into 2×2 cells. (b) The resulting feature descriptor.

4.3 Performance Comparison of FAST+LBP and SIFT

The performance comparison of FAST+LBP and SIFT is carried out in this experiment. The SIFT we adopt is implemented by Andrea Vedaldi [19]. Because most of the current robot's cameras are color ones, and the images in the COLD database are color images, we also compare the color version of FAST+LBP together. In our color version of FAST+LBP, the feature detector still uses the gray values of images, but the descriptor is computed in all of the R, G, B color channel, so its dimension is three times of that of the gray version. Two pairs of images are used. The first one is that in Fig.1, and they are acquired when the

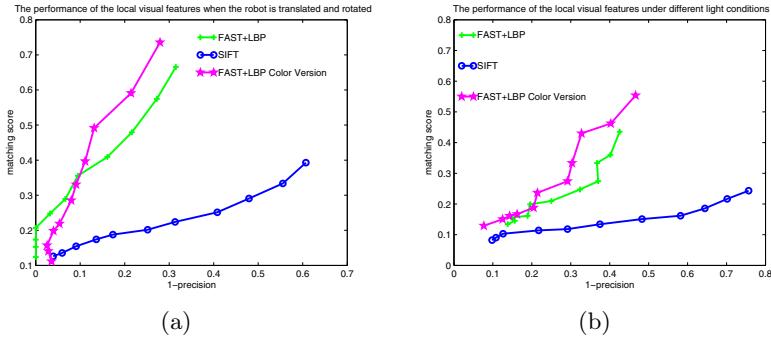


Fig. 5. The performance comparison of FAST+LBP, color version of FAST+LBP, and SIFT. (a) Robot is translated and rotated. (b) Under different lighting conditions.

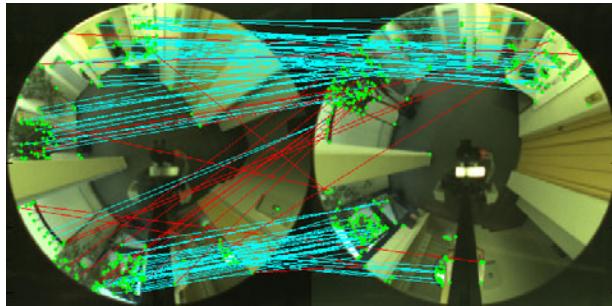


Fig. 6. The matching results of the pair of images in Fig.1 by FAST+LBP. The cyan lines represent the correct matches, and the red lines represent the false matches.

robot is translated and rotated. The second pair of images are captured when the robot is in the same position but under different lighting conditions. The matching results of these two pairs of images are depicted in Fig.5 (a) and (b) respectively.

We fix the threshold T_{ratio} as 0.95 after making a compromise between *matching score* and *precision*. The matching result of the pair of images in Fig.1 by FAST+LBP with this threshold is shown in Fig.6. Then we can evaluate how *matching score* changes with the different imaging conditions of omnidirectional vision caused by the robot's translation, rotation, and different lighting conditions. Three image series are used in this evaluation. The first one includes 30 images which are acquired as the robot is only translated. The translation increases with the image number, and the maximal translation is 1.7975 m. The second one includes 17 images which are acquired as the robot is only rotated. The rotation increases with the image number, and the maximal rotation is π . The third one includes 5 images which are acquired in the same position and under different lighting conditions. We perform the feature matching between

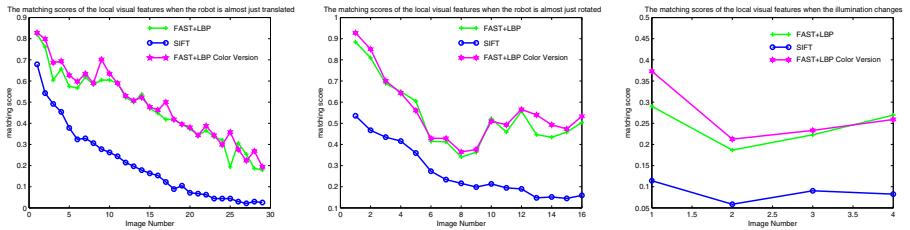


Fig. 7. The *matching score* with different imaging conditions by FAST+LBP, color version of FAST+LBP, and SIFT. (left) The robot is only translated. (middle) The robot is only rotated. (right) Under different lighting conditions.

the first image and all the other images in each series, so how *matching score* changes with different imaging conditions is acquired, as shown in Fig.7.

From the above experimental results, we clearly see that FAST+LBP provides better performance than SIFT in image matching, and it is a excellent local visual feature for omnidirectional vision. The matching results are not bad even when the robot is translated and rotated greatly and the lighting conditions are very different. So FAST+LBP is robust to rotation, different lighting conditions, and robot's certain translation. The color version seems a little better than the gray version. However, its computation cost is much higher, because the descriptor of the color version should be computed in each of the three color channels. Furthermore, it takes much more time to match features for the color version because of the larger descriptor dimension. So we prefer the gray version rather than the color version.

4.4 Comparison of the Needed Computation Time

In this experiment, we collect 125 panoramic images from the COLD database, and then extract local visual features from these images using FAST+LBP and SIFT respectively. Our FAST+LBP is implemented by C++, and the SIFT we use is implemented by C++ and Matlab using C-Mex technique [19]. The computer is equipped with 2.26GHz Duo CPU and 1.0G memory. The number of features, the time needed to extract all the features in an image, and the average time needed to extract one feature are demonstrated in Fig.8. The time needed in the three steps of FAST+LBP is also shown. We see that FAST+LBP extracts about 150~350 features per image, less than SIFT, but it can be performed much faster. After doing statistics on the computation time, we find that the time needed to extract all the features by SIFT in an image is 508 times that of FAST+LBP, and the average time needed to extract one feature by SIFT is 115 times that of FAST+LBP. The computation time needed to extract all the features in an image by FAST+LBP is from 5ms to 20ms, so it can be performed in real-time.

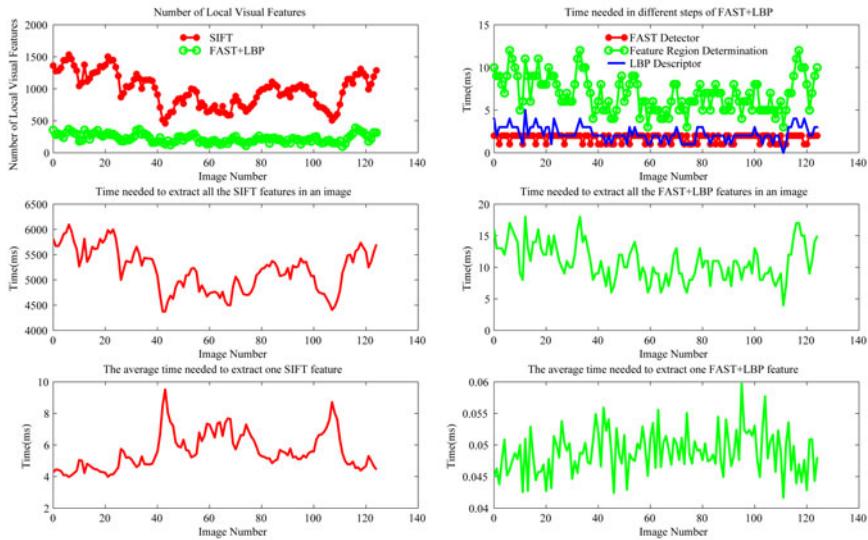


Fig. 8. The comparison of the needed computation time by FAST+LBP and SIFT

5 Conclusions

A novel local visual feature, namely FAST+LBP, is proposed for omnidirectional vision. It combines the advantages of two computationally simple operators by using FAST as the feature detector and LBP operator as the feature descriptor. The best algorithm parameters were determined by experiments. The comparisons between FAST+LBP, color version of FAST+LBP, and SIFT were performed, and the experimental results show that our algorithm has better performance than SIFT, and features can be extracted in real-time. Our local visual feature can be applied to computer/robot vision tasks with high real-time requirements.

Acknowledgement

We thank Edward Rosten and Tom Drummond for their release of the FAST source code, Marko Heikkilä and Timo Ahonen for their release of the LBP source code, Andrea Vedaldi for his release of the SIFT source code, and Andrzej Pronobis, Barbara Caputo, et al. for providing their COLD database.

References

1. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of Alvey Vision Conference, pp. 147–151 (1988)
2. Smith, S.M., Brady, J.M.: SUSAN-a new approach to low level image processing. Int. J. Comput. Vision 23(1), 45–78 (1997)

3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* 60(2), 91–110 (2004)
4. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. In: *Proceedings of BMVC*, pp. 384–393 (2002)
5. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. *Int. J. Comput. Vision* 60(1), 63–86 (2004)
6. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006. LNCS*, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
7. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding* 110, 346–359 (2008)
8. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* 27(10), 1615–1630 (2005)
9. Heikkilä, M., Pietikäinen, M., Schmid, C.: Description of interest regions with local binary patterns. *Pattern Recognition* 42, 425–436 (2009)
10. Mikolajczyk, K., Tuytelaars, T., Schmid, C., et al.: A comparison of affine region detectors. *Int. J. Comput. Vision* 65(1), 43–72 (2005)
11. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. *Int. J. Comput. Vision* 37(2), 151–172 (2000)
12. Li, J., Allinson, N.M.: A comprehensive review of current local features for computer vision. *Neurocomputing* 71, 1771–1787 (2008)
13. Grabner, M., Grabner, H., Bischof, H.: Fast approximated SIFT. In: Narayanan, P.J., Nayar, S.K., Shum, H.-Y. (eds.) *ACCV 2006. LNCS*, vol. 3851, pp. 918–927. Springer, Heidelberg (2006)
14. Tamimi, H., Andreasson, H., Treptow, A., et al.: Localization of Mobile Robots with Omnidirectional Vision using Particle Filter and Iterative SIFT. *Robotics and Autonomous Systems* 54, 758–765 (2006)
15. Pronobis, A., Caputo, B.: COLD: The Cosy Localization Database. *The International Journal of Robotics Research* 28(5), 588–594 (2009)
16. LBP website, http://www.ee.oulu.fi/mvg/page/lbp_bibliography
17. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.* 24(7), 971–987 (2002)
18. Andreasson, H., Treptow, A., Duckett, T.: Self-Localization in non-stationary environments using omni-directional vision. *Robotics and Autonomous Systems* 55, 541–551 (2007)
19. Vedaldi, A.: SIFT source code,
<http://www.vlfeat.org/~vedaldi/code/sift.html>

Mixed 2D/3D Perception for Autonomous Robots in Unstructured Environments

Johannes Pellenz¹, Frank Neuhaus²,
Denis Dillenberger², David Gossow², and Dietrich Paulus²

¹ Bundeswehr Technical Center for Engineer and General Field Equipment,
Universitätsstr. 5, 56070 Koblenz, Germany
Johannes.Pellenz@bwb.org

² Active Vision Group, University of Koblenz-Landau,
Universitätsstr. 1, 56070 Koblenz, Germany
{fneuhaus,dillenberger,dgossow,paulus}@uni-koblenz.de,
<http://robots.uni-koblenz.de>

Abstract. Autonomous robots in real world applications have to deal with a complex 3D environment, but are often equipped with standard 2D laser range finders (LRF) only. By using the 2D LRF for both, the 2D localization and mapping (which can be done efficiently and precisely) *and* for the 3D obstacle detection (which makes the robot move safely), a completely autonomous robot can be built with affordable 2D LRFs. We use the 2D LRF to perform particle filter based SLAM to generate a 2D occupancy grid, and the same LRF (moved by two servo motors) to acquire 3D scans to detect obstacles not visible in the 2D scans. The 3D data is analyzed with a recursive principal component analysis (PCA) based method, and the detected obstacles are recorded in a separate obstacle map. This obstacle map and the occupancy map are merged for the path planning. Our solution was tested on our mobile system Robbie during the RoboCup Rescue competitions in 2008 and 2009, winning the mapping challenge at the world championship 2008 and the German Open in 2009.

This shows that the benefit of a sensor can dramatically be increased by actively controlling it, and that mixed 2D/3D perception can efficiently be achieved with a standard 2D sensor by controlling it actively.

1 Introduction

Autonomous mobile systems are still a current research topic; the testing ground for the systems is often a obstacle free laboratory or office like environment or structured, well defined areas such as the RoboCup Soccer fields. However, the ongoing development also enables using robots in rough terrain as well. This was demonstrated in the DARPA Grand Challenge in 2004 and 2005, as well as in the DARPA Urban Challenge in 2007 [Thrun, 2006]. Most of the successful robots in the DARPA competitions were equipped with either multiple 2D laser range finders (LRFs), or with a 3D LRF that scans in 64 planes at the same time (Velodyne HDL-64E¹). Using multiple LRFs or a Velodyne HDL-64E is not an option for most small robots due to the high cost,

¹ See: <http://www.velodyne.com/lidar>

the weight and the power consumption. An alternative to multiple 2D LRFs is to rotate a single 2D LRF (for an application in rescue robotics, see [Ohno et al., 2008] or [Nüchter, 2006]). Using such a rotating 2D LRF while driving results in a lack of accuracy, which finally yields to non precise maps. Sheh et al. [Sheh et al., 2007] avoid this problem by using a SwissRanger SR-3100 time of flight (TOF) camera. They use the Hough transform to generate a model of the environment, which is assumed to have the form of a NIST random step field (a well defined structure with a significant direction). Therefore, their approach does not work in unstructured environments. Kadous et al. [Kadous et al., 2006] used extracted features in the surrounding of the robot to select behaviors to control the robot over the rough terrain. The focus of their work is to find a suitable representation of the environment for machine learning.

Our goal is to combine – in one device – the complete perception in 3D using a 3D LRF and the speed and precision of a static 2D LRF for accurate 2D position estimation and precise 2D maps. Our approach works as follows: The localization and mapping is done in 2D while the robot is driving. The 2D laser scans come from short range 2D LRF which is mounted on two servo motors. The global map is stored in an single occupancy grid. The grid is composed of $50\text{ mm} \times 50\text{ mm}$ cells that store the information if the cell is occupied or not [Elfes, 1989]. The information is stored in two planes: One plane stores the number of times the cell was "touched" by a laser beam, and the other plane stores the information if the cell has been seen as occupied. The ratio of the values on both cells yields the probability value for an occupation. A frontier based algorithm (Exploration Transform, see [Wirth and Pellenz, 2007]) selects the next frontier that the robot is going to explore and also calculates the safest path to this target. Whenever such a frontier is reached, a 3D scan is acquired. To capture the 3D scan, the same LRF that is used for the 2D mapping is utilized by tilting the LRF with one of the servo motors. To detect the regions where obstacles occur, the 3D data is analyzed using a hierarchical principal component analysis (PCA): First, a larger area is analyzed. If the whole area is "flat" (by checking the properties of the eigenvalues), then no 3D obstacles were detected. If there are obstacles in the area, the area is split in two parts, and the PCA is applied for both sub-regions. This is done until the remaining areas are flat or are smaller than $10\text{ cm} \times 10\text{ cm}$. After the analysis, the location of 3D obstacles is known in the local robot coordinate system. Using the localization information from the 2D particle filter based localization step, these spots are stored in a separate map, the obstacle map. Keeping the information separate helps the localization algorithm to match the following 2D laser scans with the occupancy map. However, for the navigation algorithm it is important to know about both, the obstacles detected by the 2D and the 3D LRF. Therefore, both maps are merged into one navigation map, which is then used by the Exploration Transform to plan the path.

The mixed 2D/3D perception was implemented on our mobile robot Robbie and was tested during the RoboCup in 2008 and 2009. Using this technique, our wheel-based robot was able to detect obstacles in different heights such as step fields and ramps, but also objects hanging from the arena elements (stalactites).

The rest of the paper is organized as follows: Sec. 2 describes the self made 3D sensor that is used for most of the experiments. Sec. 3 explains the algorithms that are used to analyze the 3D data and to store the gained information in two maps. Sec. 4

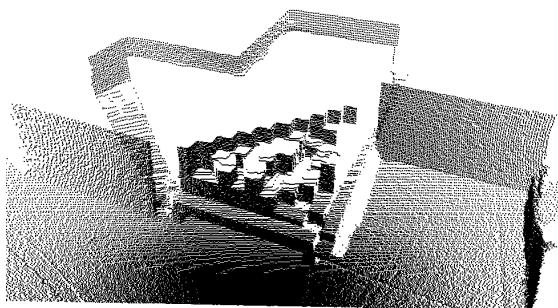
describes the experiments that were performed using two different types of data sets, an outdoor dataset (captured using a commercial 3D LRF) and an indoor RoboCup Rescue data set (captured with the self made 3D LRF). Sec. 5 describes the results, and Sec. 6 finally lists the open issues and the future work.

2 Sensor Description

The 2D laser scans for the indoor datasets come from a Hokuyo URG-04LX LRF; this sensor can measure distances up to 5.6 meters. The LRF is mounted on two servo motors, which – in 2D mode – keep the sensor always adjusted [Pellenz, 2007]. This way, the laser beams still measure the surrounding walls, and not the ground or obstacles behind the next barrier. The setup is depicted in Fig. 1a. The servo motors are controlled by a microcontroller and a laptop; the declination data comes from a cheap two-axis gravity sensor (ADXL202). In the 3D mode, the roll servo is used to keep the servo straight, and the tilt servo is used to change the scanning plane. 55 scans are taken in about 1° steps in front of the robot. An example of a 3D laser scan is depicted in Fig. 1b.



(a) Laser scanner mounted on two servo motors.



(b) Resulting 3D laser scan.

Fig. 1. Laser range finder Hokuyo URG-04LX gimbal mounted on two servo motors (the roll servo motor is occluded) and the resulting 3D point cloud of a random step field

3 Algorithms

The analysis of the 3D point cloud is based on the principal component analysis (PCA). In the following subsections, first the PCA and the grid-based PCA is introduced, followed by our extension, the Hierarchical PCA. Finally, an approach to analyze the roughness of the ground using the distribution of 3D laser distances is presented.

3.1 PCA

The PCA of the covariance matrix of a three-dimensional point cloud (which is in this specific case in fact equivalent to the eigendecomposition) yields three eigenvectors

with corresponding eigenvalues. The latter indicate the variance of the point cloud along the corresponding axes, and can be used to determine the shape of the point cloud. Fig. 2 shows the three different cases that can occur: If all eigenvalues are roughly equal, the point cloud is more or less unstructured without any specific principal direction. If one of them is significantly higher than the remaining two, the shape of the point cloud is cylindrical and the eigenvector corresponding to the highest eigenvalue defines the direction of the cylinder. If one of the eigenvalues is very low in comparison to the other two, the point cloud is more or less a plane, and the eigenvector corresponding to the lowest eigenvalue defines the normal of the plane.

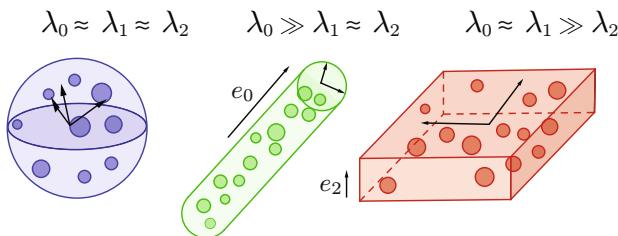


Fig. 2. Three cases of point cloud analysis

The above mentioned methodology only yields good results for reasonably small point clouds. Those containing multiple planes/cylinders or other complex structures simply end up in the case where all three eigenvalues are large. In order to gain local information about a large point cloud (such as the one delivered by the 3D laser scanner), the data has to be subdivided into smaller chunks, which can then be analyzed with the help of the PCA. There is a multitude of possibilities to subdivide the point cloud. We use a simple 2D grid, which is centered around the origin of the sensor. Other researchers such as [Lalonde et al., 2006] used 3D grids. The problem with these uniformly sized cloud-fragments is that the ever increasing spread of sensor's field of view makes information about far-away ground surfaces sparse. Selecting a small cell-size for the grid results in many distant cells being completely empty, simply because not a single laser beam endpoint happens to be inside this cell. Selecting a large cell size allows larger regions to be considered in the PCA step. However, this causes problems in regions close to the sensor: If a small obstacle is inside a cell, the whole cell is classified as occupied. Subsequent path planning algorithms operating exclusively on this grid would have to keep significantly more distance to obstacles than really needed. In some cases, such as narrow passages, where precise navigation is required, this methodology can even impede path-planning altogether, simply because almost everything is marked as impassable. What is really needed is an algorithm that combines the advantages of both, small *and* large cells.

3.2 Hierarchical PCA

The Hierarchical PCA is an algorithm we developed to recursively subdivide the point cloud. We use it on a 2D grid, however the extension to 3D is straightforward. The cell size is chosen to be relatively small.

The initial input of the algorithm is the whole grid. In each step, it performs a PCA analysis on the points in the considered region. If the point cloud contained in the region is not flat enough, the algorithm splits the region in two pieces and recursively calls itself on the two resulting fragments. The splitting simply occurs at the middle of the longest edge of the region. If the input region can no longer be subdivided, and it is still not flat enough, it is considered an obstacle. To quantify flatness, we first define the local height disturbance h as:

$$h = \lambda_k \quad \text{with} \quad k = \operatorname{argmax}_{i \in \{0,1,2\}} |\mathbf{e}_i^T \mathbf{z}| \quad (1)$$

where \mathbf{z} is a vector pointing up, \mathbf{e}_i are the eigenvectors, and λ_i the eigenvalues of the point cloud in the current region. A region is considered flat, if h is below some threshold. This definition makes sure, that even sloped but flat areas get low values of h .

Algorithm 1. Recursive PCA Lterrain analysis

```

function RECURSIVEPCA(area  $a$ )
     $h \leftarrow$  local height disturbance ( $a$ )            $\triangleright$  Using the eigenvalues calculated by the PCA
    if ( $h \leq t$ ) then                          $\triangleright t$  is a threshold
        mark all cells in  $a$  as drivable
    else
        if (size( $a$ )  $\leq$  (10 cm  $\times$  10 cm)) then
            mark cell in  $a$  as obstacles
        else
            ( $a_0, a_1 \leftarrow$  split ( $a$ )            $\triangleright$  Split the area in two parts
            recursivePCA( $a_0$ )
            recursivePCA( $a_1$ )
        end if
    end if
end function

```

3.3 Roughness Analysis

Having distinguished between passable and non-passable areas, one issue remains: Even though a region may technically be drivable, it may still be desirable to prefer one region over another. An example of this is when the robot is driving on a track and next to this track a still passable rubble pile is located. Obviously, one expects the robot to continue its way on the track instead of driving through the debris. One may be tempted to use simply the local height disturbance as a measure for the roughness of the terrain at that spot. Unfortunately, it turns out that due to the large sensor noise relative to the height of the bumps that are to be detected, it is infeasible to distinguish rough from flat areas properly.

As [Montemerlo et al., 2008] have already noticed, a much better indicator of roughness is the local *distance* disturbance, since small bumps and dents in the ground cause large changes in distance of adjacent laser measurements because of the grazing angles with which the laser rays hit the ground. This can be seen in Fig. 3: The box represents

the laser range finder, mounted at a height h_1 . The robot is standing in front of a bumpy area. The bumps are assumed to have a fixed height h_2 in the considered region. Now each ray will either hit a bump or barely graze above one, hitting the ground behind. The distance difference e between a measurement that has hit the ground, and one that has hit the bump is relatively large compared to the height of the bump. In addition to that, it linearly increases as the distance to the bump grows. This can be seen in the following equation for e , which can be derived using the intersecting lines theorem:

$$e = d \cdot \frac{h_2}{h_1 - h_2} \quad (2)$$

Note how d scales up the remaining term, which in fact describes the intensity of the bump.

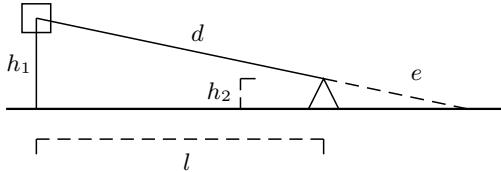


Fig. 3. Geometry of distance disturbances

The 3D laser scanner based on the Hokuyo URG-04LX provides 55 2D laser scans per 3D scan: one for each orientation the tilt servo motor was set to. We consider each 2D laser scan separately. If one laser is not able to measure a distance at a certain spot, a very low distance value will be reported (values below 20 indicate an error). In a first fast preprocessing step, we eliminate these erroneous measurements. This is done by looping over each scan, overwriting every faulty measurement with the last valid value. In the next step, a heavily smoothed copy of this data is made. We use a simple mean filter (of size nine)—mostly because of speed and because superior low-pass filters such as Gaussian or even edge-preserving smoothing filters did not show any significant improvement. Finally, the unfiltered version is subtracted from the low-pass filtered version, forming a high-pass filtered version of the data. Now, the distance deviation δ_i from the smoothed version is known for each laser measurement. For each grid-cell containing points, we compute the variance of distance deviations $\hat{\sigma}_\delta^2$ of all n points in that cell, using

$$\hat{\sigma}_\delta^2 = \frac{1}{n} \sum_{i=1}^n \delta_i^2 - \left(\frac{1}{n} \sum_{i=1}^n \delta_i \right)^2 \quad (3)$$

However, $\hat{\sigma}_\delta^2$ is really a sum of the inherent sensor noise σ_L^2 and the actual distance variance σ_δ^2 . Since we are only interested in the latter, and since we assume the sensor noise to be additive, we eliminate the sensor noise by computing σ_δ^2 as $\max\{0, \hat{\sigma}_\delta^2 - \sigma_L^2\}$.

This measure is not yet independent to the distance, as we have motivated in equation (2): It will be too low in areas near the robot, or too high for areas which are far away. Therefore, we obtain the local terrain roughness r as we eliminate the previously computed measure from the influence of the distance:

$$r = \frac{\sigma_\delta^2}{d_{\text{Cell}}^2} \quad (4)$$

where d_{Cell} is the distance of the laser to the respective grid-cell. The roughness r can now be used to determine the drivability of the terrain: High values of r correspond to high terrain roughness, low values to low roughness. Now a robot could, for example, try to find a path, where the sum of all r 's in the crossed cells is minimal.

3.4 Obstacle Map \mathbf{m}^{obst}

After the analysis of the 3D scan using the Hierarchical PCA, the location of 3D obstacles is known in the local robot coordinate system. For performance reasons, we do not want to perform a complete 3D mapping which would involve a time consuming and sometimes unstable 3D point cloud registration as described in [Nüchter, 2006]. In our case it is sufficient to take the 2D pose information (x, y, θ) from the 2D SLAM and to calculate the positions of the obstacles using this pose information.

It is important that the 3D obstacles are stored in a separate map and that the occupancy map (with the obstacles seen by the LRF in 2D mode) is not modified, because the occupancy map is used as a reference for the 2D SLAM algorithm. If the 3D obstacles would be marked here, mismatches between these obstacles and other objects in the 2D scans could occur.

Using the localization information from the 2D particle filter based localization step $(x_{robot}^W, y_{robot}^W, \theta)^T$, the world coordinates of the midpoint $\mathbf{p}_{obst}^W = (x_{obst}^W, y_{obst}^W)^T$ of a occupied cell is calculated:

$$\mathbf{p}_{obst}^W = \begin{pmatrix} \cos(\theta)x_{obst}^R - \sin(\theta)y_{obst}^R + x_{robot}^W \\ \sin(\theta)x_{obst}^R + \cos(\theta)y_{obst}^R + y_{robot}^W \end{pmatrix} \quad (5)$$

Finally, the obstacle is stored in a separate map \mathbf{m}^{obst} , if it is not already marked in the occupancy grid \mathbf{m}^{occ} :

$$\mathbf{m}_{\text{worldToMap}(\mathbf{p}_{obst}^W)}^{obst} = \text{occupied}, \text{ if } \text{class}(c_{i,j}) = \text{occupied} \wedge \mathbf{m}_{\text{worldToMap}(\mathbf{p}_{obst}^W)}^{occ} = \text{free} \quad (6)$$

By checking for the obstacle in \mathbf{m}^{occ} the algorithm makes sure that obstacles are stored only once if the obstacle was detected by the 2D LRF already.

4 Experiments

We tested the terrain classification algorithms on two different types of data sets: One dataset was acquired using the commercial 3D LRF Velodyne HDL-64E. These datasets represent larger scale outdoor scenes of the campus of the University Koblenz-Landau

and the surrounding. The other dataset was captured using the self made 3D LRF that was presented in Sec. 2. These datasets were captured on different RoboCup events. The result of the terrain classification is visualized: Every grid-cell is rendered as a small quadrilateral. They are oriented and located to approximate optimally the points contained in the respective grid-cells. Impassable cells are visualized by red boxes. The color of the quadrilaterals is the result of the roughness analysis algorithm which was presented in section Sec. 3.3. The colors range from green (low roughness) to orange (high roughness).



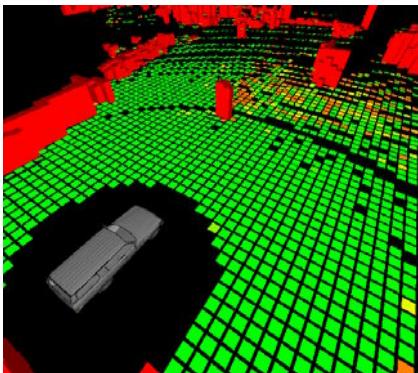
(a) Velodyne HDL-64E mounted on a (human driven) street car. (b) 3D LRF mounted on our autonomous rescue robot Robbie.

Fig. 4. Vehicles for capturing the test data

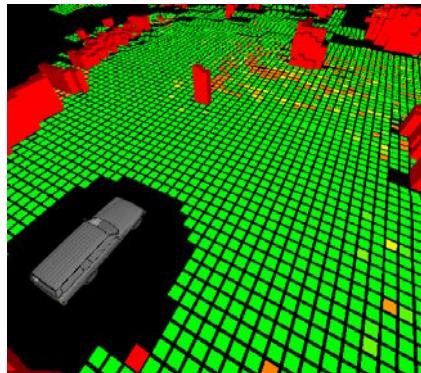
The sensors mounted on the test vehicles are shown in Fig. 4. Since only for the RoboCup dataset also a 2D map was generated, the mixed 2D/3D perception was tested with the RoboCup datasets only. For the visualization, the obstacle map m^{obst} was overlaid the occupancy grid m^{occ} so that the position of the obstacles can be verified visually. Finally, the robot equipped with the mixed 2D/3D perception was used during the RoboCups in 2008 and 2009.

5 Results

The result of the hierarchical PCA that was introduced in subsection 3.2 can be seen in Fig. 5a and 5b for a typical outdoor scene. Fig. 5a shows the result of the classical PCA with fixed size cells: Most cells are classified, but several cells—where not enough points were available—remain not classified. It is clearly visible that distant cells often do not contain any laser-endpoints. Fig. 5b shows how the Hierarchical PCA handles the same data: Multiple equally flat cells are merged for the analysis and small “holes” in the data coverage are no longer a problem for the path-planner. However, since the algorithm performs the maximum subdivision in areas close to obstacles, the issue remains near these objects. This can be seen on the right in Fig. 5b (there is a small hole next to the obstacle).

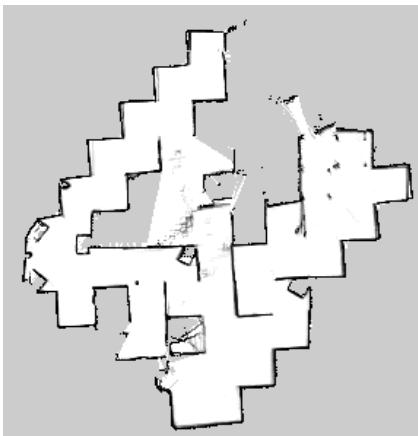


(a) PCA with fixed grid size: Some grid cells are unknown.

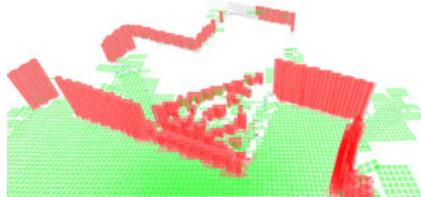


(b) Hierarchical PCA with variable grid size: Continuous surfaces.

Fig. 5. Comparison of the PCA with a fixed grid size (a) and the Hierarchical PCA (b). With the standard PCA gaps in the grid occur, whereas with the Hierarchical PCA larger regions are classified consistently as passable. (Note that the smallest possible tiles are shown in both images. The larger "logical tiles" used in the Hierarchical PCA are not visualized.)



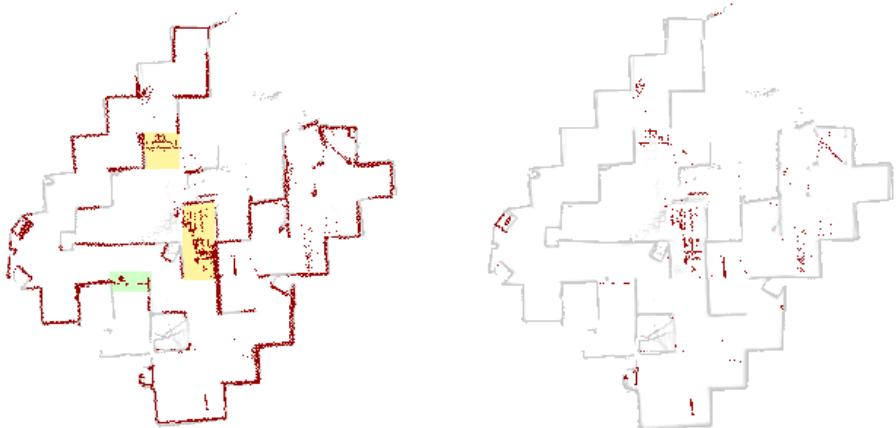
(a) 2D occupancy grid m^{occ} of a RoboCup Rescue arena (size is about 12 m × 12 m). For the robot not passable areas are marked as free, because the 2D LRF did not see the obstacles.)



(b) 2D grid with the result of the Hierarchical PCA: Red regions indicate obstacles, green regions indicate passable terrain. The position of the obstacles is stored in the map m^{obst} .

Fig. 6. Same arena, different views: Complete 2D map of the arena and a classified 3D LRF scan

The result for the RoboCup dataset is shown in Fig. 6 and Fig. 7. Fig. 6a shows the 2D occupancy map, as it is generated by our 2D particle filter based SLAM approach. Notice that only the walls are visible as obstacles (in black); neither the ramps nor the step fields are detected by the 2D laser range finder. The 3D laser scan and the



(a) Visual representation of the 2D map of the detected 3D obstacles m^{obst} . Obstacles are marked as dark dots.

(b) Like Fig. 7a, but obstacles that are already recorded in the occupancy grid m^{occ} are filtered out.

Fig. 7. Two versions of m^{obst} : With all detected obstacles and only with obstacles that were not seen by the 2D LRF. (For better orientation, the occupancy grid is also shown in light gray.)

classification result (occupied or not) is shown in Fig. 6b. The walls are clearly visible, but also the stepfield (in the middle of the scan) is detected. Using the pose estimation from the 2D SLAM, the position of all occupied cells (in red) is stored in m^{obst} . This map is shown in Fig. 7a. Since some obstacles (such as the walls) would be stored in both maps, obstacles that are already stored in the occupancy map are filtered out from m^{obst} , because cells here have a larger size and this redundant information would reduce the resolution of the merged map for the path planning. The map that is finally used by the path planner is depicted in Fig. 7b.

Our solution was tested on our mobile system Robbie during the RoboCup Rescue competitions in 2008 and 2009. The robot won the mapping challenge at the world championship in 2008 and was the only robot that was able to map the whole arena. In 2009, the robot won the RoboCup German Open, both the challenge for autonomous robots and the overall competition.

6 Future Work

So far, the roughness value are calculated and visualized, but not used in the following frontier selection and path planning step. The use of this value in the path planning is easy, because the Path Transform requires a cost function, which could easily be extended by the roughness value. Another subject of research is the utilization of neighborhood information during the classification of grid-cells.

References

- [Elfes, 1989] Elfes, A.: Occupancy grids: A probabilistic framework for robot perception and navigation. PhD thesis, Carnegie Mellon University (1989)
- [Kadous et al., 2006] Kadous, M.W., Sammut, C., Sheh, R.: Autonomous traversal of rough terrain using behavioural cloning. In: The 3rd International Conference on Autonomous Robots and Agents (2006)
- [Lalonde et al., 2006] Lalonde, J.-F., Vandapel, N., Huber, D., Hebert, M.: Natural terrain classification using three-dimensional ladar data for ground robot mobility 23(1), 839–861 (2006)
- [Montemerlo et al., 2008] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Hähnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paeffgen, J., Penny, I., Petrovskaya, A., Pfleuger, M., Stanek, G., Stavens, D., Vogt, A., Thrun, S.: Junior: The stanford entry in the urban challenge. *J. Field Robotics* 25(9), 569–597 (2008)
- [Nüchter, 2006] Nüchter, A.: Semantische dreidimensionale Karten für autonome mobile Roboter. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn (2006)
- [Ohno et al., 2008] Ohno, K., Kawahara, T., Tadokoro, S.: Development of 3d laser scanner for measuring uniform and dense 3d shapes of static objects in dynamic environment. In: ROBIO 2008: IEEE International Conference on Robotics and Biomimetics, February 22–25, pp. 2161–2167 (2009)
- [Pellenz, 2007] Pellenz, J.: Rescue robot sensor design: An active sensing approach. In: SRMED 2007: Fourth International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster, Atlanta, USA, pp. 33–37 (2007)
- [Sheh et al., 2007] Sheh, R., Kadous, M., Sammut, C., Hengst, B.: Extracting terrain features from range images for autonomous random stepfield traversal. In: IEEE International Workshop on Safety, Security and Rescue Robotics, SSRR 2007, pp. 1–6 (2007)
- [Thrun, 2006] Thrun, S.: Winning the darpa grand challenge: A robot race through the mojave desert. In: ASE 2006: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, p. 11. IEEE Computer Society, Washington (2006)
- [Wirth and Pellenz, 2007] Wirth, S., Pellenz, J.: Exploration transform: A stable exploring algorithm for robots in rescue environments. In: Workshop on Safety, Security, and Rescue Robotics, pp. 1–5 (2007), <http://sied.dis.uniroma1.it/ssrr07/>

Hierarchical Multi-robot Coordination

Viktor Seib, David Gossow, Sebastian Vetter, and Dietrich Paulus

Active Vision Group
University of Koblenz-Landau
Universitätsstr. 1
56070 Koblenz, Germany
vseib@uni-koblenz.de
<http://www.uni-koblenz.de/~agas>

Abstract. The complexity and variety of household chores creates conflicting demands on the technical design of domestic robots. One solution for this problem is the coordination of several specialized robots based on the master-slave principle. One robot acts as a master system, tracking and remotely controlling the slave robots. This way, only the master robot needs to be equipped with sophisticated sensors and computing hardware. We implemented a tracking system using an infra-red camera for the master and active markers on the slave robot. The master system is able to interact with the user using natural language. It builds a map of its environment automatically using a laser range finder. It can track a cleaning robot for which we use the commercially available platform “Roomba” by iRobot. The master safely navigates it to a given destination, avoiding obstacles. We successfully demonstrated the system during the RoboCup@Home competitions 2009 in Graz, Austria. We evaluate the performance of the two systems and describe the accuracy of localization and navigation.

1 Introduction

The RoboCup@Home competition concentrates on natural human-robot-interaction in typical domestic environments. In order to assist humans in accomplishing their daily tasks, robots have to be fully autonomous. That even applies in limited domains such as a household. Autonomy not only allows service robots to complete tasks on their own but also raises confidence in the machine and acceptance by human users.

The corner stones of autonomy, self localization and mapping, path planning, object recognition and manipulation with a robot arm have become an inherent part of the RoboCup@Home league. The hardware required for these purposes can be integrated on one single robot without facing major difficulties. However, even for the manipulation of objects, restricting decisions on the hardware design have to be made. Depending on the height the robot arm is mounted at, its reach is limited to certain heights. A vertically adjustable robot arm has been presented at the RoboCup@Home [SGK⁺09]. In reserving lots of space for the

vertical movements this approach asks for great compromises in the arrangement of the remaining hardware.

Our approach aims at transferring certain service functions from the main system to auxiliary systems - in this case a slave robot. This way, the complexity of the main robot can be reduced. The slave robot neither needs computing hardware nor sophisticated sensors that exceed its particular task.

One can expect many future homes to be equipped with simple cleaning robots by the time that sophisticated service robots become commercially available. We present an approach to integrate such a simple cleaning robot into a household robot system by applying several extensions. The slave robot is tracked using a commercially available, economically priced infra-red camera. The tracking is carried out by the main robot in real time, which itself is a mobile platform. As slave robot we use a Roomba vacuum cleaner to perform targeted cleaning jobs. Its small size enables it to clean corners and even to reach narrow parts of a room – places that are out of range of the main robot. The modifications allow the user to instruct the master robot with tasks expressed in natural language, e.g. “clean the kitchen”. This job can be performed completely autonomously even if the vacuum cleaner is somewhere else. The main robot locates it and navigates it to the desired location. After the task is completed the slave robot is navigated back to its initial position. A benefit arising from our approach is the ability of the main robot to monitor the areas already cleaned by the slave, thus enabling the slave robot to perform a complete coverage of the cleaning area.

The approach presented here can be easily transferred to other robots to cooperate in a household environment.

The next section describes previous works dealing with the coordination of multiple robots. In chapter 3 our approach is presented in detail, the results are described in section 4. Finally, chapter 5 deals with possible future extensions of our approach.

2 Related Work

In this chapter we describe previous work on fields related to the coordination of multiple robots and point out differences to our approach. Apart from multi-robot coordination, division of tasks among robots and master-slave systems, we also present related work on cleaning robots and robot tracking due to the character of the slave robot we use.

2.1 Multi-robot Coordination

The idea of using several robots to complete one task is widely studied. In [How06] an approach for multi-robot SLAM is presented. To complete this task, multiple robots can be placed at (unknown) random initial positions. Whenever two robots meet for the first time, they use their previously recorded measurements to fuse their mapping data into one map. This approach works in real time.

Another example of solving a complex task using multiple robots is presented in [BMF⁺00]. The individual robots spread to different target positions for fast exploration of an area. This probabilistic approach takes into account the utility of the target position as well as the costs of reaching it. The utility is determined by the unexplored area a robot can cover when reaching it. Whenever a robot chooses a position, its utility for other robots decreases.

The soccer playing robots in the RoboCup also have to coordinate themselves. [VV03] presents an approach of coordination and task assignment based on shared potential fields. Although this simple approach is not scalable to large numbers of robots, it can be successfully applied to small teams.

Another example for multi-robot coordination is [LAL⁺04]. It presents a distributed architecture for multi-robot task allocation based on an enhanced Contract-Net protocol.

In all of these examples, the robots have identical capabilities and are organized without using a hierarchy.

2.2 Division of Tasks

In the RoboCup@Home finals 2009, team NimbRo simultaneously used two robots, which were designed for different tasks. The strength of “Robotinho”, the first robot, was communication - both verbally and mimically [FBE⁺09]. Whereas the other robot, “Dynamaid”, handled object manipulation at variable heights. It picked up items from tables and from the floor. Robotinho walked the guest around the apartment, Dynamaid served a drink [SGK⁺09]. Due to the fact that these two actions took place in parallel, they were independent. An interaction between the two robots has not taken place. Instead they interacted simultaneous with a human. This is also a demonstration of transferring services to a second robot, although the robots here were – unlike in our approach – at an equal hierarchy level and both complex systems.

2.3 Master-Slave Systems

Robotic master-slave systems are used in minimally invasive surgery, where instruments are inserted into the human body to perform medical procedures [YTEM02], [TPM03]. The benefits are improved precision and less pain to the patient. Such an instrument (slave) is controlled through an interface (master) by a human operator. The works in the medical field deal with increasing precision and reliability of these systems.

2.4 Cleaning Robots

One main goal of previous works on cleaning robots is to achieve a complete coverage of the cleaning area [NdCVVR97], [SCPZ04]. These two works use different map representations and planning algorithms to complete this task. In [JN02], a dynamic approach for multiple cleaning robots is described. The whole area is divided into polygons, which can be allocated by the robots in order to be cleaned. Since allocations occur at runtime, it is possible to compensate

for the breakdown of a robot, because it would not allocate further polygons. However, the polygons already allocated by the broken robot remain uncleared. A complete coverage of the cleaning area can also be achieved with our approach, but there is no compensation for a broken robot.

[Kur06] presents a collection of possible extensions for Roomba. Some extensions can be incorporated into domestic projects, as we did with the wireless communication. In [TD07] Roomba's abilities are evaluated for research and education purposes. The authors give a detailed technical description of Roomba. They also present basic mapping and SLAM algorithms based on the assumption of only parallel or perpendicular walls. They use Roomba's odometry and bump sensors to obtain data that is then processed externally.

2.5 Tracking

The library ARToolkit¹ can be used for object tracking with the aid of a passive marker. However, this approach is not suitable for our purpose. Due to the needed size of the passive markers, utilization of ARToolkit is reasonable at short distances only. According to the ARToolkit documentation it would require a marker the size of Roomba itself to detect it from a distance of two meters.

An approach for tracking a robot by means of an active marker is presented in [CTF05]. The active marker used consists of infra-red LEDs. In contrast to our approach, the tracking cameras are located on fixed positions, so their positions are constantly known. The cameras used operate in the visible light spectrum, which complicates the procedure of detecting the marker in the camera image. Our approach bypasses this step by the application of a specialized infra-red camera. Although specialized, the camera we use is a low-cost commercial product.

3 Our Approach

We have implemented an autonomous hierarchical multi-robot system for household chores. The main robot localizes itself and the slave robot. Note, that the localization and tracking of the slave robot is achieved in real time from a mobile platform. To track the slave, the master uses an infra-red (IR) camera. It allows it to detect the active marker consisting of IR LEDs, which is placed on top of the slave. In this scenario the master robot remotely controls the slave robot Roomba, a commercial vacuum cleaning robot built and distributed by iRobot². By determining its own position and the position of the slave, the master navigates the cleaning robot safely to the desired location. Using the slave's capabilities the master fulfils a task previously given by a human user. Since the slave robot gets all required information from the master, it has no need for expensive sensors. This allows for concentrating a major part of the hardware expenses on the master robot. Furthermore, by transferring tasks to an auxiliary system, some of them can be executed more efficiently. In our example, the cleaning robot reaches even narrow parts and corners of a room.

¹ www.hitl.washington.edu/artoolkit

² www.irobot.com

3.1 Hardware Design

Our mobile platform “Lisa” has already participated successfully in RoboCup@Home in 2008 and 2009. In this work, it takes the role of a master robot. Lisa localizes itself using a laser range finder. A pan-tilt unit (PTU) is mounted on top of it. It can be rotated horizontally by 180° in each direction. Vertically, its possible positions range from 30° upwards to 80° downwards. We are using the IR camera of a Nintendo Wiimote mounted on the PTU to localize the slave robot.

The role of the slave robot is taken by Roomba [TD07]. Several ways to expand Roomba’s abilities are presented in [Kur06]. The wireless connection to the slave robot as implemented here, was inspired by this book. We placed an ASUS WL500G Premium router on the slave. Thus, the master can send commands by opening a network socket to the router on top of Roomba. One of the USB ports of the router is used to communicate with the robot. Unfortunately, the original router firmware does not allow arbitrary data transfer through the USB ports. Instead, we had to use a modified firmware, called dd-wrt³, a mini Linux system. To communicate with Roomba, we use the Roomba-Open-Interface protocol specification provided by iRobot. This open protocol allows to send control commands to Roomba via a serial connection. The hardware design is illustrated in Fig. 1.

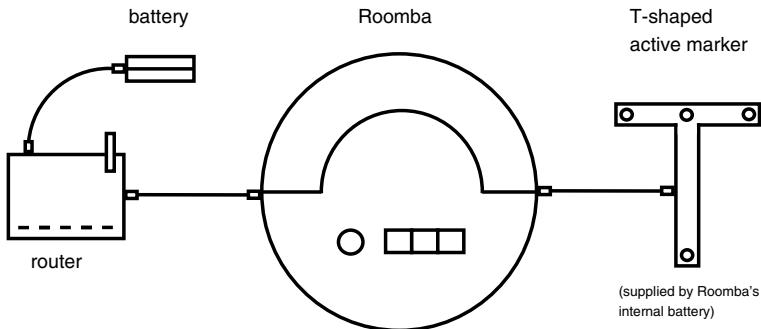


Fig. 1. Design of the hardware on the slave robot

3.2 Tracking

The projects of Johnny Chung Lee⁴ inspired us to use a Wiimote, a game controller for the Nintendo Wii, for tracking. To communicate with the Wiimote we use the open library cwiid⁵. This controller includes an infra-red camera and circuits for elementary image processing. It tracks up to four infra-red sources simultaneously.

³ www.dd-wrt.com

⁴ johnnylee.net/projects/wii

⁵ abstrakraft.org/cwiid/

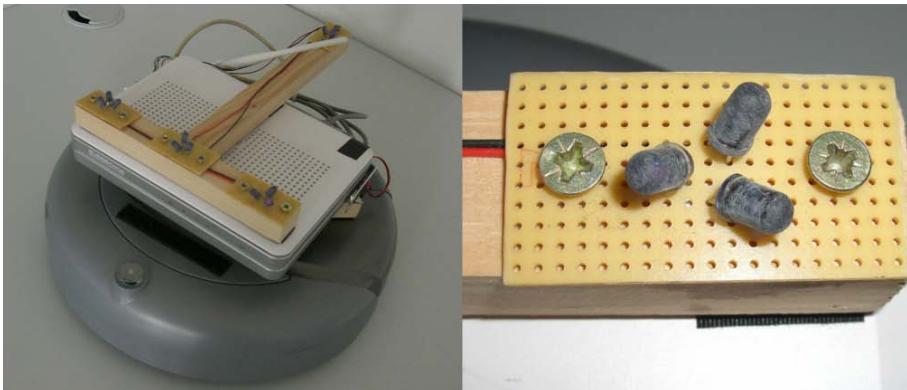


Fig. 2. Modified Roomba (left), One out of four infra-red LED groups (right)

This is sufficient to determine the position and orientation of one slave robot. If more than one slave robot is to be used, each one has to carry a marker with a different arrangement of IR sources. In order to distinguish these markers and thus the slave robots, it is necessary to use a different IR camera. In that case the IR camera's image has to pass more complex computations in order to find and identify the different markers.

We place an active T-shaped marker with IR-LEDs on top of the slave robot. The IR-LEDs that are used in the Wii sensor bar can be tracked from a distance of five meters if pointed vertically at the Wiimote. Since we expect to point at the slave robot's infra-red marker at angles between 25 to 55 degrees to the vertical (i.e. the PTU is tilted 35 to 65 degrees), the light sources have to be slightly modified. To increase the maximum tracking distance, the four infra-red sources are composed of three LEDs. Each one is tilted outwards at an angle of 40 degrees (see Fig. 2) thus emitting the IR light into the expected direction of the IR camera.

As it is only sensitive to IR light, the infra-red camera allows to bypass the task of (possibly erroneous) image processing in order to find the slave robot: the only thing contained in the image is the active marker. Hence computation time can be saved. With the detected marker the relative position of the slave to the main robot can be calculated. To accomplish this, we use a graph-based representation of the robot geometry. It is permanently updated to reflect the movements of actuators or the robot itself (see Fig. 3). As the IR camera is attached to the PTU, its position and orientation is always known in local robot coordinates.

The IR camera's sensor⁶ yields 2D coordinates of the detected infra-red sources. Depth information cannot be obtained in this manner. Therefore we face the challenge of converting 2D sensor coordinates to 3D local robot coordinates.

⁶ Note that we ignore the radial distortion of the camera in the following transformations.

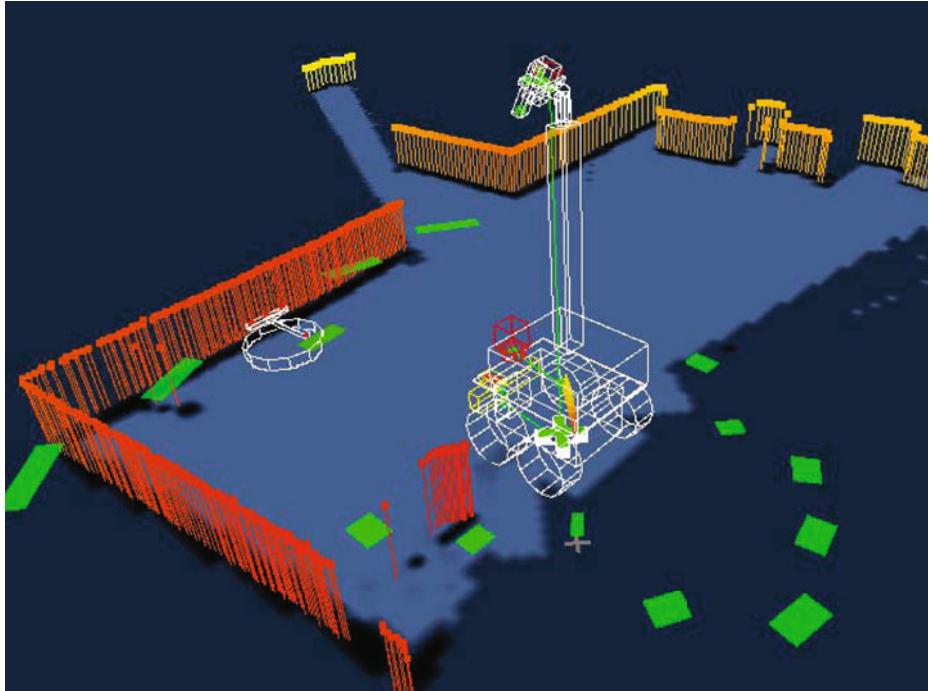


Fig. 3. The main robot's belief about its state and the slave's pose

First we need to determine the distance of the IR camera to the IR sources before applying the scene graph's transformation. This is done as follows: Let x and y be the 2D coordinates of a detected IR source. Furthermore, $w = 1024$ and $h = 768$ describe the resolution of the camera's sensor. Then we apply the following transformation to the 2D coordinates:

$$p_x = \frac{2 \cdot x}{w} - 1 \quad p_y = \frac{2 \cdot y}{h} - 1 \quad (1)$$

As no depth information is available, we assign an arbitrary value unequal to zero to p_z . Using the scene graph, this point \mathbf{p} is transformed into local coordinates. Now the transformed coordinates are located somewhere on a straight line through the IR camera and the IR LEDs - depending on the value that was assigned to p_z . Since the slave robot always moves on the ground, we can now determine its position. We calculate the vector \mathbf{v} from the camera's position \mathbf{c} through point \mathbf{p} and intersect the corresponding line with the plane the active marker is in, which is parallel to the ground plane. Let the height of this plane be h_{led} . Then the line parameter r can be determined as

$$r = \frac{h_{led} - c_z}{v_z} \quad (2)$$

where $v_z = c_z - p_z$. Thus, the position of the IR light source in local coordinates is

$$\begin{pmatrix} x_r \\ y_r \\ z_r \end{pmatrix} = \begin{pmatrix} c_x + r \cdot v_x \\ c_y + r \cdot v_y \\ h_{led} \end{pmatrix} \quad (3)$$

Since \mathbf{v} always needs to intersect with the ground plane, this method works only if the IR camera is tilted towards the ground. Whenever the computation yields an upwards directed vector or a point further away than $2.5m$, we have detected an erroneous infra-red source (for example a sun reflection). In this case the point is ignored.

In this manner all of the four IR light sources on the active marker are converted. Now we make use of the T-shaped marker to determine the slave's orientation and obtain its pose in local robot coordinates of the main robot.

3.3 SLAM

Based on laser scans and odometry information, the main robot continuously generates a grid map of its environment and localizes itself in it. It is based on the Hyper Particle Filter (HPF) concept [PP09]. The HPF contains a fixed number of particle filters running in parallel, each one with its own map and distribution of elementary particles. Each elementary particle contains information about the robots position and orientation. For each particle filter, some measurements are randomly ignored, so that the results vary. The maps generated by each particle filter are weighted according to their contrast, which is an indicator of quality.

3.4 Path Finding and Coordination during Navigation

Using the knowledge of the master's position, the global position of the slave robot can be determined, thus providing a basis for path planning. The path planning algorithm is described in [WP07].

The slave robot always moves in two steps. First, it aligns to the next waypoint. Then, it moves to a certain distance in the new direction. After it has finished its movement, the slave's position is determined again so that it can move to the following waypoint. During the navigation phase, the master robot constantly adjusts the IR camera to the slave's position. If the slave moves too far away or the intervisibility is lost, the master sends a stop command. In this case it approaches the slave's last known position and tries to find the slave by systematically tilting and panning its PTU.

4 Results

To evaluate the infra-red marker we use the program wmgui⁷. It provides a graphical user interface in which it displays the detected infra-red sources by the Wiimote and their intensities.

If pointed vertically at a LED group the Wiimote detects the infra-red light up to a distance of $3.5m$. The minimal distance is $30cm$. The whole active marker

⁷ packages.debian.org/de/sid/wmgui

is detected up to an angle of 70 degrees to the vertical in a maximum distance of two meters. At higher angles or in greater distances not all of the four light sources are reliably detected. When in operation, the IR camera is pointed at the marker in angles between 55 and 25 degrees to the vertical, which corresponds to PTU tilt angles of 35 to 65 degrees. The distance to the slave robot is limited to two meters, whereas the angle does not restrict the operation.

Depending on the angle and orientation the slave is located in, some of its IR LEDs can be detected up to distances of three meters. Since not all of the four IR sources can be detected at this distance, a safe navigation and precise pose estimation is not possible.

The master is able to find the slave on its own in short time, if the intervisibility is lost. This can be achieved due to an immediate stop command, that is send to the slave and the systematic search using the PTU. However, when the slave is taken away and put on the ground further than two meters away from the master, it loses its position and cannot continue operation.

When the cleaning robot moves under a table or a comparable obstacle, the intervisibility is lost and cannot be recovered. Furthermore, this approach is limited to even floors, since the pose estimation of the slave robot relies on this fact.

5 Conclusion

We successfully implemented the interaction of two robots based on the master-slave principle. The master can reliably track the slave robot up to a distance of two meters and navigate it to a target in arbitrary distance. It operates indoors on even floors and relies on the intervisibility between the robots.

To avoid the necessity of permanent intervisibility, the slave robot's odometry data could be used to compensate for short-time loss of intervisibility. In our scenario, this extension would allow the cleaning of occluded areas, e.g. under a table or bed. The tracking system could also be extended by cameras installed at fixed positions in the scenario. After guiding the slave to a monitored area, the master robot could pursue other tasks while tracking the slave via the fixed cameras.

Although we use specific robots for our experiments, the approach described in this paper can be seen as a general solution for integrating the current generation of commercial household robots with a higher-level autonomous system.

References

- [BMF⁺00] Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: IEEE International Conference on Robotics and Automation (2000)
- [CTF05] Cassinis, R., Tampalini, F., Fedrigotti, R.: Active markers for outdoor and indoor robot localization (2005)
- [FBE⁺09] Faber, F., Bennewitz, M., Eppner, C., Görög, A., Gonsior, C., Joho, D., Schreiber, M., Behnke, S.: The humanoid museum tour guide robot-inho (2009)

- [How06] Howard, A.: Multi-robot simultaneous localization and mapping using particle filters (2006)
- [IB98] Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision* 29(1), 5–28 (1998)
- [JN02] Jäger, M., Nebel, B.: Dynamic decentralized area partitioning for cooperating cleaning robots (2002)
- [Kur06] Kurt, T.E.: Hacking Roomba: ExtremeTech. John Wiley & Sons, Inc., New York (2006)
- [LAL⁺04] Lemaire, T., Alami, R., Lacroix, S., LAAS, C., Toulouse, F.: A distributed tasks allocation scheme in multi-UAV context. In: Proceedings of IEEE International Conference on Robotics and Automation, ICRA 2004, vol. 4 (2004)
- [NdCVVR97] Neumann de Carvalho, R., Vidal, H.A., Vierira, P., Ribeiro, M.I.: Complete coverage path planning and guidance for cleaning robots. In: IEEE Int. Symposium on Industrial Electronics (July 1997)
- [Pel08] Pellenz, J.: Mapping and map scoring at the robocuprescue competition. In: Quantitative Performance Evaluation of Navigation Solutions for Mobile Robots (RSS 2008, Workshop CD) (2008)
- [PGP09] Pellenz, J., Gossow, D., Paulus, D.: Robbie: A fully autonomous robot for robocup rescue. *Advanced Robotics (Robotics Society of Japan)* 23(9), 1159–1177 (2009)
- [PP09] Pellenz, J., Paulus, D.: Stable mapping using a hyper particle filter. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 252–263. Springer, Heidelberg (2010)
- [SCPZ04] Seop, J., Choi, Y.H., Park, J.B., Zheng, Y.F.: Complete coverage navigation of cleaning robots using triangular-cell-based map. *IEEE Transactions on Industrial Electronics* 51(3), 6 (2004)
- [SGK⁺09] Stückler, J., Gräve, K., Kläß, J., Muszynski, S., Schreiber, M., Tischler, O., Waldukat, R., Behnke, S.: Dynamaid: Towards a personal robot that helps with household chores. In: Proceedings of RSS 2009 Workshop on Mobile Manipulation in Human Environments, Seattle (June 2009)
- [TD07] Tribelhorn, B., Dodds, Z.: Evaluating the roomba: A low-cost, ubiquitous platform for robotics research and education (2007)
- [TPM03] Tavakoli, M., Patel, R.V., Moallem, M.: A force reflective master-slave system for minimally invasive surgery. In: Proceedings of the 2003 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (October 2003)
- [VV03] Vail, D., Veloso, M.: Multi-robot dynamic role assignment and coordination through shared potential fields (2003)
- [WP07] Wirth, S., Pellenz, J.: Exploration transform: A stable exploring algorithm for robots in rescue environments. In: Workshop on Safety, Security, and Rescue Robotics, pp. 1–5 (September 2007), <http://sied.dis.uniroma1.it/ssrr07/>
- [YTEM02] Yamano, I., Takemura, K., Endo, K., Maeno, T.: Method for controlling master-slave robots using switching and elastic elements. In: Proceedings of the 2002 IEEE International Conference on Robotics & Automation (May 2002)

Biped Walking Using Coronal and Sagittal Movements Based on Truncated Fourier Series

Nima Shafii^{1,2}, Luís Paulo Reis^{1,2}, and Nuno Lau^{3,4}

¹ DEI/FEUP – Informatics Engineering Department, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias s/n, 4200 465 Porto, Portugal

² LIACC – Artificial Intelligence and Computer Science Lab., Porto, Portugal

³ UA – University of Aveiro, Campus Universitário de Santiago, 3810 193 Aveiro, Portugal

⁴ IEETA – Institute of Electronics and Telematics Engineering of Aveiro, Portugal

nima.shafii@fe.up.pt, lpreis@fe.up.pt, nunolau@ua.pt

Abstract. Biped walking by using all joint movements and DOFs in both directions (sagittal plane and coronal plane) is one of the most complicated research topics in robotics. In this paper, angular trajectories of a stable biped walking for a humanoid robot are generated by a Truncated Fourier Series (TFS) approach. The movements of legs and arms in sagittal plane are implemented by an optimized gait generator and a new model is proposed that can also produce the movement of legs in coronal plane based on TFS. Particle Swarm Optimization (PSO) is used to find the best angular trajectories and optimize TFS. Experimental results show that the using joints movements in sagittal and coronal planes to compose the walking skill allowed the biped robot to walk faster than previous methods that only used the joints in sagittal plane.

Keywords: Bipedal Locomotion, Gait Generation, Particle Swarm Optimization.

1 Introduction

In competitive non-deterministic environments like RoboCup soccer humanoid robot leagues, flexible, fast and robust biped walking is one of the keys to win a match. Such movements must accomplish different requirements. For example, in case the target is far away, the humanoid robot must be capable of covering large distances in a short time and when the target is close it must be precise and robust in its approach. In this task, a real 3D bipedal locomotion by using all joints enabling a flexible robot movement has an important role. In the literature, several approaches for bipedal locomotion have been presented until now. They can be divided in two main groups, model based and model free. Model based methods need to obtain and process accurate information of the dynamics of the robot like its parts positions and velocities and their mass after that a controller is built for it. "Zero Moment Point" (ZMP) [1] and "Inverted Pendulum" [2] are two methods that belong to this approach. In many ZMP-based trajectory planning approaches, motion planning is presume and performed in the Cartesian space and has some motion assumptions[3]. Therefore it does not give much freedom for having human like motion.

Model-free approaches use sensory information for developing the humanoid motions. Passive Dynamic Walking (PDW) [4], Central Pattern Generator (CPG) [5] and Ballistic Walking [6] are the most known methods of model-free approach. Passive dynamic walking tries to mimic natural motion in a robot. Using PDW, the robot does not have any actuators' model and it walks just by utilizing the gravity force and can go down a shallow inclined ramp only with its own mechanical dynamics without using any actuation or external control. The Ballistic Walking is originated from the simple human walking model and is based on the observation that the muscles of the swing leg are activated only at the beginning and the end of the swing phase when real humans walk.

In the CPG-based approach, bipedal trajectory generation has been achieved with the use of special neural oscillators. Using non-linear equations to model the neural activities, the oscillators can generate rhythmic walking patterns. Researchers usually focus on complex mathematical models like Hopf [7] or Matsuoka [8] to model these neural activities and generate rhythmic walk patterns (Gait). The walk model developed using CPG is flexible enough to adapt to the environment. However, it is difficult to design the relation and feedback pathways of the neural oscillators, and tune the required parameters in order to achieve the desired walking characteristics manually [9].

In 2007 Truncated Fourier Series Formulation method (TFS) was used as a gait generator and CPG in bipedal locomotion [9]. TFS together with a ZMP stability indicator was used to prove that it could generate suitable angular trajectories for controlling bipedal locomotion but it was not implemented on a real robot [9]. In 2009, an optimized gait generator based on TFS was implemented in a simulated humanoid robot and TFS parameters were also reduced by 2 dimensions (down to 6 dimensions) [10]. Shafii extended the basic TFS enabling the generation of arm angular trajectories that provide smooth and robust walking, also a new method was used to refine signals for reducing the role of inertia to improve the speed and robustness of the robot [11].

In this paper the results of the two previous papers are used to produce walking movements on sagittal plane and also on coronal plane. The method was tested on a simulated NAO humanoid and the experiments were performed using Rcssserver3d [12], a generic three-dimensional simulator which is based on Spark and Open Dynamics Engine (ODE). The robot model has 22 DOF with a height of about 57cm, and a mass of 4.5kg.

The paper structure is as follows. First, optimized TFS gait generator is introduced to generate walking movement on the sagittal plane. Arm angular trajectories generator and the method for reducing the role of inertia are also explained. Then a new model of hip angular trajectory generator based on TFS is introduced which can produce the leg's movement on the coronal plane (Y direction). Particle Swarm Optimization (PSO) is used to optimize the produced signals, to overcome inherent noise of the simulator and Resampling algorithm is used to improve robustness in nondeterministic environments. At the end of the article, results of this approach are presented and the efficiency of the method on producing trajectories to walk a robot in the forward direction are shown.

2 Movements in Sagittal Plane

There are three DOFs in each leg move in sagittal plane: one in the hip, one in the ankle and one at the knee. In this work, similar to [13], swing foot in sagittal plane was kept parallel to the ground by using the ankle joint. This is done in order to avoid swing foot colliding with the ground. Therefore, ankle trajectory can be calculated by hip and knee trajectories and ankle DOF parameters are eliminated. Trunk sagittal and coronal plane motion is fairly repeatable [14] therefore Fourier series can be used.

2.1 An Optimized Gait Generator for Leg's Movement

In this model, legs joint angular trajectories in sagittal plane are divided in two parts; the upper portion and the lower portion. Let us define C_h as the offset of the hip trajectory and C_k as the offset of the knee trajectory. From t_1 to t_2 the left leg is considered as supporting leg and the variation of its knee angle is so little that it can be assumed fixed. This phase of walking is named knee lock phase. In addition, the two leg trajectories signals are repeated once for swinging the right leg and then for swinging the left one hence by producing the trajectory of one leg the other leg's trajectory can be calculated. The trajectories for both legs are identical in shape but are shifted in time relative to each other by half of the walking period. The TFS for generating each portion of hip and knee trajectories are formulated below.

$$\begin{aligned}\theta_h^+ &= \sum_{i=1}^n A_i \cdot \sin(iw_h t) + c_h, w_h = \frac{2\pi}{T_h} \\ \theta_h^- &= \sum_{i=1}^n B_i \cdot \sin(iw_h t) + c_h, w_h = \frac{2\pi}{T_h} \\ \theta_k^+ &= \sum_{i=1}^n C_i \cdot \sin(iw_k t) + c_k, w_k = w_h \\ \theta_k^- &= c_k \geq 0\end{aligned}\quad (1)$$

In these equations, the plus (+) sign represents the upper portion of walking trajectory and the minus (-) shows the lower portion. $i=1$ and A_i, B_i, C_i are constant coefficients for generating signals. The h and k subscripts stand for hip and knee respectively. C_h, C_k are signal offsets and T_h is assumed as the period of hip trajectory. Considering the fact that all joints in walking motion have equal movement frequency and stride rates are statistically equal, the equation $w_k = w_h = \frac{2\pi}{T_h}$ can be concluded. Parameter t_1 is the start time of lock phase for knee joint and parameter t_2 represents the end time of lock phase and in this duration of time $\theta_k^- = c_k \geq 0$.

According to [10], by specifying the start and end time of the lock phase, two parameters of t_1, t_2 could be eliminated. Therefore the number of variable for optimization to produce legs movement in sagittal plane decreased to 6. This elimination has many advantages such as: reducing the search space of optimization problem and increasing the convergence speed of PSO algorithm.

2.2 Modeling of Arm Motion in Sagittal Plane

In sagittal plane, during human walking, the arms normally swing in opposite manner to legs, which helps to balance the angular momentum generated in the lower body [15]. Humans swing their arms close to 180° out of phase with their respective legs during walking [16]. Fig. 1 shows the trajectory of legs and arm swings and the relation between them in a stable straight walking [15]. It is shown that Trajectory of arms is similar to sinusoidal signal with same frequency of legs.

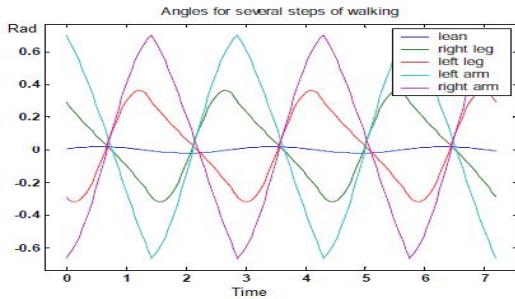


Fig. 1. Trajectories of legs and arms

The walking speed has a strong effect on arm swinging during gait. By increasing gait's speed, the arms may swing higher and faster to reduce the effects of longer, quicker steps by the legs [17]. It can be expected that the utilization of arm swing provides good performance to yaw moment stability, and recovery from stumbling. The effectiveness of this method is confirmed with an improvement of the accuracy of straight walking at different speeds. As has been shown, the trajectory of arms is a sinusoidal signal, therefore, to produce the angular trajectories of arms swinging, it is enough to obtain proper parameters for the equation 2.

$$f(t) = A \sin(\omega_{arm} t), \omega_{arm} = \omega_{arm} \quad (2)$$

In equation 2, A and w are assumed as the amplitude and frequency of the signal, respectively. In addition the shift phase of the two arm trajectories signals is half of the period of each signal, so by producing the trajectory of one arm the other arm's trajectory can be calculated. Since legs and arms have the same frequency, ω_{arms} can be considered equal to ω_{legs} . According to the fact that the angle of arms is zero at the start of walking, shift phase factor is assumed as 0.

According to equation 2 and the previous discussion, only the proper value of parameter A must be obtained. To find the best value for A we consider this parameter together with legs trajectory parameters in the optimization problem.

3 Increasing Speed at the Start Time of Walking in TFS

According to [10], when the robot starts walking the walking speed and amplitude should be increased in a controlled way from zero to their maximum values, hence improving walking stability and allowing higher final speeds. We implemented a

model for the robot to walk from smaller gait with lower amplitude to bigger gait with higher speed and acceleration. In this model a linear equation is used to lead the robot to increase the amplitude of trajectory linearly from zero (stop state) to desired angular trajectories. T is assumed as a parameter to determine how much time is needed for this increment algorithm to reach these desired trajectories. All angular trajectories such as arms and legs will be multiplied by the product of the following equation.

$$\begin{aligned} K &= \text{time } / T, \text{time } < T \\ K &= 1, \text{time } \geq T \end{aligned} \quad (3)$$

4 Movements in Coronal plane

The range of motion in the coronal and transverse plane is less than that seen in the sagittal plane [18] but it has an important role to keep the balance of walking and reach the highest speed of walking. The range of its motion depends on the speed of walking, and at higher speeds this range is smaller. Coronal plane movements are periodic motions [14]. Abduction and adduction are terms for movements of limbs relative to the coronal plane. To produce legs' motion in coronal plane and also considering keeping the balance of robot, we proposed a walking sequences and scenario (Fig. 2). It illustrates the walking sequence in a walking period. θ is assumed as the maximum of legs movement. Like in the previous section in coronal plane, feet were kept parallel to the ground in order to avoid collision and considering of the fact that just one of each hip's joint moves each time, the angle of ankle is equal to the hip's angle of the opposite leg.

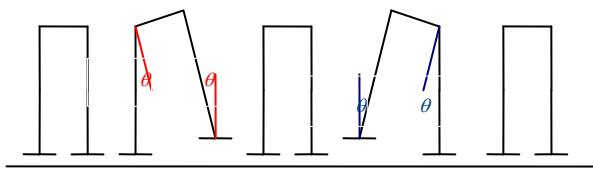


Fig. 2. Coronal plane view of proposed walking Sequence

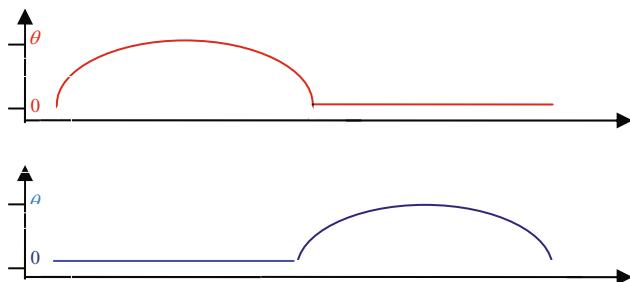


Fig. 3. Left leg and right leg hips angular trajectories

Considering the walking sequence, Fig. 3 can be assumed as the hip angular trajectory in one period of walking. It is a sinusoidal signal that has a lock phase at zero degree. Therefore in order to produce the proper angular trajectories to move the hips in coronal plane, proper parameters for the equation 4 must be obtained.

$$\begin{aligned} f(t) &= H \sin(wt), t < T_h / 2 \\ f(t) &= 0, t > T_h / 2 \end{aligned} \quad (4)$$

In equation 4, H and w are assumed as amplitude and frequency of signal respectively and T_h is assumed a period of hip. As mentioned before, ankle trajectories can be calculated from hip trajectories and as it is shown, left and right hip angular trajectories are the same but with a phase shift of $-\pi$. The period of walking in sagittal plane and coronal plane is equal. Therefore T_h and w are eliminated in this method and for producing the proper abduction and adduction, the proper value of H parameter must be found.

According to the fact that movements in coronal plane decrease at higher speeds of walking, the movement of legs in coronal plane must be reduced when increasing the speed of walking. Therefore all angular trajectories generated by above model will be multiplied by the reverse product of equation 3.

5 Implementation

Bipedal walking is known as a complicated motion since many factors affect walking style and stability such as robot's kinematics and dynamics, collision between feet and the ground. In such a complex motion, relation between Gait trajectory and walking characteristic is nonlinear. In this approach the best parameters to generate angular trajectories for bipedal locomotion must be found. According to [19], for this kind of optimization problem, Particle Swarm Optimization can achieve better results. Therefore PSO seems to be an appropriate solution. In the following sections some brief details about the PSO algorithm are explained.

5.1 PSO Algorithm

The PSO algorithm contains the three following parts: generating primitive particle's positions and velocities, velocity update and position update [20].

Equations (5) and (6) are used to initialize particles in which Δt is the constant time increment. Using upper and lower bounds on the design variables values, X_{\min} and X_{\max} , the positions, X_k^i and velocities, V_k^i of the initial swarm of particles can be first generated randomly. The swarm size will be denoted by N . The positions and velocities are given in a vector format where the superscript and subscript denote the i^{th} particle at time k .

$$X_0^i = X_{\min} + Rand(X_{\max} - X_{\min}) \quad (5)$$

$$V_0^i = \frac{X_{\min} + \text{Rand}(X_{\max} - X_{\min})}{\Delta t} = \frac{\text{Position}}{\text{time}} \quad (6)$$

Updating Velocities has an important role in finding the new search direction. current motion, particle own memory, and swarm influence, in a summation approach as shown in Equation below (7) influence on the calculation of new search direction. This equation consists of three weight factors, namely, inertia factor, w , self confidence factor, C_1 , and swarm confidence factor, C_2 . P_k^g, P^i are the best position of each particle over time and the best global value in the current swarm respectively.

$$\underbrace{V_{k+1}^i}_{\substack{\text{Velocity of Particle} \\ i \text{ at time } k+1}} = \underbrace{w^{[0.4, 1.4]}}_{\substack{\text{Current Motion}}} + \underbrace{C_1^{[1, 2]} \text{Rand} \frac{(P^i - X_k^i)}{\Delta t}}_{\substack{\text{Particle Memory Influence}}} + \underbrace{C_2^{[1.5, 2]} \text{Rand} \frac{(P_k^g - X_k^i)}{\Delta t}}_{\substack{\text{Swarm Influence}}} \quad (7)$$

Utilizing a nonlinear decreasing inertia weight as a dynamic inertia weight significantly improves its performance through the parameter study of inertia weight [21]. This nonlinear distribution of inertia weight is expressed as follow:

$$w = w_{\text{init}} * U^{-k} \quad (8)$$

W_{init} is the initial inertia weight value selected in the range [0, 1] and U is a constant value in the range [1.0001, 1.005], and k is the iteration number.

Finally the position of each particle in the swarm can be updated by using the current particle position and its own updated velocity vector as shown in the following Equation.

$$X_{k+1}^i = X_k^i + V_{k+1}^i \Delta t \quad (9)$$

The PSO algorithm may be described as follows:

```

Initialize Position ( $X_0$ ) and Velocity of  $N$  particles
according to equation (4 and 5)
 $P^1 = X_0$ 
DO
     $k=1$ 
    FOR  $i = 1$  to  $N$  particles
        IF  $f(X_i) < f(P^i)$  THEN  $P^i = X_i$ 
         $P_k^g = \min (P)$ 
        Calculate new velocity of the particle according
        to equation (6, 7 and 8)
        Calculate new position of particle according to
        equation (9).
    ENDFOR
     $k=k+1$ 
UNTIL a sufficient good criterion (usually a desirable
fitness or a maximum number of iterations) is met.

```

5.2 PSO Implementation

In PSO, the parameters of the problems are coded into a finite length of string as a particle. According to above sections, for producing movements in sagittal plane, TFS has 6 parameters to generate legs joints angular trajectories and 2 parameters are assumed to swing arms and to increase the speed of walking. There is also one parameter to produce proper legs' movement in coronal plane. Therefore there is a 9-dimension search space for the PSO to find the optimum solution.

Angular trajectories produced by each particle are applied to a simulated robot to optimize its walking capabilities. To use angular trajectory for walking, all individual robot's joints attempt to drive towards their target angles using simple proportional derivative (PD) controllers. To enable the robot with a fast walking skill a fitness function based on robot's straight movement in limited action time is considered. First the robot is initialized in $x=y=0$ (0, 0) and it walks for 15 seconds. After that, the fitness function is calculated whenever the robot falls or the time duration for walking is over. The fitness function formulation is simply expressed as the distance travelled by the robot along the x axis.

Due to the fact that there is noise in simulated robot's actuators and sensors, training walking task in this approach can be viewed as an optimization problem in a noisy and nondeterministic environment. Resampling is one of the techniques to improve the performance of evolutionary algorithms (EAs) in noisy environment [23]. In Resampling, the individual set of parameters (particle) y_i , the fitness $F(y_i)$ is measured m times and averaged yielding fitness. According to (10) the noise strength of \bar{F} is reduced by a factor \sqrt{m} .

$$\bar{F}(y_i) = \frac{1}{m} \sum_{k=1}^m F(y_i), y(i) = \text{const.} \Rightarrow \bar{\sigma}_e = \sqrt{\text{Var}[\bar{F}(y_i)]} = \frac{\sigma_e}{\sqrt{m}} \quad (10)$$

Since particles may not satisfy some constraints after updating position procedure, constraint handling is a vital step in PSO algorithm. There are many obvious constraints on parameters in this study (i.e. time parameters in TFS must be positive). Therefore Pareto [24] with multi-objective modeling is used for handling constraints.

In Pareto, a solution, $x(2)$, is dominated by solution, $x(1)$, if $x(1)$ is not worse than $x(2)$ in all objectives, and for at least one of the objectives, $x(1)$ is strictly better than $x(2)$. Without loss of generality, these conditions can be expressed as follows for the case where all of the objective functions are to be minimized:

$$fm(x(1)) \leq fm(x(2)) \text{ for } \forall m = 1, 2, \dots, M \quad \text{and}$$

$$fm(x(1)) < fm(x(2)) \quad \text{for some } m.$$

Each constraint is assumed as an object in which parameters must be satisfied. So according to Pareto method, a particle can be considered to find P_i , P_{gk} when it satisfies objects and constraints. Therefore calculating fitness for particles that cannot satisfy constraint is not necessary.

We have considered various values for each parameter of the algorithm and tried all possible combinations. Finally we chose the best combination of the parameters regarding the dynamic inertia weight and test results that C_1 and C_2 are assumed as 1,

1.5 , w_{init} as 0.8 , U as 1.0002 and Δt as 1 , respectively. We have also used for our experiments a particle swarm composed by 100 particles ($N = 100$), a maximum iteration of 10 and a Resampling factor m assumed as 3 .

6 Results

To compare the presented method with Basic TFS method which uses the joints movement just in sagittal plane, both of them were tested using the same system and the same specification. We also optimized both methods by utilizing PSO with equal specifications and with the same fitness function.

Using basic TFS with 8 parameters and running PSO algorithm on a Pentium IV 3 GHz Core 2 Duo machine with 2 GB of physical memory, 3000 trials were performed in 4 hours. Finally the robot could walk $8.6m$ in $15s$ with average body speed of $0.57m/s$ and period of $0.41s$ for each step. Fig. 4 shows the average and best fitness values during these 10 generations.

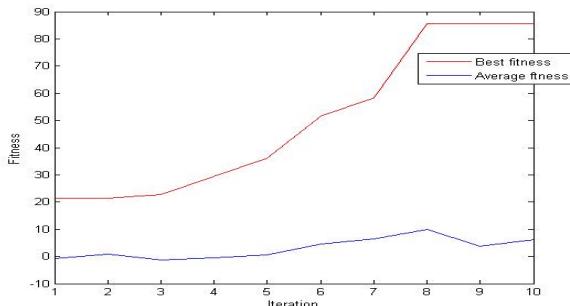


Fig. 4. PSO convergence for previous TFS

Using the new approach presented in this paper, after 3300 trails and 5 hours from starting PSO in a machine with the same specifications, the robot could walk $11.5 m$ in $15 s$. Average and best fitness are shown in fig. 5.

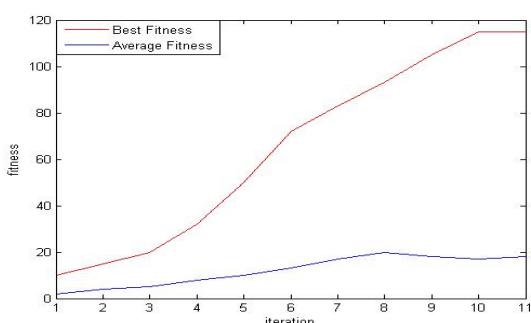


Fig. 5. PSO convergence for new approach and TFS model

Robot could walk with average body speed of 0.77 m/s by using the TFS with arm swing and increasing speed technique. The learned trajectories of left hip and knee after robot started to walk are shown in Fig. 6.A. It is determined that robot increased its speed in 0.20 m/s. The learned trajectory of left arm is also shown in Fig. 6.B. and finally, the learned trajectory of left hip in coronal plane for abduction and adduction movements is also shown in Fig. 6.C.

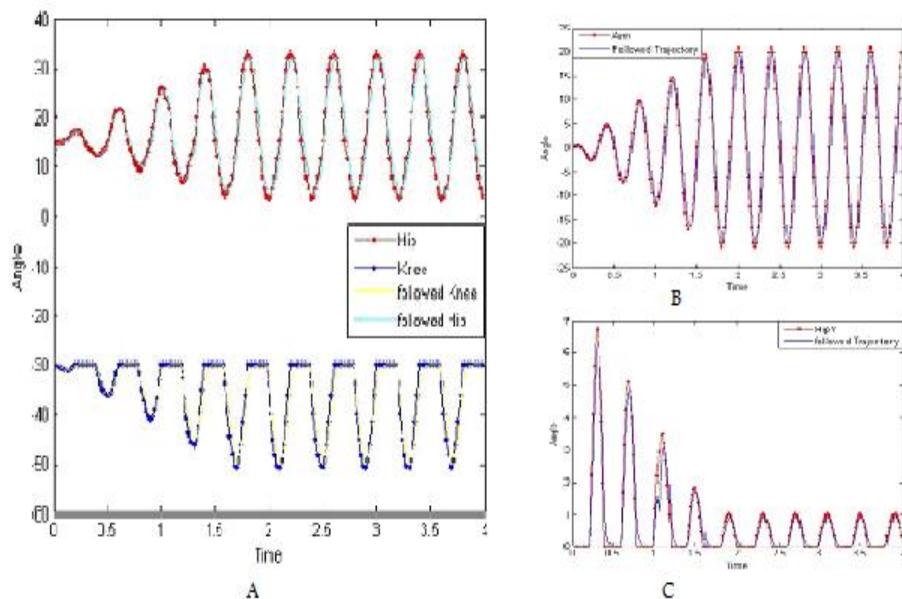


Fig. 6. A) Left Hip and Knee trajectories in the sagittal plane; B) Left Arm trajectory; C) Left hip trajectory in coronal plane (hipY)

After the learning procedure, the robot could walk with the average speed of 0.77 m/s. The best results of walking behavior for the teams that participated in RoboCup 2008 were chosen for the comparison [25]. Table 1, presents the comparison of the best results of RoboCup teams, compared with the proposed approach. Analyzing the table it is clear that forward walking achieved by our approach, may outperform the same skill of all teams analyzed except SEU.

Table 1. Comparison the average speed for forward walking in different teams (m/s)

	Proposed approach	FCPortugal	SEU	Wright Eagle	Bats
Forward Walking	0.77	0.51	1.20	0.67	0.43

7 Conclusions

This paper presented a model with 9 parameters for producing all walking angular trajectories that uses all joints' movements in coronal and sagittal plane. An optimized Truncated Fourier Series is used to produce leg movements in sagittal plane and a model for swinging arms. A new model was also used to generate legs walking movement in coronal plane. We are able to increase the speed and stability of the robot's walking when compared to previously TFS model by using this model and the method mentioned in sec. 4 which was used both in sagittal and coronal motion.

According to the fact that this approach is model free and based on robot learning, it is capable of being used on all kinds of humanoid robots with different specifications. In future works we would like to expand this model to produced turn motion and improve the approach so that the robot can walk in any direction.

Future work will be concerned with high level control of the walking behavior and navigation with obstacle avoidance in the context of FC Portugal team [25, 26, 27] for participation in RoboCup 2010 simulation 3D league.

Acknowledgements

The first author is supported by FCT under grant SFRH/BD/66597/2009. This work has been partially funded by FCT Project ACORD - Adaptative Coordination of Robotic Teams (PTDC/EIA/70695/2006).

References

1. Vukobratovic, M., Borovac, B., Surla, D., Stotic, D.: *Biped Locomotion Dynamics Stability Control and Application*. Springer, London (1990)
2. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., Hirukawa, H.: The 3D linear inverted pendulum mode A simple modeling for a biped walking pattern generation. In: Proc. of the 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 239–246 (2001)
3. Chevallereau, C., Aoustin, Y.: Optimal reference trajectories for walking and running of a biped robot. *Robotica* 19, 557–569 (2001)
4. McGeer, T.: Passive dynamic walking. *International Journal of Robotics Research* 9(2), 62–82 (1990)
5. Pinto, C., Golubitsky, M.: Central Pattern Generator for Bipedal locomotion. *J. Math. Biol.* 53, 474–489 (2006)
6. Ogino, M., Hosoda, K., Asada, M.: Learning Energy Efficient Walking with Ballistic walking. *Adaptive Motion of Animals and Machines*, pp.155–164 (2006)
7. Buchli, J., Iida, F., Ijspeert, A.J.: Finding Resonance: Adaptive Frequency oscillators for Dynamic Legged Locomotion. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3903–3910 (2006)
8. Matsuoka, K.: Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biol. Cybern.* 52, 367–376 (1985)
9. Yang, L., Chew, C.M., Poo, A.N.: Adjustable Bipedal Gait Generation using Genetic Algorithm Optimized fourier Series Formulation. In: Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4435–4440 (2006)

10. Shafii, N., Javadi, M.H., Kimiaghaham, B.: A Truncated Fourier Series with Genetic Algorithm for the control of Biped Locomotion. In: Proceeding of the 2009 IEEE/ASME International Conference on advanced intelligent Mechatronics, pp. 1781–1785 (2009)
11. Shafii, N., Khorsandian, A., Abdolmaleki, A., Jozi, B.: An Optimized Gait Generator Based on Fourier Series Towards Fast and Robust Biped Locomotion Involving Arms Swing. In: Proceeding of the 2009 IEEE International Conference on Automation and Logistics, pp. 2018–2023 (2009)
12. Boedecker, J.: Humanoid Robot Simulation and Walking Behaviour Development in the Spark Simulator Framework. Technical report, Artificial Intelligence Research University of Koblenz (2005)
13. Kagami, S., Mochimaru, M., Ehara, Y., Miyata, N., Nishiwaki, K., Kanade, T., Inoue, H.: Measurement and comparison of humanoid H7 walking with human being. *Robotics and Autonomous Sys.* 48, 177–187 (2003)
14. Konz, R.J., Fatone, S., Stine, R.L., Ganju, A., Gard, S.A., Ondra, S.L.: A Kinematic Model to Assess Spinal Motion During Walking. *J. J. Biomechanics* 31, E898–E906 (2006)
15. Elftman, H.: The function of the arms in walking. *J. Hum. Biol.* 11, 529–535 (1939)
16. Collins, S.H., Wisse, M., Ruina, A.: A 3-D passive dynamic walking robot with two legs and knees. *Int. J. Robot. Res.* 20, 607–615 (2001)
17. Murray, M.P., Sepic, S.B., Barnard, E.J.: Patterns of sagittal rotation of the upper limbs in walking. *Physical Therapy* 47, 272–284 (1967)
18. Thewlis, D., Richards, J., Hobbs, S.: The Appropriateness Of Methods Used To Calculate Joint Kinematics. *Journal of Biomechanics* 41, S320–S320 (2008)
19. Shafii, N., Aslani, S., Nezami, O.M., Shiry, S.: Evolution of Biped Walking Using Truncated Fourier Series and Particle Swarm Optimization. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 344–354. Springer, Heidelberg (2010)
20. Shi, Y., Eberhart, R.C.: Parameter selection in particle swarm optimization. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 591–600. Springer, Heidelberg (1998)
21. Jiao, B., Lian, Z., Gu, X.: A dynamic inertia weight particle swarm optimization algorithm. *J Chaos* 37, 698–705 (2008)
22. Beyer, H.G.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. *Comp. Methods Appl. Mech. Engrg.* 186, 239–267 (2000)
23. Coello, C.A., Pulido, G.T., Lechuga, M.S.: Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Trans. Evolutionary Computation* 8, 256–279 (2004)
24. Salman, A., Ahmad, I., Al-Madani, S.: Particle Swarm Optimization For Task Assignment Problem. *Microprocessorsand Microsystems* 26, 363–371 (2002)
25. Picado, H., Gestal, M., Lau, N., Reis, L.P., Tomé, A.M.: Automatic Generation of Biped Walk Behavior Using Genetic Algorithms. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M. (eds.) IWANN 2009. LNCS, vol. 5517, pp. 805–812. Springer, Heidelberg (2009)
26. Reis, L.P., Lau, N.: FC Portugal Team Description: RoboCup 2000 Simulation League Champion. In: Stone, P., Balch, T., Kraetzschmar, G. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 29–40. Springer, Heidelberg (2001)
27. Lau, N., Reis, L.P.: FC Portugal - High-level Coordination Methodologies in Soccer Robotics, Robotic Soccer, Book edited by Pedro Lima, Itech Education and Publishing, Vienna, Austria, pp. 167–192 (2007)

Realistic Simulation of Laser Range Finder Behavior in a Smoky Environment

Okke Formsma, Nick Dijkshoorn, Sander van Noort, and Arnoud Visser

Intelligent Systems Laboratory Amsterdam, Universiteit van Amsterdam,

Science Park 107, NL 1098 XG Amsterdam, The Netherlands

{okke.formsma,nick.dijkshoorn,alexander.vannoort}@student.uva.nl ,

a.visser@uva.nl

<http://www.science.uva.nl/research/isma>

Abstract. The Urban Search and Rescue Simulation used for RoboCup lacks realistic response of laser range finders on smoke. In this paper, the behavior of a Hokuyo and Sick laser range finder in a smoky environment is studied. The behavior of the lasers is among others a function of the visibility level, and in this article this function is quantified into an explicit model. This model is implemented in a simulation environment which is the basis of the Virtual Robot competition of the RoboCup Rescue League. The behavior of both real and virtual laser range finders is compared in a number of validation tests. The validation tests show that the behavior of the laser range finders in the simulation is consistent with the real world.

1 Introduction

Urban Search and Rescue (USAR) is a challenging application for teams of robots. The environment in a USAR setting is generally hostile. To make the work for human rescue teams safer, robots can be deployed to scout the environment. The robots can explore the area and create maps [1].

The RoboCup Rescue competition provides a demanding environment to benchmark research in this area [2]. There is a real robots league and a simulation league. In this paper, we will focus on the simulation league. The simulation league has a number of advantages over the real robots league. Firstly, no hardware needs to be purchased and the software is inexpensive. Secondly, there can be freely experimented with multiple robots without fear of destroying the expensive equipment in this hostile environment. Last but not least, in simulation it is easier to reproduce results, because one has full control over the environment. The downside of simulation is that all results have to be validated on real systems, as for instance done in this paper.

The simulation league has a software platform called USARSim which simulates robots on a physical realistic level. The latest incarnation of USARSim builds on the commercially available Unreal Engine 3.0 by Epic Games. This engine allows users to build environments through an editor. The engine also provides a programming language called Unreal Script. USARSim adds robots to the



Fig. 1. Snapshot of one of the worlds used in the 2009 competition of the RoboCup Rescue Simulation League

game [3]. These simulated robots resemble the robots of the Real Robots competitions closely. The movements and sensor responses of the simulated robots are very similar to the real robots [4,5,6,7].

The outdoor challenge for the RoboCup Rescue competition in 2009 was a railway station in which a train was derailed. In the scenario¹, some parts of the derailed train are on fire, producing a smoke column (see Fig. 1). This effect is only visual; the smoke does not influence the sensor readings of nearby robots in any way. This study is meant to include the influence of smoke to measurements of the laser range finders. Laser range finders are used by all teams in the competition for localization and mapping purposes [8]. The findings in this paper are also relevant outside the competition; the explicit model is complete enough to be implemented and contains details needed any study about the relationship between laser range finders and smoke.

In section 2, previous research on the behavior of laser range finders in smoky conditions is outlined. Section 3 describes the setup and findings of our own experiment with a laser range finder in a smoky environment. The implementation of this behavior in the simulation is detailed in section 4. In section 5 results of validation tests are presented.

2 Prior Work

While it is commonly known that laser range finders perform erratically under poor atmospheric conditions as dust or steam [9], little research has been conducted on the details of the conditions under which laser range finders (LRFs) fail and how they fail. Two studies relevant to this research have been found [10,11].

¹ This scenario can be downloaded from
<http://sourceforge.net/projects/usarsim/>

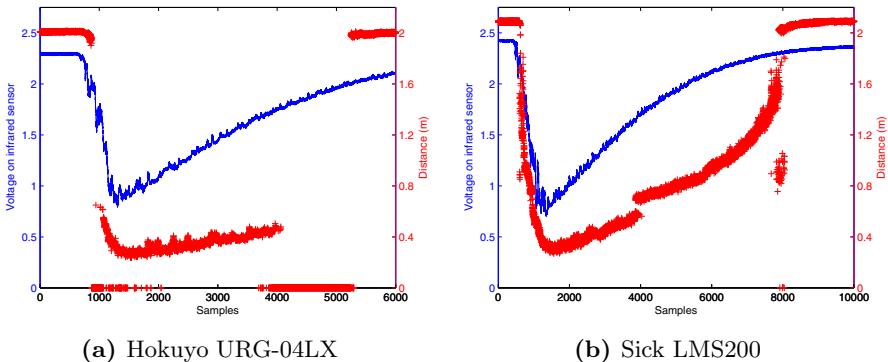


Fig. 2. Laser range finder and infra-red photo sensor readings in a smoky environment from Pascoal et al. [10]

A study on the behavior of four LRFs by Pascoal et al. [10] compared the performance of four LRFs operating under adverse conditions. Two of these are widely used in robotics: the Hokuyo URG-04LX and the Sick LMS200. In one experiment a room was filled with smoke, which slowly dissipates through a small exit. The density of the smoke is measured with an infra-red emitter and a photodetector. A laser range finder is set up two meters from a white MDF surface. Fig. 2a shows the readings of the photodetector and URG-04LX through time².

The readings start without smoke. Without smoke, the URG-04LX returns the correct distance to the MDF surface. When the smoke density gets too high, the voltage on the infra-red sensor drops and the LRF starts sending erroneous readings, shown as points on the x-axis in Fig. 2. The range of erroneous readings is for the Hokuyo quite extensive; for instance in Fig. 2 more than 20% of the samples are actually an error code. The LRF returns short distance readings when the smoke is very thick (between sample 1000 and 4000).

The Sick LMS200 laser range finder exhibits different characteristics, as can be seen in Fig. 2b. As more smoke fills the air, the distance reported drops steadily. There is a density range where the Sick returns error messages too, although this range is much smaller than for the URG-04LX.

Peynot and Scheding [11] published a data set about multi-sensor perception in natural environments with challenging conditions. A multitude of sensors were mounted on a skid-steered vehicle, including four Sick laser range finders and a video camera. Data was collected under different conditions, such as dust clouds, generated by blowing air to dusty soil using a high-power air compressor, and smoke clouds, generated using emergency smoke bombs that worked for about

² This is an analysis performed by us in Matlab on the dataset collected by Pascoal et al. [10]. On request the authors kindly provided us with the dataset of their experiment.



Fig. 3. Snapshot of the data from Peynot et al. [11]. On the left laser range finder readings in a smoky environment. The x-axis is measurement angle, y-axis is time. The darker the color, the shorter the reading is. Red pixels indicate an error code was returned by the laser range finder. The vertical black lines indicate static obstacles. The diagonal black stripes indicate dynamic smoke. On the right the corresponding visual is shown.

one minute, which were carried across the test area by the wind. This results in different smoke densities over time. Various natural environments were used as testing area, containing objects such as trees, houses, tables and carts.

The density of the smoke in this experiment varies through time. Fig. 3 shows the readings of one laser range finder through time³. The robot vehicle was standing still during these measurements. In this figure several static obstacles are visible in front of the vehicle (vertical lines). Additionally, several smoke clouds are visible, drifting from left to right (diagonal stripes). The diagonal stripes are mostly black, indicating that the laser range finder sees the smoke as an obstacle. On the edges of the smoke red lines are visible, indicating that the Sick laser range finder returns an error code. Many error codes are also visible in the lower right where multiple smoke clouds collide, generating turbulence and swiftly fluctuating smoke densities.

This shows that the Sick laser range finder reacts to this smoke similar to a solid object. The readings that result in errors are usually at the edges of a smoke cloud. At the edges of the cloud, the smoke is less dense and turbulent.

From the findings of Peynot et al., combined with the findings of Pascoal et al., we hypothesize that laser range finders react in three distinct ways to smoke, dependent on the density of the smoke.

1. Without smoke or with little smoke, a laser range finder reacts normally; it returns a measurement within the margin of error.
2. When the smoke density is between a lower and an upper bound, the laser range finder returns a failure code.

³ This is an analysis performed by us in Matlab on the dataset '07-StaticSmoke', available from <http://sdi.acfr.usyd.edu.au/datasets/>

3. If the density is higher than this upper bound, the laser range finder reacts on it as if it hits a solid object. It returns a distance which is inversely proportional to the density of the smoke.

3 Experiment

The prior work is not sufficient to quantify all parameters of a LRF smoke model. Missing was an absolute measure of the smoke density. USARSim does not have an infra-red sensor (as used in [10]) or any other sensor to assess the density of the smoke. Instead, the camera image was used to estimate the smoke density, both in the real experiment and the experiment in USARSim. This estimate was based on the color saturation of a reference object visible through the smoke. Unfortunately, those camera images were not available for the work of [10]. In the work of [11] camera images are available, but no reference object is present in the field of view. We did our own experiment to collect data that includes range measurements of the Hokuyo URG-04LX for different smoke densities and uses camera images to describe the density of the smoke. The gathered data is used to confirm the laser range finder model and find the parameters for this model.

Essential to this approach is the ability to control and measure the smoke density. The international fire department training center BOCAS⁴ provided us with facilities to generate smoke in a controlled environment.

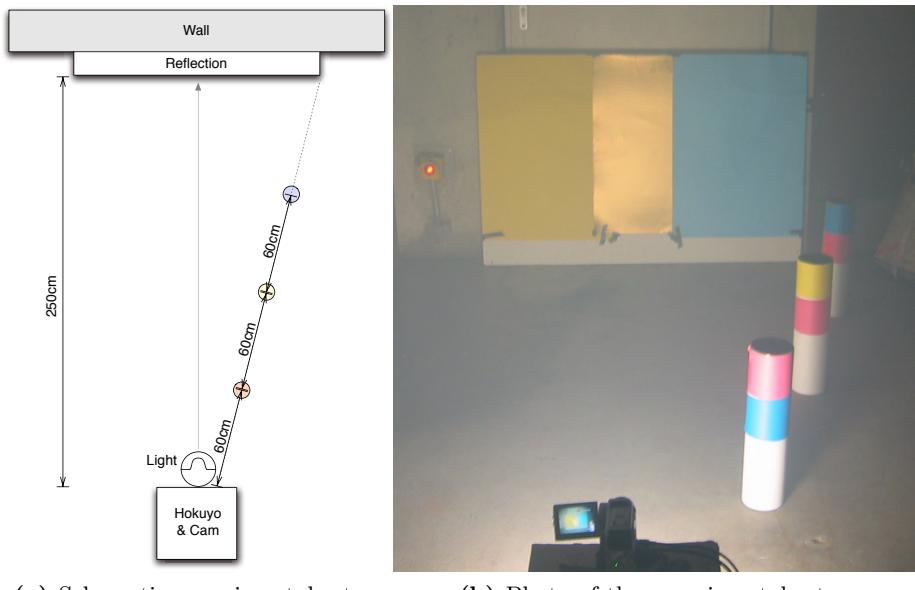
3.1 Setup

The experimental setup was build inside a training room at BOCAS. The room contained a fog machine to generate smoke. This machine produces safe smoke on an oil basis and is also used in theaters.

The Hokuyo URG-04LX range sensor has a maximum range of 4 meters. The sensor was placed at a distance of 2.5m in front of a paper surface. To the right of the paper surface are rusty steel plates, which are used as health shields for the furnace present in the room. Behind the LRF a camera is placed to visually record the smoke density. A 70W fluorescent lamp is placed below the Hokuyo, pointing towards the paper surface to illuminate it. The measurements were synchronized by quickly removing a piece of cardboard that was held in front of the laser range finder and camera.

The camera is used to determine the density of the smoke. The primary method to assess the smoke density is by calculating the color saturation of the paper surface. The paper surface we used consists of blue, yellow and gold parts (see Fig. 4b). Three different colors were chosen to minimize the influence of smoke color. A study by Donate et al. [12] describes the task of reconstructing images of scenes occluded by thick smoke. They observed that an increase in smoke density induces a decrease in both image contrast and color saturation. The saturation level is obtained by converting the RGB camera images

⁴ A description of the facilities of the center can be found at <http://www.boc.eu>



(a) Schematic experimental setup

(b) Photo of the experimental setup

Fig. 4. Experimental setup

into HSV (hue, saturation, value) color space and taking the saturation color channel. The secondary method to measure the smoke density was placing three bollards. These bollards are positioned at 60cm intervals between the camera and the static object (see Fig. 4a). They are standing below the line of sight of the Hokuyo and don't have influence on the range measurements. By counting the number of visible bollards that are visible an alternate density measure can be defined.

This setup is used to measure smoke from two distinct sources. The first kind of smoke is generated by the smoke machine. The smoke has a light grey color and can be produced at will. It is ejected from a small hole from the machine and takes some time to spread through the room, where it dissolves after some time. The second kind of smoke measured was real smoke, coming from burning newspapers, pizza boxes and a Santa Claus doll made from fabric, filled with Styrofoam. These materials were placed at a distance of 1.5m from the LRF, and lit on fire. Unfortunately, the densities of this real smoke did not reach the levels of the artificial smoke.

3.2 Dataset

The laser sensor data is recorded to file using the Hokuyo intervalScanner.exe application⁵. The intervalScanner.exe application captures 100 measurements at intervals of approximately 300 ms. This means we are able to measure at 3 Hz.

⁵ This program is available for download from
http://www.hokuyo-aut.jp/cgi-bin/urg_programs_en/

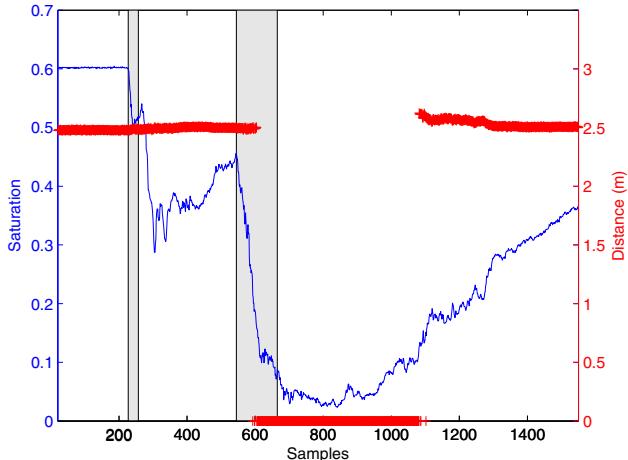


Fig. 5. BOCAS experiment run 1: saturation and laser measurements over time. The grey areas show when the smoke machine was turned on: first for 10 seconds, thereafter for 40 seconds.

Each measurement is placed on a single line and consists of a timestamp and pairs of angles (radians) and distances. The timestamps and measurements are written to a log file.

The dataset consists of five different runs⁶. The first three runs describe the behavior of the Hokuyo for smoke that has been generated by the smoke machine. During the fourth run the laser and camera were pointed at a large gas flame. The last run contains data on smoke from the bonfire described in the previous section. Each run contains a text file with laser range data and a video file.

We have interpreted the recorded data by visualizing it using Matlab. This script is able to handle datasets in different formats, including the format from Pascoal [10], Peynot [11] and our dataset. The laser sensor data is plotted and shows the measured distance for each measured angle. The data is plotted over time to analyze the changes in the sensor measurements caused by smoke. The corresponding camera image for each timestamp is displayed near to sensor data plot.

3.3 Findings

The Hokuyo URG-04LX LRF is affected by both artificial and real smoke. More precisely, the Hokuyo sensor returns correct distance measurements until the density of the smoke reaches a certain density threshold. If the smoke density exceeds this threshold, it returns error values between 0 and 20. From Fig. 5

⁶ The recorded dataset is available on the website
<http://code.google.com/p/usarsim-smoke-fire/>

we can see that this density level⁷ is exceeded when the saturation drops below 0.14. In contrast with our hypothesis, we have not observed the threshold where the URG-04LX returns short readings. See section 6 for a discussion on this.

Pascoal et al. [10] find that the distance reported by the laser is dependent of the surface optical characteristics and “although the output for all sensors does not change significantly with the target colour, there is a noticeable influence according to the specular characteristics of the surfaces”. Our results confirm this statement. Pascoal et al. found a different behavior of the Hokuyo laser range finder when the obstacle was covered with black velvet. We observed unexpected behavior of the Hokuyo URG-04LX for the reflecting gold paper and the rusty steel plates. The distance reported by the LRF to the reflecting gold paper increased with increasing smoke densities, with a maximum difference of 10%. Beyond the maximum the reflecting gold paper is not longer visible, the LRF reports error codes.

4 Implementation

The implementation of the different smoke types and smoke model is done using *UnrealScript*. UnrealScript is the scripting environment of the Unreal Engine. In the Unreal Engine, the *Actor* class is used as the base class for many classes. All functional objects in the game that is not static like walls, is an *Actor*. It provides an interface for default standard variables and functions, like the location and rotation.

USARSim provides a basic laser sensor used by the robots in the simulation. We modified the laser sensor to detect the smoke *Actors* and to behave according the behavior seen in section 2 and section 3.3.

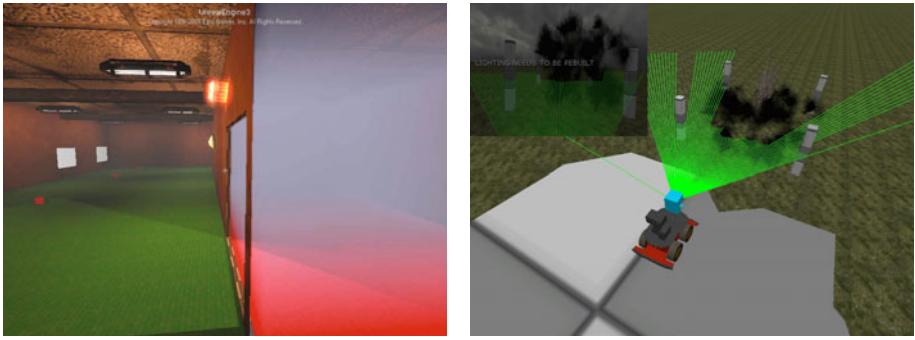
The implementation of the smoke Actors involves the visualization of the smoke and setting up collision boxes that can be detected by the laser sensor, while at the same time the robots can move through the smoke without colliding.

Two different types of smoke are implemented in UnrealScript: smoke areas and local smoke.

4.1 Smoke Areas

The first type of smoke is the *smoke area*, which fills a shape with homogeneously distributed smoke (see Fig. 6a). Such an area is defined by a *SmokeRegional Actor*, which contains a polygon mesh to define its shape. The polygon mesh can take any shape, although it cannot move around. *SmokeRegional Actors* may overlap each other, causing the smoke densities to be added up. The Unreal engine uses the *density* parameter to control the density of smoke. This parameter is in Unreal units and no conversion rule is published to translate into equivalent S.I. units of the real world. Experimentally such conversion rule was found in

⁷ The smoke density is assessed by the saturation of the paper surface (primary method). The secondary method that uses bollards is ignored because the saturation reflects the actual smoke density well.



(a) Smoke area

(b) Local Smoke

Fig. 6. On the left the rooms are filled with smoke areas. On the right a setup with local smoke is shown.

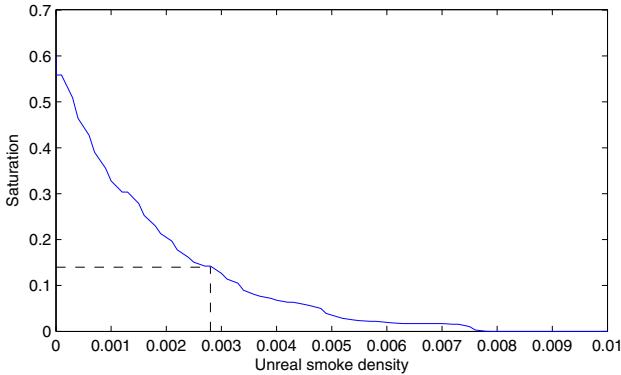


Fig. 7. Smoke density versus saturation in USARSim

section 5. The Kismet Tool in the Unreal Editor can be used to dynamically manipulate the *density* of the smoke. This provides a method for changing the smoke density over time, as demonstrated in Fig. 7.

This type of smoke allows reproduction of the smoke that was created in the BOCAS training room.

4.2 Local Smoke

The second type of smoke is *local smoke*, which is a dynamic smoke column. The column is created by emitting smoke particles from a point source (see Fig. 6b). The smoke consists of many smoke particles with a set lifetime. The color and velocity of each particle change over time to make the smoke look realistic.

The Unreal Editor does not support collision boxes for particles. UnrealScript was used to create a collision box for each particle. The collision box has a fixed size and follows the position of the particle it has been attached to. Creating

a collision box for each particle causes many Actors to be spawned, which can be quite demanding for the engine. The spawning frequency and the lifetime of each particle have to be carefully balanced. Furthermore the exact position of the particle can not be retrieved directly from the Unreal Engine. By using a reasonable estimate the collision box moves along with the particle, although not at the exact same location. This slight divergence between the measurements of two sensors is not unrealistic (as seen from the complex dynamics in Fig. 3).

4.3 Laser Range Finder

The behavior of two laser range finders are implemented. The first is the Hokuyo which behaves as observed in the BOCAS dataset. The other sensor is the Sick LMS200, which behavior was modeled after the prior work [10,11].

A laser range finder returns error values when the smoke density prevents the laser signal to reflect back to the laser range finder. We define the penetration power of a laser as a measure for the distance the laser can travel through a certain smoke density. The Hokuyo laser range finder is just able to penetrate smoke which has a saturation value of 0.14 over 2.5 meters. In section 5, the *density* parameter in the simulation that corresponds to a saturation value of 0.14 is found to be 0.0028.

The penetration power of the Hokuyo laser is defined as:

$$0.0028d \times 2.5m = 0.007dm \quad (1)$$

where d is the unit for density in the simulation.

The original USARSim laser range finder implementation uses the built-in *Trace* function to detect which object is hit. The *Trace* function traces a path to find a collision in that direction. It returns the object that was hit and the hit location.

This behavior is augmented in two ways. The first behavior we implemented is when a *smoke area* was hit. The behavior is based on the model introduced in section 3.3. The laser range finder code has been modified to check if the hit Actor is a smoke Actor. If that is the case a new trace will be started inside the smoke volume, to find the distance the laser beam needs to travel through the smoke.

$$PowerConsumed = density \times distance \quad (2)$$

The *PowerConsumed* is subtracted from the *PenetrationPower*. If the *PenetrationPower* drops below zero the sensor will return error value zero. Otherwise a new trace will be started outside the smoke volume. The sensor may still yield an error value if more smoke is encountered (wherby the penetration power drops below zero) or the maximum range of the sensor is reached.

The second behavior we implemented is for the *local smoke*. The behavior is based on the observations from Fig 2b, i.e. the Sick laser range finder reacts to smoke that has high density similar to encountering a solid object. The Hokuyo LRF exhibits different behavior. As observed in section 3.3, this LRF always

returns error values when smoke gets too dense. The Hokuyo in the simulation returns an error code for each beam that hits *local smoke*.

5 Validation

We have reconstructed the experiment setup inside a USARSim map. The differences between the virtual setup and the physical setup are minimized as far as possible. In Fig. 7, the saturation levels of section 3.1 are calibrated against the smoke densities used by Unreal. The saturation level of 0.14 (the threshold in our smoke model) is equivalent to 0.0028 unreal units.

A virtual BOCAS training room was constructed with an equivalent experiment setup as used in the real experiment. Like the real experiment, the Hokuyo data was recorded to a log file (with a slightly lower frequency of 1 Hz). At the same time, automated screen captures were performed to create a video file, which was used to measure the saturation.

The first experiment is intended to validate the local smoke area model introduced in section 4.1. The room is filled with a puff of smoke, which is allowed to dissolve slowly.



Fig. 8. Reconstructed experiment in USARSim (left), physical experiment (right)

The visualization of the data generated by the physical experiment and the simulated experiment show equivalent behaviors. Both experiments do return error codes when the smoke density becomes too large for the laser to penetrate. The simulation results are smoother than the BOCAS results. This is due to the fact that real smoke is not completely homogeneous. The simulated experiment doesn't model the variance in distance measurements that are caused by variations in the smoke density. This behavior is visible in the physical experiment between samples 800 and 1000.

The second experiment is intended to validate the local smoke introduced in section 4.2. The Sick LMS 200 is in an open area with a number of obstacles in its field of view, equivalent to the experimental setup of Peynot and Scheding [11]. From the left smoke plumes are drifting into view, carried by a slow wind.

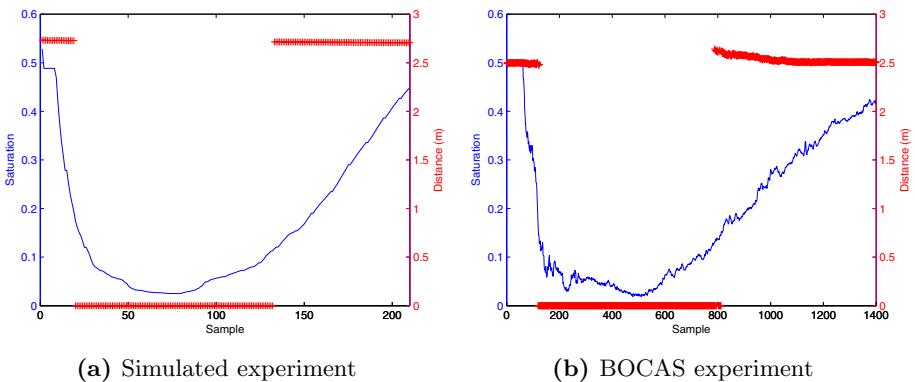


Fig. 9. Saturation versus range measurements for the Hokuyo URG-4LX

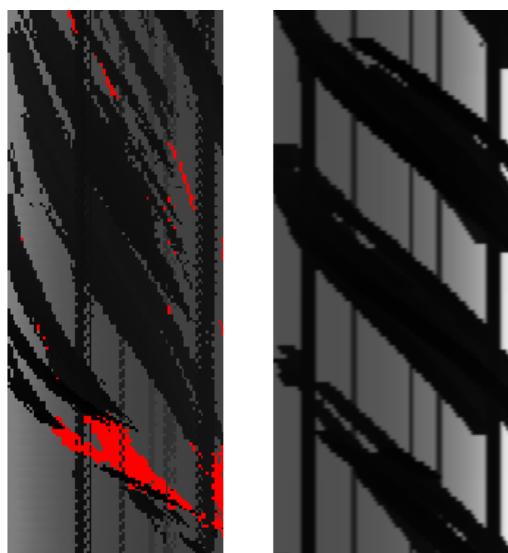


Fig. 10. Saturation and laser measurements as a function time for the Sick LMS200. The vertical axis indicates time, the horizontal axis indicates measurement angle. Left real measurements from [11]. Right a reconstruction in simulation.

In Fig. 10 the recorded laser range finder data is plotted against time⁸. Both for the simulation and for the real measurements the passage of smoke plumes from left to right can be observed. Note that for the Sick LMS200 the smoke is mainly seen as an obstacle (in contrast to the response of the Hokuyo URG-04LX, as described in the previous sections).

⁸ Movies of this experiment are available.

6 Discussion

Our smoke implementation delivers visual smoke which interacts with the Hokuyo and Sick laser range finders. For the Hokuyo URG-04LX a model is implemented which returns error codes when it encounters smoke that has too high density. In our experiments we could not reproduce the behavior of detecting smoke as an obstacle, as observed by Pascoal et al. [10]. Our current hypothesis is that this behavior results from the use of different smoke machines. Alternatively, the smoke density levels in our experiment may not have been sufficiently high to elicit the other behavior.

Our experiment shows that the Hokuyo measurements also depend on the surface optical characteristics. These characteristics have influence on the measured distances and the smoke density that is required to trigger error measurements. Incorporating this knowledge into USARSim would be a great addition. USARSim supports material characteristics such as reflectivity, so implementing more complex behavior is probably feasible.

The experiments of both Pascoal and Peynot [11] showed that the Sick LMS200 in returns the distance to the smoke if the smoke density level is higher than the threshold. This behavior is implemented as the default behavior in the simulation of the Sick laser range finder. At the edges of the smoke cloud, the SICK sometimes returns error values, as can be seen in Fig. 10. We hypothesize that the SICK LMS200 software is unable to deal with the many reflections of the laser beam caused by the dispersed smoke on the edges of the cloud. Additionally, the measured distance of the Sick seems to be a function of the saturation, with a certain penetration into the smoke plume for lower saturation levels. This is not implemented.

7 Conclusion

In this study the behavior of the Hokuyo URG-04LX and Sick LMS200 laser range finders in the presence of smoke are modeled and validated. The validation shows that the resulting behavior of the laser range finders is close to the behavior of laser range finders in the real world. Although there are some disparities between the simulation and the real world readings left, we feel that the level of detail is sufficient and in proportion with the realism of the rest of the simulation environment. This model is for the first time applied in the Virtual Robot competition during the RoboCup Iran Open 2010 competition.

Acknowledgements

We thank fire department trainings center BOCAS for usage of their facilities for our experiment, and in particular Jan Dirk van de Ven for his useful comments and Tom Groenendijk for his time and aid with the experiments. We would also like to thank José Pascoal for kindly sharing their recorded dataset. Finally, our research benefited greatly from the extensive dataset published by the Australian Centre for Field Robotics.

References

1. Murphy, R.R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., Erkmen, A.M.: Search and Rescue Robotics. In: *Handbook of Robotics*, pp. 1151–1173. Springer, Heidelberg (2008)
2. Jacoff, A., Messina, E.: Urban search and rescue robot performance standards: Progress update. In: *Proceedings of the 2007 SPIE Defense and Security Symposium*, vol. 6561 Unmanned Systems Technology IX, p. 65611L (2007)
3. Balakirsky, S., Falco, J., Proctor, F., Velagapudi, P.: USARSim Porting to Unreal Tournament 3. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2009)*, Workshop on Robots, Games, and Research: Success stories in USARSim, pp. 74–79. IEEE, Los Alamitos (2009)
4. Carpin, S., Stoyanov, T., Nevatia, Y.: Quantitative Assessments of USARSim Accuracy. In: Madhavan, R., Messina, E. (eds.) *Proc. of the Performance Metrics for Intelligent Systems (PerMIS) Workshop*, National Institute of Standards and Technology, pp. 111–118 (2006)
5. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: Bridging the Gap between Simulation and Reality in Urban Search and Rescue. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) *RoboCup 2006: Robot Soccer World Cup X*. LNCS (LNAI), vol. 4434, pp. 1–12. Springer, Heidelberg (2007)
6. Schmits, T., Visser, A.: An Omnidirectional Camera Simulation for the USARSim World. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) *CANS 2008*. LNCS (LNAI), vol. 5339, pp. 296–307. Springer, Heidelberg (2009)
7. Pepper, C., Balakirsky, S., Scrapper, C.: Robot Simulation Physics Validation. In: *Proc. of the Performance Metrics for Intelligent Systems (PerMIS) Workshop*, National Institute of Standards and Technology, pp. 97–104 (2007)
8. Balakirsky, S., Carpin, S., Visser, A.: Evaluation of the robocup 2009 virtual robot rescue competition. In: *Proc. of the Performance Metrics for Intelligent Systems (PerMIS) Workshop*, National Institute of Standards and Technology (2009)
9. Boehler, W., Vicent, M.B., Marbs, A.: Investigating Laser Scanner Accuracy. In: *Proceedings of the XIXth International Symposium, CIPA 2003: new perspectives to save cultural heritage. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXIV 5/C15, pp. 696–701 (2003)
10. Pascoal, J., Marques, L., de Almeida, A.T.: Assessment of laser range finders in risky environments. In: *Proceedings of the EURON/IARP International Workshop on Robotics for Risky Interventions and Surveillance of the Environment* (2008)
11. Peynot, T., Scheding, S.: Datasets for the evaluation of multi-sensor perception in natural environments with challenging conditions. In: *Euron GEM Sig Workshop on Good Experimental Methodology in Robotics at RSS* (2009)
12. Donate, A., Ribeiro, E.: Viewing scenes occluded by smoke. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Neffian, A., Meenakshisundaram, G., Pascucci, V., Zara, J., Molineros, J., Theisel, H., Malzbender, T. (eds.) *ISVC 2006*. LNCS, vol. 4292, pp. 750–759. Springer, Heidelberg (2006)

Cooperative Localization Based on Visually Shared Objects

Pedro U. Lima^{1,2}, Pedro Santos¹, Ricardo Oliveira¹, Aamir Ahmad¹, and João Santos¹

¹ Institute for Systems and Robotics, Instituto Superior Técnico, 1049-001 Lisboa, Portugal

² Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911 Leganés, Spain

{pal, psantos, aahmad, jsantos}@isr.ist.utl.pt,
ricardo.oliveira@ist.utl.pt

Abstract. In this paper we describe a cooperative localization algorithm based on a modification of the Monte Carlo Localization algorithm where, when a robot detects it is lost, particles are spread not uniformly in the state space, but rather according to the information on the location of an object whose distance and bearing is measured by the lost robot. The object location is provided by other robots of the same team using explicit (wireless) communication. Results of application of the method to a team of real robots are presented.

1 Introduction and Related Work

Self-localization is one of the most relevant topics of current research in Robotics. Estimation-theoretic approaches to self-localization, as well as to self-localization and mapping (SLAM) have produced significant results in recent years, mainly for single robots, providing effective practical results for different applications. One of the research frontiers in this topic concerns now cooperative localization (and possibly mapping) using a team of multiple robots.

One of the earlier works on cooperative localization [Sanderson, 1996] addresses cooperative localization within a Kalman filter framework, where the relative positions of the robots are the observations of the filtering part of the algorithm, and the state includes the positions of all the robots. Fox et al introduced an extended version of the general Markov Localization algorithm [Fox et al., 2000], where two robots use measurements of their relative distance and bearing to insert an extra step in the belief update algorithm based on the Bayes filter. They used the Monte Carlo Localization (MCL) sampled version of Markov Localization algorithm to influence the weights of the particles of the observed robot from the particles sampling the inter-robot distance and bearing measurement model of the observing robot. Other authors address multi-robot localization using similar approaches, so as to provide relative localization of the team members in one of the team robots local frame from inter-robot distance measurement [Roumeliotis and Bekey, 2002, Zhou and Roumeliotis, 2008]. All these works do not use environment information commonly observed by the team robots to improve their localization.

Other works attempt to take advantage of environment features and landmarks to help a multirobot team to improve the pose estimates of its own team members, while

simultaneously mapping the landmark locations. Fenwick et al [Fenwick et al., 2002] focus on convergence properties and performance gain resulting from the collaboration of the team members on concurrent localization and mapping operations. Jennings et al [Jennings et al., 1999] describe a stereo-vision-based method that uses landmarks whose location is determined by one of the robots to help the other robot determining its location. The first approach addresses a general model that does not take advantage of particular features of the estimation-theoretic methods used (e.g., particle filters) to improve the robustness and to speed up cooperative localization, while the second is focused on a particular application.

In this paper, we introduce a modification of MCL that changes the particle spreading step (used when a robot detects it is lost), using information provided by other robot(s) of the team on the location of an object commonly observed by the lost robot. This modification speeds up the recovery of the lost robot and is robust to perceptual aliases, namely when environments have symmetries, due to the extra information provided by the teammates. The introduced method enables cooperative localization in a multirobot team, using visually shared objects, taking advantage of the specific features of particle filter algorithms. Each robot is assumed to run MCL for its self-localization, and to able to detect when the uncertainty about its localization drops below some threshold. An observation model that enables determining the level of confidence on the ball position estimate is also assumed to be available at each robot of the team. Though these assumptions are, to some extent, stronger than those assumed by cooperative simultaneous localization and mapping methods, they allow global robot and object localization. Though other authors have explored the use of observations to initialize and/or reset particle filters adequately [Lenser and Veloso, 2000, Thrun et al., 2001], the use of shared observations of common objects to cooperatively improve multirobot MCL is novel, to the best of our knowledge.

The paper is organized as follows: in Section 2, we describe our cooperative localization method. Results of experiments with real soccer robots in the RoboCup Middle-Size League (MSL), that use the ball as the visually shared object, are presented in Section 3. Conclusions and prospects for future work are discussed in Section 4.

2 Cooperative Localization Using a Visually Shared Object

Let us consider a team of N robots, r_1, \dots, r_n . Robot r_i has pose (position + orientation) coordinates $\mathbf{l}_{r_i} = (x_{r_i}, y_{r_i}, \theta_{r_i})$ in a global world frame, and estimates them using a MCL algorithm.

Each robot can determine the position of an object o in its local frame, therefore being able to determine its distance and bearing to that object as well. Robots can also determine if they are lost or kidnapped, i.e., if their confidence in the pose estimate drops below some threshold. If a robot is not lost, it can also determine the object position in the global world frame using the transformation between its local frame and the global world frame that results from the knowledge of its pose. The estimate of the object position in any frame is determined based on a probabilistic measurement model that includes the uncertainty about the actual object position. When the global world frame is used, additional uncertainty is caused by the uncertain pose of the observing robot.

The position of the object as determined by robot r_i in the global world frame is denoted by $\mathbf{p}_o^i = (x_o^i, y_o^i)$, while the distance and bearing of the object with respect to the robot, as measured by the robot, are given by d_o^i and ψ_o^i , respectively.

2.1 Overall Description

The original MCL algorithm used by each of the team robots to estimate its pose is as follows:

```

Algorithm MCL(L(t - 1), u(t), z(t), map)
static  $w_{slow}, w_{fast}$ 
 $\bar{\mathbf{L}}(t) = \mathbf{L}(t) = \emptyset$ 
 $w_{avg} = 0$ 
for  $m = 1$  to  $M$  do
     $\mathbf{l}^{[m]}(t) = \text{sample\_motion\_model}(u(t), \mathbf{l}^{[m]}(t - 1))$ 
     $w^{[m]}(t) = \text{measurement\_model}(z(t), \mathbf{l}^{[m]}(t), \text{map})$ 
     $\bar{\mathbf{L}}(t) = \bar{\mathbf{L}}(t) + \langle \mathbf{l}^{[m]}(t), w^{[m]}(t) \rangle$ 
     $w_{avg} = w_{avg} + \frac{1}{M}w(t)^{[m]}$ 
endfor
 $w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$ 
 $w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$ 
for  $m = 1$  to  $M$  do
    with probability  $\max\{0.0, 1 - w_{fast}/w_{slow}\}$  do
        add random pose to  $\mathbf{L}(t)$ 
    else
        draw  $i \in \{1, \dots, M\}$  with probability  $\propto w^{[i]}(t)$ 
        add  $\mathbf{l}^{[i]}(t)$  to  $\mathbf{L}(t)$ 
    endifwith
endfor
return  $\mathbf{L}(t)$ 
```

where $\bar{\mathbf{L}}(t)$ is a set of M particles and their weights at step t of the iteration process, $(\mathbf{l}^{[m]}(t), w^{[m]}(t))$, $m = 1, \dots, M$, $\mathbf{L}(t)$ is a set of M unweighted particles $\mathbf{l}^{[m]}(t)$, $m = 1, \dots, M$, $u(t)$ are odometry readings at time t , $z(t)$ are robot observations at time t , concerning its self-localization, map is a map of the robot world (e.g., a set of landmarks or other), and w_{fast}, w_{slow} are auxiliary particle weight averages, with $0 \leq \alpha_{slow} \ll \alpha_{fast}$, such that w_{slow} provides long-term averages and w_{fast} provides short-term averages. The algorithm uses a **sample_motion_model** and a **measurement_model** to update, at each step, the robot pose, from the odometry $u(t)$ and measurements $z(t)$ information. It keeps adding random particles to those obtained in the re-sampling step (where the probability of cloning an existing particle is proportional to its weight), in a number which increases with the deviation of the w_{fast} average from the long-term w_{short} average. In the limit case, when all particle weights tend to zero in the short-term, all particles are reset according to an uniform distribution.

When cooperative localization in a multirobot team, using visually shared objects, is intended, MCL running in a given robot r_i must be modified so as to use information from other robot(s) in the team, when w_{fast}/w_{slow} drops below a given confidence threshold $C_{threshold}$, meaning that r_i is lost or was kidnapped. That information comes in the form of the object position determined by the other robot(s). Assuming r_i (the lost/kidnapped robot) can observe the same object, the re-sampling is then based on a spatial probability distribution which depends on the distance and bearing of the lost robot to the object and on the uncertainty associated to the object position measurement provided by the other robot(s). This way, while a uniform distribution is still used to keep a certain level of exploration of the pose space to make the algorithm robust to measurement and motion errors, if those errors influence becomes too high, the particles are completely reset according to the cooperative information from teammates about a visually shared object.

The new MCL algorithm used by robot r_i from the team to estimate its pose becomes (the subindex r_i is used for local estimates, odometry readings, observations and particle weights):

```

Algorithm Cooperative_Shared_Object_MCL(Lri(t - 1), uri(t), zri(t), map)
static wslow, wfast
L̄ri(t) = Lri(t) = ∅
wavg = 0
for m = 1 to M do
    l̄ri[m](t) = sample_motion_model(uri(t), l̄ri[m](t - 1))
    wri[m](t) = measurement_model(zri(t), l̄ri[m](t), map)
    L̄ri(t) = L̄ri(t) + ⟨l̄ri[m](t), wri[m](t)⟩
    wavg = wavg + 1/M wri[m](t)
endfor
wslow = wslow + αslow(wavg - wslow)
wfast = wfast + αfast(wavg - wfast)
if wfast/wslow < Cthreshold and info about object position in global world frame
available from teammate(s) rj ≠ ri and object visible to ri then
    draw Lri(t) according to object pose spatial probability distribution determined
from ri and rj information
else
    for m = 1 to M do
        with probability max{0.0, 1 - wfast/wslow} do
            add random pose to Lri(t)
        else
            draw k ∈ {1, ..., M} with probability ∝ wri[k](t)
            add l̄ri[k](t) to Lri(t)
        endwith
    endfor
endif
return Lri(t)

```

Spatial Probability Density

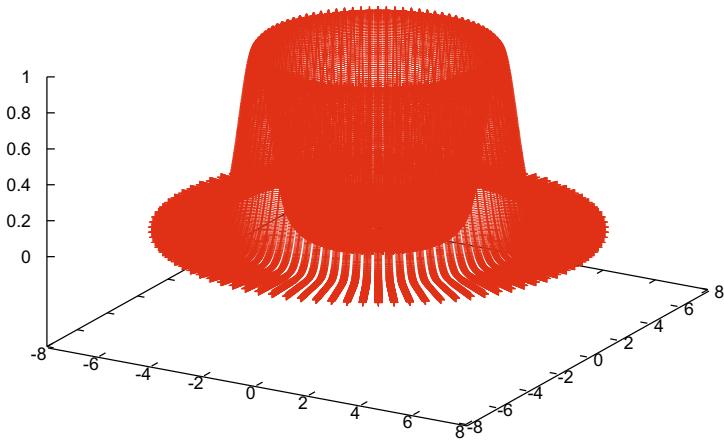


Fig. 1. Typical spatial probability density function from which particles are drawn, after a decision to reset MCL

In the new algorithm one needs to further detail how to handle the following issues:

1. info about object position in global world frame available from teammate(s) $r_j \neq r_i$;
2. draw $\mathbf{L}_{r_i}(t)$ according to object pose spatial probability distribution determined from r_i and r_j information.

Item 1. concerns \mathbf{p}_o^j , i.e., the object position in the global world frame, as determined by r_j (in general, r_j may be any robot but r_i , or several such robots, in which case the object position results from the fusion of their information). Furthermore, we assume that, associated with \mathbf{p}_o^j , r_j provides a confidence measure regarding that information. That confidence measure depends on r_j 's

- object observation model;
- self-localization estimate uncertainty.

Assuming a bivariate Gaussian object observation model for r_j centered on \mathbf{p}_o^j and with a covariance matrix $\Sigma_o^j(d_o^j, \psi_o^j)$ dependent on the distance and bearing to the object, this item contributes to the confidence measure with $|\Sigma_o^j(d_o^j, \psi_o^j)|^{-1}$.

The self-localization estimate confidence factor is not so simple, since one must determine it from the particle filter set. One good approach to this is to consider the *number of effective particles* $n_{eff}^{r_j} = \frac{1}{\sum_{m=1}^M (w_{r_j}^{[m]})^2}$ [Thrun et al., 2005].

Considering both factors, the measure of confidence of r_j on its own estimate of object o position $n_{\mathbf{p}_o^j}$ is given by

$$CF_{\mathbf{p}_o^j} = \eta |\Sigma_o^j(d_o^j, \psi_o^j)|^{-1} n_{eff}^{r_j}$$

where η is a normalization factor.

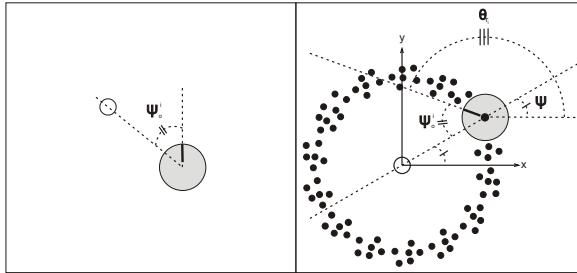


Fig. 2. Computing the orientation of a particle representing a robot pose hypothesis. On the left, bearing of the object with respect to the robot. On the right: relevant angles for the computation of the robot orientation hypothesis for each particle.

Regarding item 2., and assuming that the object is visible to the lost robot r_i , this robot determines the distance and bearing of the object in its local frame, (d_o^i, ψ_o^i) . The distance d_o^i is used to parametrize the spatial probability density function (pdf) from which particles representing the robot position in polar coordinates are drawn, after a decision to reset MCL. This bivariate (using polar coordinates d and ψ centered in the object) pdf is:

- Gaussian in the d variable, with mean value d_o^i and variance inversely proportional to the confidence factor $CF_{p_o^j}$;
- uniform in the ψ variable, in the interval $[0, 2\pi]$ rad.

An example of this pdf is shown in Figure 1.

One can trivially map the polar coordinates onto Cartesian coordinates, thus obtaining the x_{r_i}, y_{r_i} position components of the pose l_{r_i} for robot r_i .

The orientation component θ_{r_i} of l_{r_i} is computed from the bearing angle ψ_o^i of the object, i.e., the angle between r_i longitudinal axis (the one pointing towards its "front") and the line connecting its center with the object center (see Figure 2 - left), and the actual angle ψ of this line with respect to the x -axis of a frame centered on the object, i.e., the particle angle in polar coordinates centered on the object (see Figure 2 - right). We add some random noise θ_{rand} with a zero mean Gaussian pdf representing the bearing measurement error model. Hence, from Figure 2:

$$\theta_{r_i} = \psi + \pi - \psi_o^i + \theta_{rand}.$$

In summary, the **Cooperative_Shared_Object_MCL** algorithm modifies the plain MCL algorithm, replacing the "standard" particle reset, using a spatially uniform pdf, by a particle reset based on the information about the position, in the global world frame (determined by one or more teammates), and the distance and bearing, in the local frame of the lost/kidnapped robot, of an object visible to all the intervening robots. Additional input parameters of this algorithm are (assuming r_i as the lost/kidnapped robot and r_j as any of the robots providing information to improve its localization):

- $C_{threshold}$ to determine if the robot is lost or was kidnapped;
- determinant of the object measurement model covariance matrix $|\Sigma_o^{j}(d_o^j, \psi_o^j)|^{-1}$ — sent by r_j to r_i when r_i detects it is lost and requests support from teammates;
- the *number of effective particles* in $n_{eff}^{r_j}$ MCL algorithm — sent by r_j to r_i when r_i detects it is lost and requests support from teammates;
- distance and bearing of the object in r_i local frame, (d_o^i, ψ_o^i) — measured by r_i when it is lost and receives the above information from teammate(s);
- variance of the zero mean Gaussian pdf representing the bearing measurement error model at r_i (see previous item).

The first two items can be combined first in r_j , that sends to r_i its confidence factor $CF_{p_o^j}$ on the object position in the global world frame.

The regular procedure for each of the team robots is to run **Cooperative_Shared-Object_MCL**. When $w_{fast}/w_{slow} < C_{threshold}$ at r_i , this robot requests help to teammates. One or more teammates r_j send the object position in global world coordinates and the associated confidence factor. Then, r_i particles are spread uniformly over a circle centered on the object, with nominal radius equal to the distance to the object measured by r_i , added to a Gaussian uncertainty around this value, with variance proportional to r_j confidence factor. The orientation component of the pose results from the bearing ψ_o^i of the object measured by r_i in its local frame, including an uncertainty proportional to the variance of the Gaussian representing this measurement model.

A couple of practical issues to be considered are:

- after a robot detects it is lost and spreads its particles over a circle centered with the visually shared object, it should run the regular MCL algorithm in the next steps (a number dependent on the application), so that it does not keep resetting its particles over a circle around the object, while its pose estimate has no converged to the actual value;
- the decision on which teammates can contribute with useful information may be taken by considering their own confidence factor and only using the information provided by robots with $CF_{p_o^j}$ above some given threshold. In general, all teammates can contribute, but in some cases their information may be highly uncertain.

2.2 Particle Spreading Validation

There is a specific situation where the proposed algorithm requires some improvement, e.g., when a robot is kidnapped to a pose where it observes the object at approximately the same distance of where the robot was before. In this case, when the robot detects it is lost by checking its w_{fast}/w_{slow} value, it will still spread new particles over a circle centered with the object, including a region around where the robot wrongly estimates its pose. To prevent such situations, the algorithm must include a restriction that requires all the re-spread particles to be out of a region (e.g., a circle) that includes the majority of the particles at the MCL step when the robot detected it was lost. Nonetheless, it is important to note that particles may be correct in the old pose, if they just have similar positions but fairly different orientations. Because of this, the algorithm must also check if the orientation hypothesis associated to each particle is within a small range of values around the orientation at kidnapping detection time. If they are not, the restriction above does not apply.

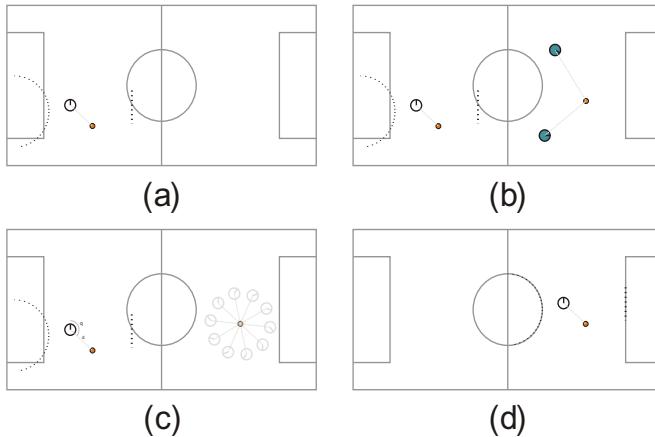


Fig. 3. Cooperative robot localization in RoboCup Soccer MSL: (a) The white robot determines the ball position in its local frame but its estimated pose (based on field line detection - lines observed by robot are dashed and do not coincide at all with the actual solid field lines) is incorrect, because the robot was kidnapped. (b) Teammates (green robots) communicate the ball position in the global world frame, as well as the corresponding confidence factor. (c) Lost robot measures its distance and bearing to the ball and re-spreads the particles according to this and to the ball position team estimate. (d) The previously lost robot regains its correct pose.

3 Results of Implementation in Real Soccer Robots

We have applied the **Cooperative_Shared_Object_MCL** algorithm to real robots in RoboCup Soccer Middle-Size League (MSL), in which the robots use the ball to regain their pose when they are kidnapped and detect to be lost on the field. Figure 3 provides an example that illustrates the algorithm application in this scenario.

3.1 Experimental Setup 1

In this setup tests were made by kidnapping a robot to nine different positions, in one quarter of the whole soccer field, as depicted in Figure 4. The other field regions would not provide extra information, due to the soccer field symmetry. One teammate stopped in a random position always sees the ball and informs the kidnapped robot of the ball's position on the global field frame. Both robots use MCL with 1000 particles, as described in a previous paper [Santos and Lima, 2010]. The standard deviation of the Gaussian used to model the bearing angle measurement error at the kidnapped robot was adjusted experimentally as $\pi/12$ rad.

We kidnapped the robot five times for each of the nine positions. Kidnapping was carried out by picking up the robot and moving it to another location with MCL on and after its convergence to a correct estimate. After kidnapping, the increase of uniformly distributed particles was visible, turning quickly to a re-spread over circle centered with the ball position, as estimated by the teammate.

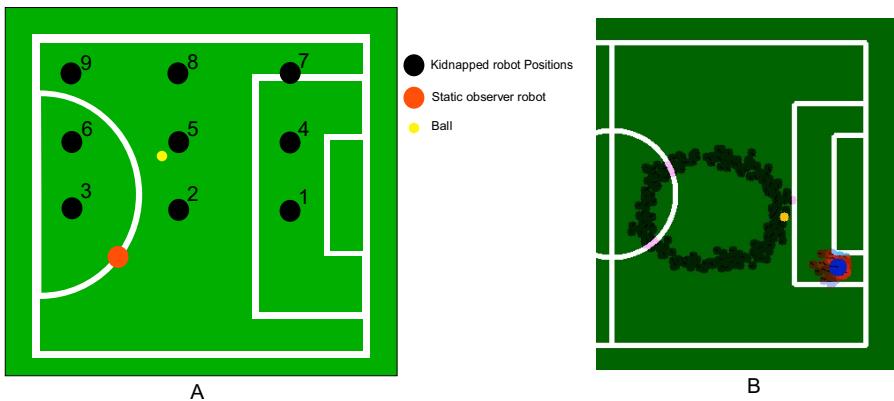


Fig. 4. Layout of experiments. A) The black numbered spots correspond to the positions to where one of the team robots was kidnapped. The red circle represents a static robot, always well localized, and watching the ball (smaller yellow circle). B) The figure in B is an example snapshot of the global frame interface which plots real robot and ball positions as well as particles used by MCL in real-time. Here it shows the kidnapped robot spreading particles after getting lost and using the shared ball.

The algorithm runs on a NEC Versa FS900 laptop with a Centrino 1.6GHz processor and 512Mb of memory, using images provided by a Marlin AVT F033C firewire camera, and in parallel with the other processes used to make the robot play soccer. The camera is mounted as part of an omnidirectional dioptric system, using a fish-eye lens attached to it.

The robots disposed as in position 4 are depicted in Figure 5.



Fig. 5. Real robot example for one of the layout locations: the robot on the left is the robot that informs about the ball position, while the robot on the right is the kidnapped robot, in location 4

3.2 Experimental Setup 2

In this setup, the teammate which observes the ball tracks and follows the ball by maintaining a fixed orientation and distance with respect to ball. We kidnapped the other robot in the following 4 cases during this setup:

- Case 1: Both observer robot and the ball moving when the ball is in the field of view (FOV) of both robots. Robot kidnapped twice in this case.
- Case 2: Observer robot stopped and the ball is moving while the ball is in the FOV of both robots. Robot kidnapped once in this case.
- Case 3: Both observer robot and the ball stopped when the ball is in the FOV of both robots. Robot kidnapped twice in this case.
- Case 4: Both observer robot and the ball moving when the ball moves away from the FOV of the kidnapped robot during the time of kidnapping and then later reappears in its FOV. Robot kidnapped once in this case.

The rest of the details for this setup is similar to experimental setup 1.

3.3 Results and Discussion

Results of experiments in setup 1 are shown in Table 1. In the table, successes correspond to the number of experiments where the robot could regain its correct pose after kidnapping occurred. Iterations to converge refer to the mean value of iterations required by the algorithm to converge after a kidnapping, over 5 experiments for a given kidnapping location. Note that one prediction and one update iteration of MCL take approximately 0.1s each, therefore the mean value of iterations should be multiplied by 0.2 s to have an idea of the time taken by the algorithm in each case. Overall, the algorithm performed quite well in real situations, including cases where we kidnapped the robot to a position where its distance to the ball remained the same. Some field locations are clearly more demanding than others (e.g., 5, 6, 7) causing the robot to fail to regain its posture in one of the tests, possibly due to perceptual aliasing relatively to other field positions.

In the experimental setup 2 the robot successfully recovers and re-localizes itself in 5 situations (3 Cases) and fails in 1. The number of iterations performed by the

Table 1. In experimental setup 1 results of kidnapping a robot to 9 different positions on the field (third column is the average of 5 experiments per location)

Field position	Successes	Iterations to converge
1	5	14.6
2	5	14.8
3	5	7.2
4	5	19.5
5	4	25.5
6	4	10
7	4	15.3
8	5	21
9	5	16

Table 2. Results of kidnapping as explained in experimental setup 2

Case	Situation	Result	Iterations to converge
1	1	Success	17
	2	Success	26
2	1	Failure	18
	3	Success	15
4	2	Success	19
	1	Success	61

algorithm to converge are presented in Table 2. In case 4 of this setup, the robot performs a very high number of iterations to converge mainly due to the absence of the ball from kidnapped robot's FOV for a while before it comes in the FOV of both robots.

In all the sets of experiments, communication delay between the robots was consistently monitored during the run-time of the algorithm. Older data (> 2 seconds) was discarded. A chunk of iterations performed by the robot to converge to the right posture is attributed to this communication delay.

4 Conclusions and Future Work

In this paper we presented a modified MCL algorithm for cooperative localization of robots from a team, where an object visually observed by all the team members involved in the cooperative localization is used. The algorithm takes advantage of the information on the visually shared object, provided by teammates, to modify the particle reset step when a robot determines it is lost (e.g., because it was kidnapped). The algorithm was applied to real robots in RoboCup Soccer MSL with considerable success.

The major issue with our approach is the confusing situation which can arise due to false positive identification of the shared object. A proper approach to solve it would be to use a fused information of the shared object, where the fusion algorithm can discard false positives detected by teammates. Secondly, a fast moving ball creates larger uncertainty about its position which also affects the robustness of our approach to some extent.

Future work will include testing the algorithm in more demanding situation, such as during actual games, with the robots continuously moving. Furthermore, we plan to improve the algorithm by modifying the original MCL such that a fraction of the particles is always spread over a circle algorithm, depending on the ratio between the short-term average and long-term average of their weights, instead of checking when this ratio drops below a given threshold. Other objects, such as the teammates, can also be shared to improve cooperative localization, as long as one can determine their position and track them.

Acknowledgment

This work was supported by project FCT PTDC/EEA-CRO/100692/2008 (author Aamir Ahmad) and also a Banco de Santander Chair of Excellence in Robotics grant from the U. Carlos III de Madrid (author Pedro Lima).

References

- [Fenwick et al., 2002] Fenwick, J.W., Newman, P.M., Leonard, J.J.: Cooperative Concurrent Mapping and Localization. In: Proc. of the IEEE Intl. Conf. on Rob. and Autom. (2002)
- [Fox et al., 2000] Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A Probabilistic Approach to Collaborative Multi-Robot Localization. Autonomous Robots 8(3) (2000)
- [Jennings et al., 1999] Jennings, C., Murray, D., Little, J.: Cooperative Robot Localization with Vision-Based Mapping. In: Proc. of the IEEE Intl. Conf. on Rob. and Autom., vol. 4, pp. 2659–2665 (1999)
- [Lenser and Veloso, 2000] Lenser, S., Veloso, M.: Sensor Resetting Localization for Poorly Modelled Mobile Robots. In: Proc. of the IEEE Intl. Conf. on Rob. and Autom., San Francisco, CA, USA (2000)
- [Roumeliotis and Bekey, 2002] Roumeliotis, S.I., Bekey, G.: Distributed Multirobot Localization. IEEE Transactions on Robotics 18(5), 781–795 (2002)
- [Sanderson, 1996] Sanderson, A.C.: Cooperative Navigation Among Multiple Mobile Robots, pp. 389–400 (1996)
- [Santos and Lima, 2010] Santos, J., Lima, P.: Multi-Robot Cooperative Object Localization — a Decentralized Bayesian Approach. LNCS (LNAI), vol. 2010 (2010)
- [Thrun et al., 2005] Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
- [Thrun et al., 2001] Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust Monte Carlo localization for Mobile Robots. Artificial Intelligence 128(1-2), 99–141 (2001)
- [Zhou and Roumeliotis, 2008] Zhou, X.S., Roumeliotis, S.I.: Robot-to-Robot Relative Pose Estimation from Range Measurements. IEEE Transactions on Robotics 24(5), 1168–1185 (2008)

Parameter Optimization of a Signal-Based Omni-Directional Biped Locomotion Using Evolutionary Strategies

Bařış Gökçe and H. Levent Akın

Boğaziçi University, Department of Computer Engineering, 34342, Bebek, Istanbul, Turkey
{sozbilir,akin}@boun.edu.tr

Abstract. The ultimate goal of RoboCup depends heavily on the advances in the development of humanoid robots. Flexible walking is a crucial part of playing soccer and biped walking has been a very active research topic in RoboCup. In this paper a signal-based omnidirectional walking algorithm for the Aldebaran Nao humanoid robot is presented. Inspired from the existing methods in the literature, the proposed method models the omni-directional motion as the combination of a set of periodic signals. The parameters controlling the characteristics of the signals are encoded into genes and *Evolutionary Strategies* is used to learn an optimal set of parameters.

1 Introduction

Locomotion is one of the main modules for a humanoid robot expected to work in human inhabited environments. Consequently, research on biped walking have become one of the most exciting topics in robotics. Most of the researchers prefer to use precise knowledge about the robot dynamics; mass, location of center of mass, inertia of each link to generate reference trajectory for a stable walking motion. In this approach, the most popular method is to keep the point on the ground where these active forces are zero which is called as Zero-Moment Point(ZMP) into the support polygon [1]. Another approach for biped walking, Passive Dynamic Walking(PDW), is inspired from a bipedal toy that can walk on a slope without any explicit control [2]. This approach is mainly based on gravity and inertia. In this approach, motor control is used to generate the effects of gravity in level floors or shallow up slopes. Recently usage of Central Pattern Generators(CPG) have become popular. The CPG approach is inspired from the symmetric characteristic of vertebrate animals' locomotion[3]. In these animals, locomotion is generated by synchronized rhythms of different gaits. In the application of this idea, a signal is generated for each joint and these signals are synchronized by a central signal. There are mainly two approaches; model-driven and model-free. While the properties of the signals are determined in the model-driven approach, they are left to the training procedure in the model-free approach. Because neither stability nor feasibility is considered in CPG based biped walking, the training procedure is a very crucial part of this type of walking.

In [4], a model for a CPG based biped walking is proposed. In this work, the position of the leg is modeled by three features and motion is defined as a combination

of five movements. The features are calculated with respect to these movements and synchronization is obtained by a central clock. They have applied feedback by using gyroscopes and foot pressure sensors in a later study[5]. While the foot pressure sensors are used to manage the central clock, the gyroscopes are used to reduce the angular velocity by using P-controller. Additionally, policy gradient reinforcement learning and particle swarm optimization is used to optimize the walking speed.

The Standard Platform League (SPL) in RoboCup has changed the platform from 4-legged robot to a humanoid robot *Nao* produced by Aldebaran Robotics [6] in 2008. Since then many studies have been carried on the locomotion module of this robot. In [7], preview control is used to keep the ZMP in the desired path and it requires precise modeling of the robot. On the other hand, Shafi et al. [8] use *Truncated Fourier Series* to generate angular velocities. After the implementation of biped locomotion, *Genetic Algorithm* is used for optimization.

In this work, our main concerns are two important constraints for the locomotion in soccer domain: stability, and the speed to approach a point. For this purpose, we propose an omni-directional model-driven CPG based biped walking algorithm with a training procedure to satisfy these constraints. In the model of locomotion, the position of the leg is represented by two terms; *leg extension* which represents the distance between the foot and the trunk, and *leg angle* representing the angle between the pelvis plate and the line from hip to ankle. In addition to the representation of leg with respect to the trunk, the main motion is divided into four components; shifting, shortening, swinging and loading. Each component is defined as a continuous and periodic signal and the whole motion is generated by summing up these signals. Since the balance and the speed of the bipedal walking algorithm highly depends on the synchronization of the signals, coefficients and constants of the signals are defined as the parameters of the system and the whole system is trained by *Evolutionary Strategies*[9] engine. As a test platform, we used Aldebaran Nao humanoid robot. Firstly, we tested the algorithm in a simulator, Webots and then realized it on real robots.

In the next section, we analyze the proposed biped walking algorithm. Section 3 describes the training procedure for the proposed parametric walking algorithm and Section 4, the experimental results in the simulation and real-world environments. Conclusions and future works are given in Section 5.

2 Proposed Biped Locomotion Approach

In this study, we developed a model for CPG-based omni-directional biped walking algorithm with variable speed. While modeling the system, we use a modified form of the approach given in [4]. The tests on bipedal walking have shown us that changing the angle between the foot plate and pelvis plate causes disturbances on the balance when an unexpected landing of the swinging leg occurs because of the rigidness and existence of sharp edges on the foot. This problem is solved by keeping that angle zero. That way, if the foot touches the ground before or after the expectation, the foot plate will be parallel to the ground and the reaction force will be distributed to not a single point but the whole surface. Therefore in the modeling of the leg we used two features to determine the position and orientation of the foot with respect to the pelvis. These features are

- *Leg Extension:* The distance between the hip joint and the ankle joint. It determines the height of the step while moving.
- *Leg Angle:* The angle between the pelvis plate and the line from the hip to the ankle.

In addition to the model of the leg, we divide the main motion into four components; *shifting*, *shortening*, *loading*, and *swinging*. Each of these corresponds to a component of the motion. They are defined as continuous and periodic signals. Since the algorithm is developed for omni-directional and variable speed walking, in addition to the central clock, the speed values also have effects on the signals. Speed components are forward (v_{fwd}), sideways (v_{strafe}) and turn (v_{turn}).

Central Clock: While modeling such a system, a central clock is required to synchronize the signal. Therefore, a central clock (ϕ_{trunk}) is generated for the trunk which is between $-\pi$ and π . Each leg is fed with different clocks (ϕ_{leg}) with $lid \times \frac{\pi}{2}$ phase shift where lid represents leg ID; -1 for the left leg and +1 for the right leg. This way, the phase difference between legs is preserved as π . In the calculations of the motion features for each leg, the corresponding phase is considered and the features depend on that value.

- **Shifting Motion:** In the shifting motion, lateral shifting of the center of mass is obtained. For this purpose, a sinusoidal signal given in Equation 1 is used. In the equation, a_{shift} determines the shifting amount and it depends on the speed values as shown in Equation 2.

$$\theta_{shift} = lid \times a_{shift} \times \sin(\phi_{leg}) \quad (1)$$

$$a_{shift} = c_{shift} + \frac{c_{shift}^f}{|v_{fwd}|} + \frac{c_{shift}^s}{|v_{strafe}|} + c_{shift}^t \times |v_{turn}| \quad (2)$$

- **Shortening Motion:** The second important motion is the shortening signal. The shortening signal is applied in only a part of the shifting phase. A constraint to prevent both legs from being in the shortening phase at the same time is also added. Therefore, another clock is generated from the clock of the leg with the Equation 3. According to the clock generated for the shortening phase, a part of leg extension is calculated with Equation 4. This phase difference determines the time to shorten the leg.

$$\phi_{short} = \frac{2 \times \pi}{\pi - 2 \times p_{short}} \times (\phi_{leg} + \frac{\pi}{2}) \quad (3)$$

$$\gamma_{short} = \begin{cases} -(c_{short} + \sqrt{v_{fwd}^2 + v_{strafe}^2}) \times (\cos(\phi_{short}) + 1) & \text{if } -\pi \leq \phi_{short} < \pi \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

- **Swinging Motion:** Swinging is the most important part of the motion. In this part, the leg is unloaded, shortened, and moved along the way of motion which reduces the stability of the system considerably. Additionally, we have another constraint for this part. Normally, the swinging signal is always applied. In the double support phase, the weight of the body is shifted through the motion direction, the distance

between the legs for further steps is changed in the single support phase. In the Aldebaran Nao humanoid robot [6], which is our development platform, the yaw joints at the hip of each leg are connected to each other and work in the opposite direction. It is impossible to change the orientation of the body in the double support phase. Because of changing the behaviour of the leg angle feature and the explained additional constraint, two additional clocks are generated; one for the yaw component of the leg angle feature (Equation 5), one for the pitch and roll components (Equation 6). This phase differences determine the time to swing the swinging leg. Therefore, the calculations of the yaw component and other components of leg angle feature are different (Equations 7, 8 and 9).

$$\phi_{swing}^y = \left(\frac{\frac{\pi}{2}}{2 \times p_{swing} - \pi} \right) \times (2 \times p_{swing} + \phi_{leg}) \quad (5)$$

$$\phi_{swing} = c_{swing} \times (\phi_{leg} + \frac{\pi}{2} + p_{swing}) \quad (6)$$

$$\theta_{swing}^y = |v_{turn}| \times (lid \times \cos(\phi_{swing}^y - (sign(v_{turn}) + lid) \times \frac{\pi}{4})) \quad (7)$$

$$\theta_{swing} = \begin{cases} \sin(\phi_{swing}) & \text{if } -\frac{\pi}{2} \leq \phi_{swing} < \frac{\pi}{2} \\ b \times (\phi_{swing} - \frac{\pi}{2}) + 1 & \text{if } \frac{\pi}{2} \leq \phi_{swing} \\ b \times (\phi_{swing} + \frac{\pi}{2} - 1) & \text{otherwise} \end{cases} \quad (8)$$

$$b = -\frac{2}{2 \times \pi \times p_{swing} - \pi} \quad (9)$$

According to the speed and direction of the motion, θ_{swing} is distributed to the components of leg angle features with the Equations 10 and 11.

$$\theta_{swing}^r = -v_{fwd} \times \theta_{swing} \quad (10)$$

$$\theta_{swing}^p = v_{strafe} \times \theta_{swing} \quad (11)$$

- Loading Motion: The last motion of the step is loading which is also not always applied. As in shortening phase, we added the same constraint which prevents both legs from being in the loading phase at the same time. The formula to calculate the clock for the loading phase is given in Equation 12. This phase difference determines the time to load the weight of the robot to the stance leg. This phase affects only the leg extension feature as given in Equation 13.

$$\phi_{load} = \frac{2 \times \pi}{p_{load}} \times \phi_{leg} + \pi; \quad (12)$$

$$\gamma_{load} = \begin{cases} (c_{load} + 0.5 \times (1 - \cos(|v_{fwd}|))) \times \cos(\phi_{load}) + 1 & \text{if } -\pi \leq \phi_{load} < \pi \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

$$\theta_{leg}^r = \theta_{swing}^r + \theta_{shift} \quad (14)$$

$$\theta_{leg}^p = \theta_{swing}^p \quad (15)$$

$$\theta_{leg}^y = \theta_{swing}^y \quad (16)$$

$$\gamma = \gamma_{short} + \gamma_{load} \quad (17)$$

There is a direct relation between the joints of the robot and motion features. In other words, by using these features, the values for hip, knee, and ankle joints can be calculated easily as summarized below:

- There is only one joint that changes the distance between the hip and ankle which is the leg extension. For this reason, the angle of the knee joint is determined according to the value of only the leg extension feature of the motion. From the five sub-motions of a step, the leg extension feature is calculated and it is between -1 and 0. While calculating the angle of the knee joint from the extension, the normalized leg length, n , is calculated as $n = n_{max} + (1 - n_{min}) \times \gamma$. The knee joint value is directly proportional to the arccosine of leg length; $\theta_{knee} = -2 \times \arccos(n)$.
- Another special case for the motion is its turn component. If we consider the physical construction of the robot, there is only one joint which turns the body; the hip yaw joint. Therefore, the turn component of the motion directly determines the value for the hip yaw joint which means $\theta_{hipy} = \theta_{legy}$.
- There are two more joints at the hip. By definition, the roll and pitch components of the leg angle feature directly determines the angles of the hip joints. This means that $\theta_{hipr} = \theta_{legr}$ and $\theta_{hipp} = \theta_{legp}$. In order to compensate the effects of the leg extension on the pitch component of the leg angle feature, half of the knee should be rotated around the turn component of the motion and added to the hip pitch and roll joints. In other words, the pitch and roll components of hip joints can be calculated by the Equation 18.

$$\begin{pmatrix} \theta_{hip}^r \\ \theta_{hip}^p \end{pmatrix} = \begin{pmatrix} \theta_{leg}^r \\ \theta_{leg}^p \end{pmatrix} + R_{\theta_{hip}^y} \times \begin{pmatrix} 0 \\ -0.5 \times \theta_{knee} \end{pmatrix} \quad (18)$$

- Ankle joints are calculated by applying the inverse of the rotation around the turn component of the motion to the negative of the leg angle feature. In order to compensate for the effects of the leg extension on the pitch component of the foot angle feature, half of the knee should also be added to the ankle pitch joint. In other words, the pitch and roll components of ankle joints can be calculated by the Equation 19.

$$\begin{pmatrix} \theta_{ankle}^r \\ \theta_{ankle}^p \end{pmatrix} = \begin{pmatrix} 0 \\ -0.5 \times \theta_{knee} \end{pmatrix} + R_{\theta_{hip}^y}^{-1} \times \begin{pmatrix} -\theta_{leg}^r \\ -\theta_{leg}^p \end{pmatrix} \quad (19)$$

3 Parameter Optimization Using Evolutionary Strategies

After the successful realization of the proposed model, we focus on the stability and speed of the locomotion. We have defined an open-loop model of motion which does not consider stability. Therefore, there is no guarantee for the stability. In addition to the stability of the motion, it is difficult to define the speed by using only the signals. Even if we solve these problems, the effects of the signals on stability and speed may change with the properties of the floor surface like friction, and softness. Therefore, training is an essential part of the locomotion for stability, speed and adaptability. In order to define an optimization on the signals, we use the characteristics of the signals as the parameters of the locomotion and problem is reduced to a parameter optimization problem. As an optimization method, we used *Evolutionary Strategies* [9] because it works on the parameters with real-values. Additionally, the *population size* is an important hyper-parameter in the optimization algorithm which determines the number of individuals existing in a generation. Our model consists of thirteen parameters which determine the walking pattern. These parameters are given in Table 1.

3.1 Fitness Function

The most crucial component of an evolutionary algorithm is the fitness function which determines the goodness of the parameter set. While constructing the fitness function, we analyzed the problem we are trying to optimize. In the soccer domain, gaining possession of the ball is the most important task to be successful in the game. From this point of view, we can conclude that our fitness function should have two components:

- **Reaching target:** The first component evaluates the success of the parameter set in reaching the target object. We measure the distance the robot traveled and divide it by the distance it should have taken. This value should be around 1 if the robot can accomplish the task with success. The importance of this part arises when the robot can not reach the target. The parameter set which can get closer to the target will get more reward from this part.
- **Speed:** The second component considers the speed of the robot. For this purpose, a maximum speed is defined as 20 cm/s and closeness of the obtained speed is used as the second part of the fitness function. That way, slower parameter sets can be punished.

By multiplying these two parts, the fitness value for the corresponding parameter set is calculated. At that point, another important property of the fitness function comes into question. The parameter sets which can accomplish the task should always be better than those which fail. If we analyze our fitness function, it should always be between 0 and 1, but faster parameters sets which cannot reach the target position can have better fitness values than the successful but very slow parameter sets. In order to give reward success, 1 is added to the fitness value if the robot can reach the target. As a result of this correction, while fitness values of successful parameter sets are between 1 and 2, that of failed parameter sets are in between 0 and 1. We can formulate the fitness function as in Equation 20.

Table 1. Parameters of the walking algorithm

Parameter	Description
p_0	Frequency
p_1	c_{shift} (Shifting Constant)
p_2	c_{shift}^f (Shifting Forward Speed Coefficient)
p_3	c_{shift}^s (Shifting Strafe Speed Coefficient)
p_4	p_{load} (Phase Difference of the Loading with the Leg)
p_5	p_{short} (Phase Difference of the Shortening with the Leg)
p_6	c_{short} (Shortening Constant)
p_7	c_{shift}^t (Shifting Turn Speed Coefficient)
p_8	c_{load} (Loading Constant)
p_9	c_{swing} (Swinging Speed)
p_{10}	p_{swing} (Phase Difference of the Shortening with the Swinging)
p_{11}	n_{max} (Maximum Height of the Hip)
p_{12}	n_{min} (Inverse Proportion for the Step Height)

$$\text{Fitness Value} = \begin{cases} \left(\frac{d_{traveled}}{d_{target}} \times \frac{v}{v_{max}} \right) + 1 & \text{if task is accomplished} \\ \frac{d_{traveled}}{d_{target}} \times \frac{v}{v_{max}} & \text{otherwise} \end{cases} \quad (20)$$

In addition to the definition of the fitness function, the way to calculate fitness value is also very important. In order to get rid of noisy calculations, each parameter set is evaluated five times. The resulting fitness value is the average of three calculations by excluding the minimum and the maximum values.

4 Experiments and Results

Since the fitness function defined in the previous section summarizes the expectations from the bipedal walking in the soccer domain, the training procedure requires the robot to move between two targets which are the beacons used in the 2008 set up of the 4-legged league. They are placed at a distance of 450 cm. The important drawback of the beacons is the difficulty of perceiving these objects if the robot is very close to them. Therefore, the robot gets as close as 100cm to the target and turns to go to the other one as the next trial. This gives us a path with length of 250 cm which is sufficient to measure the success of the parameter set.

All processes are handled by two threads on the robots. While one, called *cognition*, is used to perceive the environment and make plans, the other one, called *motion*, is used to run the motion engine with the speed values calculated by the *cognition* thread. For the perception of the environment and low-level decisions(*search_object*, *goto_object* etc.), we used the code developed in our team. Because there is no role in the match to go between two objects continuously, to realize this behaviour we developed a 7-state high level finite state machine, shown in Fig.1.

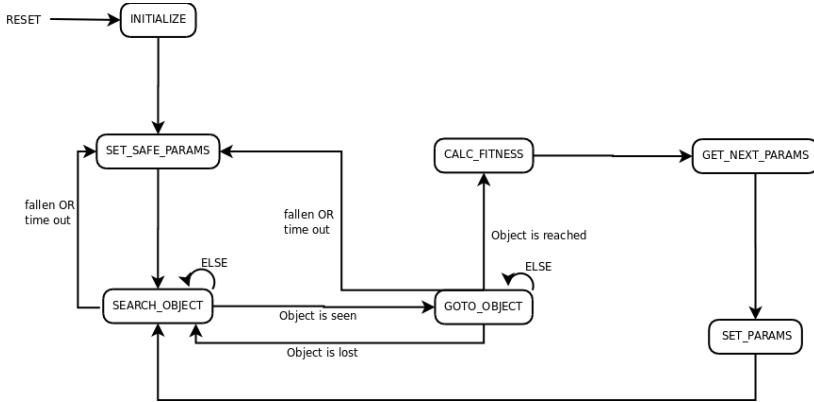


Fig. 1. Finite State Machine(FSM) used as training behaviour

4.1 Experiment Environments

Experiments are conducted in two different environments; simulation and real-world. We aim to obtain two different results with two different experiments. In the first experiment, we aim to conduct a wider search in the domain and we distributed the initial population uniformly in the whole search space. Because the population covers the domain, we should have a bigger population and continue for more generations. Therefore, this experiment is considerably time-consuming. On the other hand, our goal for the second experiment is to fine-tune the parameter set obtained at the end of the first experiment for a different environment in the real-world with a limited time. For this purpose, the initial population is distributed around the base parameter set.

Experiments in the Simulation Environment: As the simulation environment, we used the field constructed by Cyberbotics[10] for the Webots' RoboCup package. There are mainly two important motivations to use this tool. The first one is that it is one of the two supported 3D simulators by RoboCup and the second one is to execute the same binary on the real robot. In the experiments on the simulation environment, we worked on the controllers of two nodes; *Nao* and *Supervisor*. *Supervisor* is the master node which has access to each node in the environment and permission to move the robot from one point to another and change the orientation. In the simulation experiments, this node is used only to correct the posture of the robot when it falls down. *Nao node* is the model of the robot we are using. This node is used to run the *NaoQi* middleware which is developed by Aldebaran and links the shared libraries including our methods. As controller, we have used the one provided by Aldebaran called *nao_in_webots*. The main advantage of this controller is that it lets us to load the same module implemented for the simulation to the real robot.

In the simulation experiment, the population has 20 individuals. Simulation results for the average and the overall best fitness values are given in Fig.2. As a result of the experiment, the best fitness value is increased from 1.23 to 1.29. In order to calculate performance improvement, we should consider the frictional part which determines the

proportion of the average speed to the maximum speed, 20 cm/s. Therefore, the average speed to go to the target is increased from 4.6 cm/s to 5.8 cm/s which can be interpreted as an improvement of more than 25 percent. Although, we have optimized the parameter set for omni-directional walking, we also obtained an improvement for the forward walking which is the most dominant motion to reach the target. While the forward speed of the best parameter set in the initial population is 5.2 cm/s, that of the best parameter set obtained at the end of the training applied for omni-directional walking is 7.1 cm/s which is an improvement more than 35 percent.

Average fitness value of a generation shows us the fitness of the generation to reach the target. When we analyze the average fitness values of the population during the training procedure, it started around 0.1 and increased sharply until the 7th generation. In the later generations, average fitness value started to oscillate and the overall best fitness value did not change. From this point of view, we concluded that the population has converged. This result was important in order to decide when to finish the training in real-world experiment where we have more restricted time limitation.

Experiments in the Real World Environment: In the first experiment, we made a wide search in the domain of the parameter set. However the results obtained from the simulation does not exactly fit to the real world environment. So, we need another experiment in the real world. Because of hardware and time limitations, we aim to fine-tune the parameter set in the real world. For this purpose, we generate initial population in the neighborhood of the parameter set which will be fine-tuned. As the experiment environment, we constructed the same field as the simulation environment with the carpet and beacons used in the 2008 set up of 4-legged league. One important difference with the simulation experiment is that no supervisor is used to correct the posture. Its own '*getUp*' movement is used when the robot falls down. Therefore, time spent to get up is automatically added as a punishment.

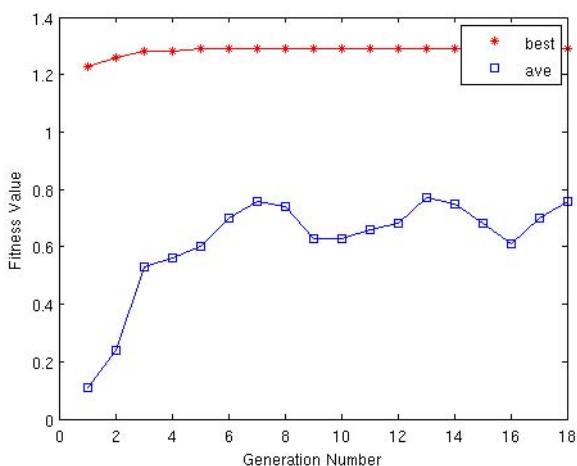


Fig. 2. Results on simulation experiments for the parameter optimization

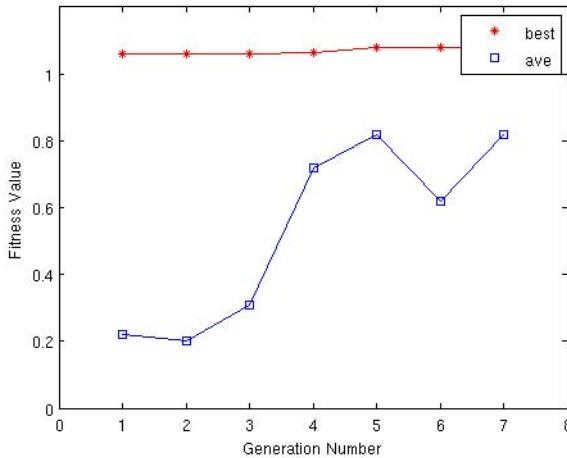


Fig. 3. Results in the real world experiments for the parameter optimization

The results of the experiment for the average and the overall best fitness values are given in Fig.3. In the experiment, we obtained results similar to the simulation experiment with smaller improvements. At the end of the experiments in the real-world, the best fitness value is increased from 1.061 to 1.08045. When we calculate the average speed to go to the target as in the simulation experiments, we find that it is increased from 1.220 cm/s to 1.609 cm/s which is an improvement more than 30 percent. When we analyzed the effects of the fine-tuning on the forward speed, we see an increase from 4.6 cm/s to 6.5 cm/s which is an improvement more than 45 percent.

As we explained in the previous section, the average fitness value of the population shows us the goodness of the current population. Additionally, since fractional parts in the fitness values are very small and changes very slowly in the real-world experiment, the average fitness value can be interpreted as the proportion of the successful parameters in the population. The population has 10 individuals in the real-world experiment. When we analyze the average fitness values of the population during the fine-tuning process, it starts around 0.2 which means only two individuals, where one of them is the hand-tuned one, are classified as successful. For the next generations, the number of successful individuals in the population increases, and eventually most of the individuals can accomplish the task. After the 5th generation, average fitness value became close to the best fitness value and started to oscillate. Therefore, we concluded that the population has converged.

5 Conclusion

In this work, we proposed a model for a CPG-based bipedal walking algorithm. While constructing the model, we started with the representation of the position and orientation of the foot. In this representation, we used two features for the distance and

orientation of the foot with the torso. In the representation of the orientation, we added a constraint to keep the bottom of the foot parallel to the ground surface. This constraint is necessary because of the rigidity of the sharp edges of the sole. By holding the foot parallel to the ground, we distribute the ground reaction force to the whole surface and the stability of the robot is not affected that much when the swinging leg is landing. In addition to the representation of the foot, we divided the motion with four components and represented each component with the synchronous periodical signals. Omni-directional and variable speed movement capabilities were also obtained by embedding the properties of these capabilities into the definitions of these signals.

In addition to the implementation of the bipedal walking algorithm, we applied a parameter optimization technique, namely *Evolutionary Strategies* to optimize the parameter set of the locomotion system. We trained our bipedal walking algorithm in both simulation and real environments with different population sizes. The fitness function is the same for both environments which is the main aim of the locomotion system: approach a target as fast as possible. In the simulation experiment, the initial population is uniformly distributed to cover the parameter domain as much as possible. The optimization procedure for the real world experiment is more like fine-tuning instead of training. In this optimization procedure, the initial population is distributed in the neighborhood of the initial parameter set. Because the experiment is run in a small search space, improvement in the fitness value is not as good as the one in simulation.

Although a stable omni-directional walking is obtained with the proposed model, it is open-loop and vulnerable to the cumulative disturbances on the balance. As a future work, a feedback from foot pressure sensors can be used to increase the stability of the movement. Additionally, the training procedure can be distributed and parallelized. By the help of parallelization, we can reduce the training time and increase the search space.

Acknowledgments

This project was supported by Boğaziçi University Research Fund project 09M105.

References

1. Vukobratovic, M., Juricic, D.: Contribution to the synthesis of biped gait. *IEEE Transactions on Bio-Medical Engineering* 16, 1–6 (1969)
2. McGeer, T.: Passive dynamic walking. *The International Journal of Robotics Research* 9, 62–82 (1990)
3. Pinto, C., Golubitsky, M.: Central pattern generators for bipedal locomotion. *J. Math. Biol.* (2006)
4. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: ICRA, pp. 1597–1603 (2006)
5. Faber, F., Behnke, S.: Stochastic optimization of bipedal walking using gyro feedback and phase resetting. In: Proceedings of the International Conference on Humanoid Robots (Humanoids) (2007)
6. Aldebaran Robotics, <http://www.aldebaran-robotics.com/eng/>

7. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. *Robotics and Autonomous Systems* 57, 839–845 (2009)
8. Shafii, N., Seyed Javadi, M.H., Kimiaghalam, B.: A truncated fourier series with genetic algorithm for the control of biped locomotion. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (2009)
9. Beyer, H.-G.: *The Theory of Evolution Strategies*. Springer, Heidelberg (2001)
10. Cyberbotics (2010), <http://www.cyberbotics.com/>

Designing Effective Humanoid Soccer Goalies

Marcell Missura, Tobias Wilken, and Sven Behnke

University of Bonn,
Computer Science VI, Autonomous Intelligent Systems,
Roemerstr. 164, 53117 Bonn, Germany
{missura,wilken,behnke}@cs.uni-bonn.de
<http://ais.uni-bonn.de>

Abstract. Most of the research related to the topic of falling strategies considers falling to be an unavoidable part of bipedal walking and is focused on developing strategies to avoid falls and to minimize mechanical damage. We take an alternative point of view and regard falling as a means to an end. We present our falling strategy for the specific case of a robot soccer goalie that deliberately jumps in front of a moving ball to prevent it from rolling into the goal. The jump decision is based on observed ball position, speed and direction of movement. We show how we implement a targeted falling into the appropriate direction, minimize the time from the jump decision to ground impact, and what solutions we developed to prevent mechanical damage. The presented falling technique was used in RoboCup Humanoid KidSize and TeenSize competitions and proved to be essential for winning.

Keywords: humanoid robots, robot soccer, falling, motion generation, mechanical design.

1 Introduction

Falling is an inevitable part of bipedal walking, especially in dynamic environments such as robot soccer games. However, falling can also be intentional. In highly dynamic sports players frequently decide to take risky actions that result in falling to the ground. Two soccer-related examples are kicking from an unstable position to attempt to move the ball towards the opponent goal even at the cost of falling, or jumping in front of the ball to prevent it from reaching the own goal. Unlike accidental falls, intentional falls do not strike the player by surprise. The location and time of the ground impact can be estimated more precisely and the player has more time to prepare a fall sequence to minimize the risk of damaging the body.

In this paper, we present our falling strategy for the specific case of a robot soccer goalie that deliberately jumps in front of a moving ball to prevent it from rolling into the goal. The jump decision is based on observed ball position, speed and direction of movement. We show how we designed a diving motion that does not damage the robot's body. The fall is accelerated to reach the ground as soon as possible and targeted to the left or right side depending on where we expect

to catch the ball. The proposed method was used in the German Open 2009 and RoboCup 2009 competitions, where they proved to be essential for winning, for example the TeenSize Dribble & Kick competition in Graz 2009.

The remainder of the paper is organized as follows. In Section II we present a review of related work. In Section III our simple, trainable algorithm is presented that handles the jump decision. Section IV describes how we developed the diving motion for the goalie and explains the motion itself in detail. In Section V we show our mechanical solutions for damage avoidance and finally, in Section VI we present experimental results from RoboCup competitions.

2 Related Work

While the research of humanoid bipedal walking has been a hot topic for decades, the issue of falling has barely been addressed. Some work exists on the detection of situations that would lead to a fall [1-3] and on avoiding a fall altogether by stopping [4], squatting for a short while [3], or stepping into a capture region [5]. Results addressing actual falling strategies when the ground impact is inevitable, are scarce. The research groups of Fujiwara et al. [7-12] and Ogata et al. [13-14] have proposed UKEMI techniques that distribute the impact force over a sequence of designated impact points [10]. Forward and backward falling motions were optimized using inverted pendulum based models in [11-12]. Yun et al. [6] presented a different approach of changing the fall direction using targeted stepping and inertia shaping to avoid hitting an obstacle. In the closest related work, Ruiz del Solar et al. [15] also came to the conclusion of distributing the impact force onto multiple contact points and proposed a methodology to design fall sequences that can also be used for intentional falling. To the best of our knowledge, no research up to date is published on intentional falling, although it is applied by several teams in the KidSize league of the RoboCup humanoid soccer competitions.

3 The Jump Decision

The first thing the goalie does when it enters the field, or after getting up from the floor, is to use global localization on the soccer field to position itself on the center of the goal line looking into the direction of the opponent goal. When the position of the goalie is good enough, the robot stops, bends its knees to lower the CoM, and observes the field in this special goalie halt position that is lower than the halting posture of the field players. In order to protect the goal, the goalie tracks the ball and estimates the position and the velocity of the ball in egocentric coordinates that are used to calculate when and in which direction the goalie should jump. Currently, our robots are equipped with a 3 camera vision system. This is subject to change due to a rule update in the RoboCup Humanoid Soccer league that prohibits the use of more than two cameras. Our vision system processes approximately 25 frames per second distributed over 3 cameras. In the exceptional case of the goalie halt position, all but one cameras are switched

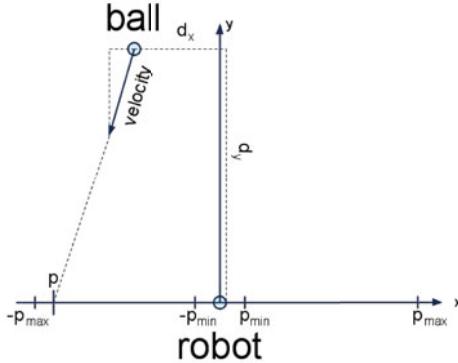


Fig. 1. A sketch of the egocentric view of the goalie and estimated contact point p of the ball with the lateral plane x . The goalie will only jump if the ball is expected to hit the goal between the goal posts (p_{max}) and with some distance to the feet of the goalie (p_{min}).

off. This way the available 25 frames per second are concentrated on one camera only to boost the precision of the velocity estimations. The position of the ball is measured by inverting the ground projection onto the camera plane. The velocity of the ball is estimated by averaging the change of position in three consecutive frames.

Figure 1 illustrates the egocentric view of the goalie. We denote the lateral direction as x . Negative values are to the left and positive values to the right of the robot. The sagittal direction was labeled y with positive values to the front. The jump decision of the goalie is based on the estimated time and point of contact of the ball with the lateral plane.

The one-dimensional contact point p on the lateral plane is given by:

$$p = d_x - d_y \frac{v_x}{v_y}, \quad (1)$$

with d_x and d_y being the distance of the ball in lateral and sagittal direction relative to the goalie and (v_x, v_y) is the velocity of the ball. Only a certain range of contact points are interesting because they are located inside the goal area. Assuming the goalie is standing in the middle of the goal and $|p| > p_{max}$, the ball will entirely miss the goal on the left or the right side. On the other hand, if the contact point is too close to the goalie ($|p| < p_{min}$), the ball will most likely bounce off of the feet of the goalie. In this case instead of a diving motion the goalie squats down quickly holding its arms to the sides of its feet in order to achieve a larger blocking surface. We set p_{max} to half of the width of the goal and p_{min} to half of the width of the robot.

Apart from the point of contact, the time of contact is also an important determinant of the jump decision. If the ball is slow and takes long enough to reach the goal for the goalie to have enough time to position itself between the

ball and the goal, there is no need to jump. The diving motion is associated with high costs, such as a certain time needed for the goalie to get back on its feet and to reposition itself in the goal. During this time the goalie is incapacitated and the goal is unprotected. In addition, the risk of mechanical damage is always present. We only want the goalie to perform the diving motion when it is necessary to block the ball. To determine the time of contact is a more difficult task than to calculate the contact point, because the ball does not travel with a constant velocity. In fact, the friction with the ground slows the ball down at a rate which is highly location dependent on the floor material and possibly uneven ground. Additionally, in the KidSize and the TeenSize leagues balls of different size and weight are used. Because an analytic calculation would be complex and possibly misleading, we developed a very simple trainable model. We collect training data by placing the goalie in the goal and do not allow it to move. Then we kick the ball towards the goal from various distances and random velocities. After each kick we indicate with a joystick whether the goalie should have jumped or not and label a whole series of observations. After we collected enough data, we reduce the state space to two dimensions by discarding the x components of the ball and velocity observations and use libsvm [16] to train a support vector machine to separate the two classes. Figure 2 shows an example. The blue colored region shows the cases where the goalie should not jump, the red region indicates the class where the goalie should jump. The training data is colored according to the manually assigned labels. It is noticeable that red and blue observations overlap slightly. This is because for the border cases even for humans it is difficult to decide if the goalie should have jumped or not. Furthermore it can be seen that starting from approximately two meters, the noise in the position and velocity estimations increases significantly with distance. The black line marks the border between the two classes. A nice property of libsvm is that it provides a probability estimate of the certainty that an observation belongs to a class. By default we separate

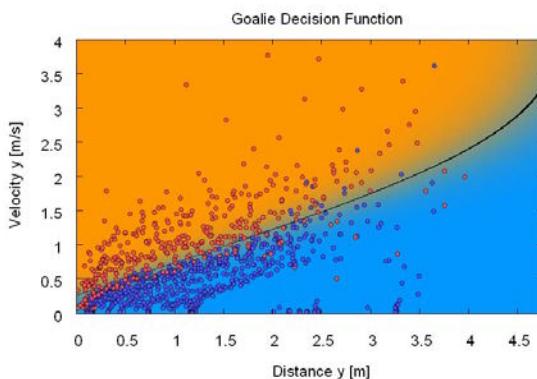


Fig. 2. Classification of ball observations into two categories: the goalie should jump (red) and the goalie should not jump (blue). The black line marks the border between the two classes, which is fine tunable with one parameter.

the classes where an observation belongs to each class with a probability of 0.5. Later on this threshold can be altered to manually fine tune the “sensitivity” of the goalie with a single parameter in case the goalie occasionally lets a ball pass or behaves too jumpy.

4 The Goalie Motion

The diving motion of the goalie is comparable to an inverted pendulum. If started in a perfectly upright position at 90° , neither the pendulum, nor the goalie will fall. The more the starting angle of the pendulum deviates from the vertical, the faster it reaches the ground.

Excluding any kind of active pulling towards the ground, we can regard a free falling point mass dropped from the height of the robot as the lower bound of possible falling times. The time the free falling point mass takes to reach the ground from a height of 0.6 m and an initial velocity of zero can be calculated using Newton’s law.

$$h(t) = h_0 - \frac{1}{2} g t^2 \quad (2)$$

Setting $h(t) = 0$ m, $h_0 = 0.6$ m, $g = 9.81$ m/s², and solving for t we obtain a falling time of approximately 0.35 seconds. We are using an inverted pendulum as a benchmark that approximates the size of the robot’s body (0.6 m), starting from a small angle of 0.1 radians. We determined the falling time of the pendulum numerically with Euler’s method using

$$\ddot{\phi}(t + \Delta t) = -\frac{g}{l} \cos \phi(t) \quad (3)$$

$$h(t) = l \sin \phi(t) \quad (4)$$

and setting $\dot{\phi}(0) = 0$, $\phi(0) = \pi/2 - 0.1$, $l = 0.6$ m and $g = 9.81$ m/s². ϕ describes the angle of the pendulum with $\phi = \pi/2$ being the vertical and angles are increasing counter clockwise. We used a very small time step of $\Delta t = 10^{-6}$ s. According to the simulation it takes 0.87 seconds for our benchmark pendulum to reach a height of zero. When inspecting the height of the falling pendulum, as shown in Figure 3, one can clearly see that the extra time is needed mainly for the first degrees of falling. Once the pendulum passes an angle of 45° , which is equivalent to a height of approximately 0.4 m, the remaining fall is almost as fast as free falling. Apart from avoiding damage, the challenge in designing a diving motion for the goalie is to accelerate the fall with an initialization motion, comparable to starting an inverted pendulum at angles deviating from the vertical as much as possible.

The diving motion of our goalie is driven by a short sequence of key frames as depicted in Figure 4. Due to the symmetry of our robots, the same motion can be applied for jumps to the left or the right by mirroring the key frames respectively. First, in key frame a) the goalie stands motionless in the goal and observes the field in a special goalie halt position that is bent more in the knees

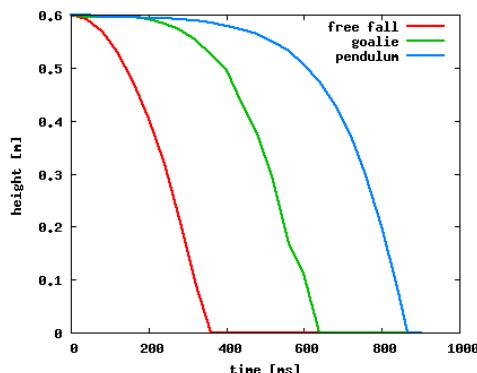


Fig. 3. Time series of the fall height of the goalie motion (middle) compared to a free falling body (left) and an inverted pendulum started at an angle of 0.1 rad (right)

than the posture of the field players. When it is time to jump, the motion starts in key frame b) with a sideways hip swing using the hip and the ankle joints to accelerate the torso towards the yielding leg. We named the leg towards which the robot is falling the yielding leg, the other one the support leg. A few moments later the yielding leg starts to shorten in key frame c), while the feet are constantly rolling to support the fall of the robot to the side. Before the support leg loses ground contact, it is extended and moved inwards closer to the yielding leg in key frame d). We found that the resulting rotation of the torso reduces the time until the ground impact, while the sudden leg extension increases the blocking distance. Simultaneously, the arms are lifted up high above the head with the maximum speed the servos allow. The arms reach their target position just in time before they touch the ground in key frame e), so that when the goalie lands, it can touch the goal posts with its arms. Finally, when lying on the floor, the goalie performs a “sweeping” motion with the yielding leg to clear a ball that may have been blocked and is now lying dangerously close to the goal. When performing the diving motion, the special goalie halt

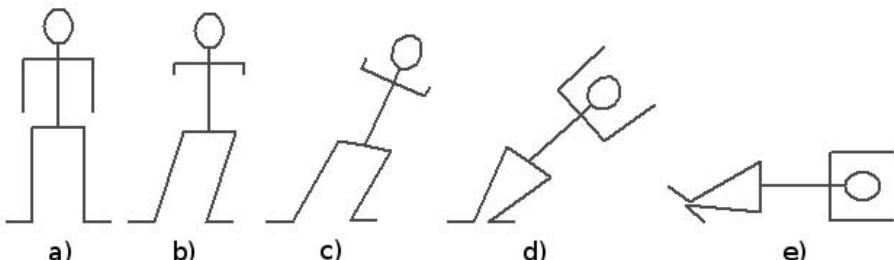


Fig. 4. The key frames of the goalie motion

position plays a key role. As mentioned before, the posture of the goalie is lower than the default standing position of our field players for two reasons. First, the CoM is lower and it takes less time for it to reach the ground. And second, the bent knees of the robot give room for a longer, more powerful push with the support leg. Earlier versions of our goalie started in full upright position and were significantly slower. Furthermore, the arms of the goalie are already raised a bit when standing in the halt position. This is important for the arms to reach their target position above the head in time. The shoulder servos are not fast enough to complete the full 180 degrees from the bottom to the top in the short time available. If the arms are not up at the time of the ground impact, the goalie may be injured by the fall and the blocking distance would also be shorter since the arms would not reach as far when the robot is down on the floor.

Our diving motion takes 0.64 seconds from the jump decision to ground contact. We have determined this time by recording the goalie motion several times with a digital camera and counting the frames from when the goalie started to move until the robot obviously touched the ground. Multiplying the frame rate of the camera (12 fps) with the average number of frames, we obtain the total duration of the motion. Figure 3 shows the results as the height of a reference point in all three experiments depending on time. The reference point of the goalie is on the top of the head. The height of this reference point was determined by counting the pixels in the frames of a video from a base line up to the top of the head and converting them into centimeters. Please note that these numbers were obtained by experiments with our KidSize robot. The actual diving times of our TeenSize robot may differ slightly.

We developed the goalie motion in a simulation with a three dimensional physical model of our KidSize robot. Only after we achieved promising results in the simulation we made the first attempt on a real robot. The motion worked close to the expectations without damaging the hardware, so we continued to optimize the motion in a real environment. When the motion was first applied to a real robot after it was developed in simulation, it was significantly slower with 0.8 s until ground contact. By making small changes to the keyframes manually and testing it on the real robot we have been able to improve the timing. With the current duration of 0.64 s it is equivalent of starting an inverted pendulum of 0.6 m length at approximately 14 degrees (0.25 rad).

5 Preventing Mechanical Damage

Although the fall of the goalie is intentional, preventing mechanical damage is still a crucial aspect. The intention of the fall, in contrast to accidental falling, makes it easier to prepare a damage minimizing pose in time before the ground impact. Earlier versions of our goalie motion turned the robot's torso towards the ground and used the arms to soften the impact of the fall, as can be seen in Figure 5. Our KidSize robots now have less sensitive shoulder joints after a degree of freedom in lateral direction was removed. We found that raising the elastic arms above the head is sufficient to avoid damage to the shoulder joints

and designed a new motion without the more time-consuming torso twist. The first point of impact occurs on the upper arm as can be seen in Figure 6. The arms are somewhat flexible and are padded with soft guards, which cushion the impact and also protect the head that contains the cameras. With only one degree of freedom in the shoulder in sagittal direction, the ground forces are orthogonal to the rotational plane of the servo and the gears are not stressed by the impact. Nevertheless, triggered by the last keyframe of the motion shortly before the impact, all joints of the robot are completely relaxed to effectively protect the gears of the servos, the most sensitive parts when it comes to falling. Except for the joints, the robot is well protected against falls, since the aluminum frame is sturdy enough to withstand the ground impact from a low height such as the size of the KidSize models.

As already mentioned, our TeenSize model Dynaped was performing the same goalie motion. Dynaped is significantly larger and heavier than our KidSize robots. In addition to relaxing the joints shortly before the impact, we also designed special mechanical precautions to avoid damage to the frame of the robot, as shown in Figure 7. Dynaped's elastic arms are completely padded with a thick layer of foam and have no joints apart from the sagittal joint in the shoulder. The shoulder joint itself is attached to the torso with a flexible connection. Six struts made from hard rubber hold the shoulder in place and yield even to weak forces. A rigid connection would likely bend or break sooner or

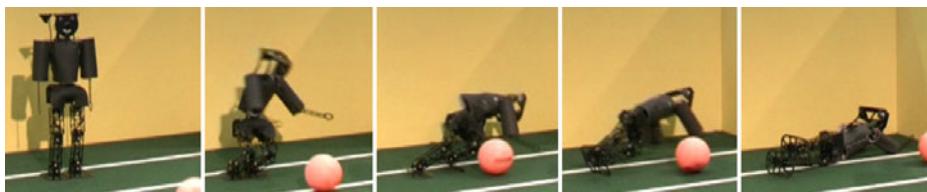


Fig. 5. TeenSize 2007 robot Bodo protects the goal by twisting its torso and leaning on its arms to land the fall

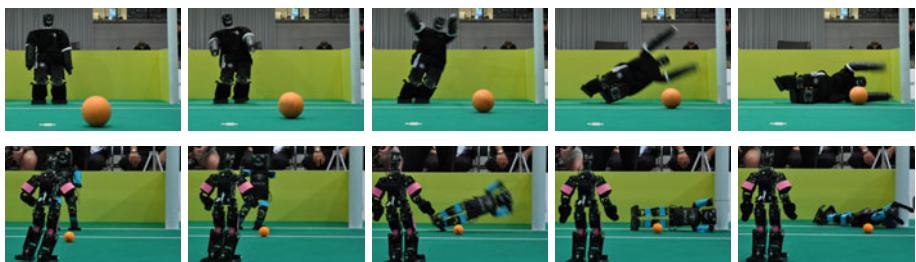


Fig. 6. TeenSize robot Dynaped (top row) and KidSize robot Ariane (bottom row) take a dive for the ball during RoboCup 2009 in Graz. The attacker in the bottom row was a penalty striker from team CIT Brains.

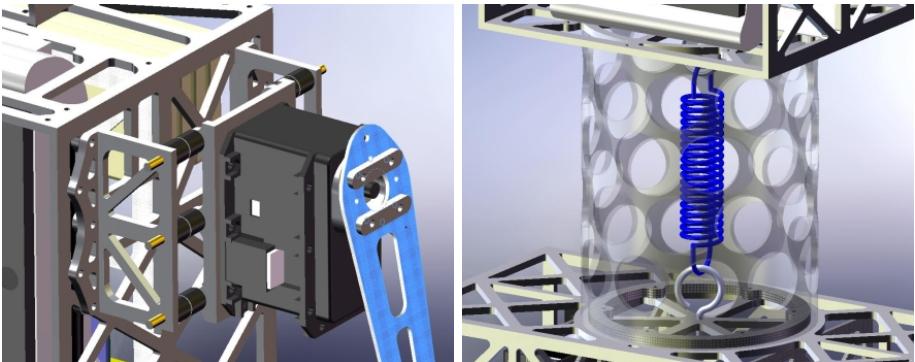


Fig. 7. TeenSize robot Dynaped's mechanical precautions against damage resulting from falls. The shoulder (left) is fixed to the frame by six flexible struts made from hard rubber. The hip (right) is a pull linkage with a spring that holds the torso in place.

later when the goalie repeatedly hits the floor with its arm. Furthermore, the torso of the robot is attached to the hip with a pull linkage that is equipped with a strong spring. The right part of Figure 7 shows this hip construction. The spring loaded linkage works much like a mechanical fuse. The trunk cylinder shown semi transparently is not mechanically fastened to the hip. It is loosely positioned on a ring that prevents the cylinder from sliding sideways. The blue spring holds the cylinder firmly in place, but it yields to strong-enough forces. We are using a spring with a spring constant of 1.753 N/mm expanded by 36.6 mm in the base position. This results in a minimum force of 64 N exerted by the spring at all times that holds the torso firmly in place. Using the screws at both ends of the spring the initial tension can be changed easily. We found the appropriate setting by manually adjusting the screws until the spring force was high enough to keep the torso from shaking during walking.

Figure 8 sketches an analysis of the ground impact. The expected point of attack of the ground reaction force is the shoulder joint. The height of the torso from the base of the cylinder up to the middle of the shoulder joint is 249.25 mm , the total width of the torso at the shoulder joint is 310 mm , and the cylinder has a diameter of 100 mm . When pushing the torso by the shoulder joint, the cylinder creates a fulcrum on the hip 50 mm away from the center. The spring has an angle of attack of approximately 21.8° and the ground reaction force has an angle of attack of approximately 129.44° . Taking the initial spring load of 64 N and the leverage effect into account, this results in a force of approximately 12.85 N that is needed to start extending the spring and moving the torso out of its place. This calculation neglects the effects of the arm padding and the yield of the rubber struts in the shoulder joint. Since the initial force is relatively small, we expect that the torso is moved out of place each time the goalie falls to the ground. After the impact forces are absorbed, the spring pulls the torso back and the cylinder snaps back onto the ring. If the forces are too strong or

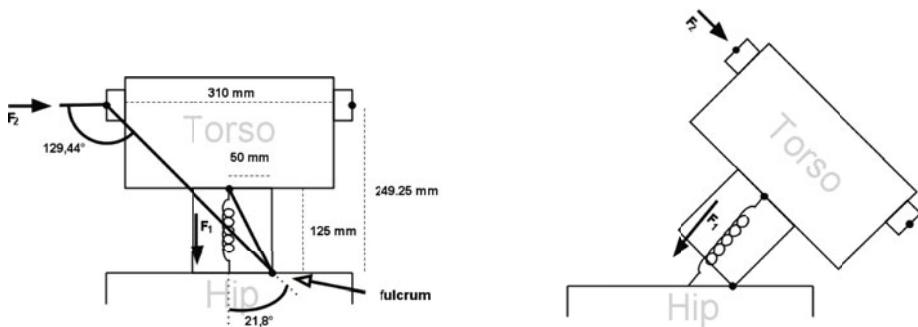


Fig. 8. Analysis of the effect of the ground reaction force. The cylinder creates a fulcrum on the hip frame and the torso works as a lever against the spring. At the shoulder a joint a force of 12.8 N (F_2) is needed to expand the spring pulling at the center of the torso with an initial force of 64 N (F_1).

attack the torso at unexpected points and angles, it can happen that the cylinder completely detaches from the ring and the torso “dangles” off the hip. In this case, the spring is unable to pull the torso back and it has to be put back in place manually.

6 Experimental Results

Even though the soccer fields in the Humanoid Soccer league still provide structured and well defined environments, the real performance of the robots during the competitions can be regarded as a much more realistic benchmark than lab experiments. Our team NimbRo has a successful history in the Humanoid Soccer league and this can partially be accounted to the performance of our goalies. Our TeenSize goalie Bodo performed well in Suzhou in the year 2008 and successfully defended the goal several times with an earlier version of our goalie motion, as can be seen in Figure 5. We have applied our improved goalie motion as presented in this paper to our KidSize and TeenSize models and both were performing well in the German Open 2009 in Hannover and the RoboCup 2009 in Graz, where they successfully blocked several balls and prevented the opponent team from scoring. In particular in the TeenSize Dribble & Kick finals this was decisive for winning the game 2:0 against the reliable striker of the Japanese Team CIT Brains.

7 Conclusions

We have presented our solution for a robot soccer goalie motion, which is an intentional fall in a targeted direction. The intention of the fall makes it possible to execute the motion every time the same way and to prepare a pose that

reliably directs the impact of the ground forces into mechanical precautions. The diving motion was developed in a simulation first and then optimized on real hardware. We reached a falling duration of approximately 0.64 seconds with our KidSize robots. The diving motion is triggered by a decision based on two factors: the point of contact and the time of contact of the ball with the lateral plane relative to the goalie. To avoid incapacitation and possible damage, the goalie only dives if it is absolutely necessary to block the ball. Our strategy to avoid damage consists mostly of mechanical solutions. The main protection of the KidSize goalie is a padded upper arm, while the goalie motion makes sure that this is the place of the first ground contact. Our TeenSize robot is protected by a flexible shoulder joint and a pull linkage in the hip held in place by a strong spring. Additionally, for both robots the joints are relaxed shortly before ground impact. None of our soccer goalies sustained any damage during games in the past years.

In future work we are planning to further automate the training process of the goalie decision and to investigate possibilities how a robot can learn to optimize the diving motion.

Acknowledgments

Funding for the project is provided by Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under grants BE 2556/2-2,/4.

References

1. Baltes, J., McGrath, S., Anderson, J.: The Use of Gyroscope Feedback in the Control of the Walking Gaits for a Small Humanoid Robot. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 628–635. Springer, Heidelberg (2005)
2. Hhn, O., Gacnik, J., Gerth, W.: Detection and Classification of Posture Instabilities of Bipedal Robots. In: Proceedings of the 8th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines - CLAWAR 2005, London, England, pp. 409–416 (2005)
3. Renner, R., Behnke, S.: Instability Detection and Fall Avoidance for a Humanoid Using Attitude Sensors and Reflexes. In: Proc. 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 2967–2973 (2006)
4. Morisawa, M., Kajita, S., Harada, K., Fujiwara, K., Fumio Kanehiro, K.K., Hirukawa, H.: Emergency Stop Algorithm for Walking Humanoid Robots. In: Proc. 2005 IEEE/RSJ Int. Conf. in Intelligent Robots and Systems, pp. 2109–2115 (2005)
5. Pratt, J., Carff, J., Drakunov, S., Goswami, A.: Capture Point: A Step toward Humanoid Push Recovery. In: Proc. 2006 IEEE-RAS Int. Conf. on Humanoid Robots, pp. 200–207 (2006)
6. Yun, S.-K., Goswami, A., Sakagami, Y.: Safe Fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping. In: ICRA 2009 (2009)
7. Fujiwara, K., Kanehiro, F., Kajita, S., Yokoi, K., Saito, H., Harada, K., Kaneko, K., Hirukawa, H.: The first human-size humanoid that can fall over safely and stand-up again. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 1920–1916 (2003)

8. Fujiwara, K., Kanehiro, F., Saito, H., Kajita, S., Harada, K., Hirukawa, H.: Falling motion control of a humanoid robot trained by virtual supplementary tests. In: IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 1077–1082 (2004)
9. Fujiwara, K., Kanehiro, F., Kajita, S., Hirukawa, H.: Safe knee landing of a human-size humanoid robot while falling forward. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 503–508 (2004)
10. Fujiwara, K., Kanehiro, F., Kajita, S., Kaneko, K., Yokoi, K., Hirukawa, H.: UKEMI: Falling motion control to minimize damage to biped humanoid robot. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 2521–2526 (2002)
11. Fujiwara, K., Kajita, S., Harada, K., Kaneko, K., Morisawa, M., Kanehiro, F., Nakaoka, S., Harada, S., Hirukawa, H.: Towards an optimal falling motion for a humanoid robot. In: Proc. of the Int. Conf. on Humanoid Robots (Humanoids), pp. 524–529 (2006)
12. Fujiwara, K., Kajita, S., Harada, K., Kaneko, K., Morisawa, M., Kanehiro, F., Nakaoka, S., Harada, S., Hirukawa, H.: An optimal planning of falling motions of a humanoid robot. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 524–529 (2007)
13. Ogata, K., Terada, K., Kuniyoshi, Y.: Real-time selection and generation of fall damage reduction actions for humanoid robots. In: Proc. of the Int. Conf. on Humanoid Robots (Humanoids), pp. 233–238 (2008)
14. Ogata, K., Terada, K., Kuniyoshi, Y.: Falling motion control for humanoid robots while walking. In: IEEE-RAS 7th Int. Conf. on Humanoid Robots (Humanoids), Pittsburgh (2007)
15. Ruiz del Solar, J., Palma Amestoy, R., Marchant, R., Parra-Tsunekawa, I., Zegers, P.: Learning to fall: Designing low damage fall sequences for humanoid soccer robots. In: Robotics and Autonomous Systems, pp. 796–807 (2009)
16. Chang, C., Lin, C.: LIBSVM: a library for support vector machines (2001), <http://www.csie.ntu.edu.tw/~cjlin/libsvm>

A Supporter Behavior for Soccer Playing Humanoid Robots

Karen Petersen, Georg Stoll, and Oskar von Stryk

Simulation, Systems Optimization and Robotics Group,
Department of Computer Science,
Technische Universität Darmstadt

Abstract. In this paper, an efficient behavior for a humanoid soccer robot supporting a teammate manipulating the ball is proposed and evaluated. This approach has been derived from human soccer tactics, requires only minimal information about the current state of the game and is therefore well suited for robots with directed vision and limited motion abilities like humanoid robots. The position of the supporter is chosen in a way to reduce the chances of the opponent team for scoring a goal and to increase the chances of the robot to take over the role of its teammate manipulating ball in case the teammate falls down or is being dodged by the opponent team. By different parameterizations more defensive or more offensive player tactics can be realized as well as adaptations to specific tactics and skills of the opposing robot players. The developed behavior was evaluated in a simulation based statistical analysis. Also an evaluation in games during RoboCup 2009 is given.

1 Introduction

In the RoboCup humanoid kid size league, teams of 3 two-legged robots play soccer against each other in a continuous, dynamic game. Within a team, usually one robot is the goalkeeper and two robots are field players. The field player which is in a better position for playing the ball is the striker, the other field player is called supporter. The robots can change roles at any time, if appropriate. This paper deals with the question of how the supporter should behave in order to contribute to the team's success. The goal is, that the supporter is able to continue an attack, if the striker fails, without obstructing the striker in any way. Particularly, the supporter should not approach the striker too close or cross the line between the ball and the opponent goal, to prevent blocking the striker's goal shots.

Compared to other RoboCup leagues, the implementation of tactical behavior faces specific difficulties. A two-legged walking robot has far more limited mobility than a wheeled system, like in the small size or mid size leagues. Further, the camera field of view is limited and can only view the area in front or sideways of the robot by moving its head. This leads to specific, larger

uncertainties in world modeling than with omnidirectional or overhead camera systems like those being used in the mid or small size leagues.

In this paper, a supporter behavior which accounts for the specific locomotion and vision abilities of humanoid robots is derived from human soccer tactics. The proposed behavior strengthens the defense by covering the own goal, but enables the supporter to continue an attack if the striker fails at the same time. The behavior requires only minimal information about the current state of the game: the teammate positions are required to negotiate roles. For the actual supporter positioning only the current ball position and sufficiently good self localization are required. The implementation consists of four states. The only required basic ability is approaching a position while avoiding obstacles. The developed behavior is minimalistic with respect to required computing power and motion and sensing capabilities, and is therefore very well suited for humanoid robots with directed vision.

The remainder of this paper is structured as follows: Related work is discussed in Sect. 2. In Sect. 3 first the basics of human soccer tactics are introduced, afterwards the possibilities for transferring the tactics to humanoid robot soccer are discussed. The developed behavior is presented in Sect. 4. In Sect. 5 the experiments in simulation and with real robots are described and the results are discussed. A conclusion is given in Sect. 6.

2 Related Work

Until now, most effort in the humanoid kid size league was put into the development of basic skills like locomotion, localization and perception, as well as the behavior of the striker, which usually approaches the ball and kicks or dribbles it towards the opponent goal. In most matches, it appears like the second field player does not add a significant advantage to the team in many game situations. Often, the supporter is either placed at a fixed position on the playing field, or it executes the same behavior as the striker, which leads to both field players approaching the ball and, in the worst case, obstructing each other.

A basic strategy for advantageous positioning of the second field player for humanoid robots is presented in [1], where the supporter always attempts to stay between the ball and the own goal. For the AIBO robots, in [3] the supporting robot is positioned close to the attacker to either recover a lost ball, or to be able to receive a pass. Similarly, in [7] the robots currently not controlling the ball are positioned either close to the striker, to support the attack, or at positions where they are likely to find the ball after it was out of bounds. In [4], in a simulation based on mid size robots, one robot is always positioned close to the striker, to act as backup. In [6], supporter strategies for the NAO robot are described; here a distinction is made between an offensive behavior that follows the striker, and a defensive behavior that supports the goalkeeper in defending the goal.

3 Transferring Human Soccer Tactics to Humanoid Robots

3.1 Human Soccer Tactics and Strategy

There are three important aspects in human soccer team play: formations, the style of play, and tactics.

Formations are used to define the distribution of the players on the field. A specific formation is described as $a : b : c$, where a denotes the number of defenders, b the number of midfielders, and c the number of strikers. Frequently applied formations are $4 : 4 : 2$ (4 defenders, 4 midfielders, and 2 strikers), and $4 : 3 : 3$ (4 defenders, 3 midfielders, and 3 strikers). Besides zone defense it is the purpose of the formation to provide depth and width in defense and attack. Width means having players to the left and to the right of the player currently playing the ball, to allow the team to move forwards quickly in attack and to keep the opponent from playing around the defender in defense. Depth means having a player behind the striker or defender currently playing the ball, in case the first defender gets dodged in defense or to have a play-to for a back pass if the forward player is stopped in his attack.

The style of play determines how the players act during play. Widespread styles of play are the counter-attacking style, the possession style and the direct soccer style. With the counter-attacking style the team waits for the opponent close to the own goal and tries to gain an opportunity of a fast counter attack. A team playing possession style soccer tries to keep possession of the ball for a long time through lots of passings until a good opportunity for an attack comes along. In direct soccer style, the ball is moved or passed towards the opponent goal as fast as possible, trying to score a goal with as little ball touchings as possible. This describes best what currently soccer playing humanoid robots do.

Tactics describe conceptual actions depending on the specific situation of the match. In soccer the basic tactical decision is to play offensive or defensive. Applying tactics depends on an overall understanding of the conceptional playing situation, including the current score, the opponents abilities and the position and movement of the ball, the teammates and also the opponent players.

3.2 Transfer to Humanoid Robots

As long as there are only two field players and one goalie in a match, it does not seem meaningful to apply formations to humanoid soccer robots. Different styles of play are also still difficult to achieve, because current humanoid robots are not able to control the ball (dribbling, passing) in a way needed to realize these styles. For the same reason it is difficult to distinguish between offensive and defensive situations, because almost never one team is really in control of the ball. Usually, if players from both teams are close to the ball, both are similarly able to kick the ball. As robot capabilities in these fields are getting better, applying tactics and styles of play will be a topic of future work.

From the suggested behavior for individual players not possessing the ball, it can be derived, that, no matter if the team is attacking or defending, it is

important to place supporting players next to and behind the striker, to achieve width and depth in attack or defense. Because the passing abilities of humanoid robots are not yet reliable, width in attack and defense is considered to be more important. Therefore a good position for the supporter is diagonally behind the striker. In humanoid robot soccer this position has another advantage: When assuming that the ball is somewhere close to the striker, the supporter has a high probability of seeing the ball from that position. If the striker itself does not see the ball, and if localization confidence of both, striker and supporter, is above a given threshold, the communicated ball position is used as measurement update for the striker's Kalman filter. In that way, if the ball is occluded from the striker's point of view by an opponent or by the striker's own arms, the ball model of the striker gets updated with information coming from the supporter.

The best position of the supporter has to be identified with respect to the current situation of the match. Variables determining the current situation are the positions of the opponent players, the positions of the teammates, and the position of the ball. Perception capabilities of the robots are very restricted, therefore the players are not able to perceive the situation in the same way as a human does. Currently, opponents are only modeled as obstacles that have to be avoided. Further information about the opponents is not available, and therefore cannot be considered for behavior planning. It was chosen to position the supporter relative to the position of the ball, instead of relative to the position of the striker, for several reasons:

1. Just as opponents, teammates are observed only as obstacles, but additionally, each robot communicates its current position to its teammates. With this information, the supporter is able to calculate its relative position to the striker, but with two sources of error: the position error of the supporter's self localization and the position error of the striker's self localization. In the worst case, these errors sum up when calculating the relative position from supporter to striker. This error can be significantly large, and therefore positioning relative to the striker is not reliable.
2. The ball position is sensed directly by the supporter and therefore can be considered reliable enough. It can be assumed that the own striker is closer to the ball than the supporter, otherwise they would switch roles. Further, opponents can be assumed to approach the ball from the other side.
3. If the supporter positions relative to the striker, and the way between the striker and the ball is blocked (e. g. by an opponent), none of the robots will reach the ball, until the striker finds a way. If, however, the supporter positions relative to the ball, it potentially gets closer to the ball while the striker's way is blocked, and the robots can switch roles. In that way, the attack can be performed faster.

It is therefore reasonable to position the supporter diagonally behind the current position of the ball, instead of behind the striker.

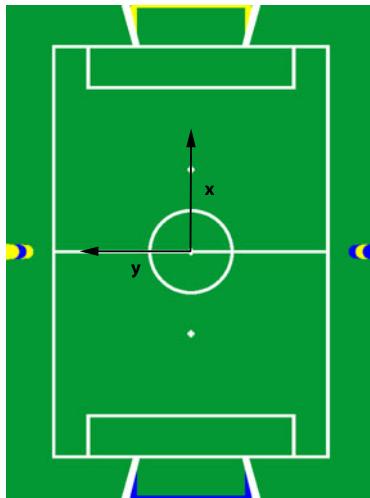


Fig. 1. The playing field with coordinate system for the team defending the blue goal

4 The New Playing Supporter Behavior

Apparently, different strategies of the opponent team cannot be overcome with the same behavior, e. g. a team known to never kick the ball farther than 1m can be handled differently than a team being able to score goals from any point on the field. Therefore, an important issue when designing the new behavior was flexibility, to allow adaptation to different opponent strategies and capabilities.

The best position for the supporter has to be identified with respect to the current situation of the match. All positions in this section refer to a world coordinate system with its origin in the middle of the center circle, the x -axis pointing towards the opponent goal, and the y -axis parallel to the centerline, as it is depicted in Fig. 1. As described in Sect. 3.2, it is reasonable to position the supporter diagonally behind the ball. This position can be described, relative to the position of the ball, with an offset sideways (y -direction) and backwards (x -direction). These two parameters can be controlled to achieve different variations of the behavior. For example, a larger y -offset is appropriate if the opponent is known to frequently play diagonal passes, while a smaller y -offset gives a higher probability of blocking direct goal shots. Likewise, a small x -offset is suitable if the opponent only kicks the ball slightly, whereas a larger x -offset prevents that both, striker and supporter, can get dodged simultaneously.

The sideways shift should not always be fixed left or right to the ball, rather the supporter should be positioned on the side of the ball that is farther away from the closest touchline, and therefore is closer to the middle of the field. At this position the supporter has a high probability of blocking opponent goal shots. To avoid frequently switching sides when the ball is close to the middle, a hysteresis is used that causes the robot to stay on the current side. Overall,

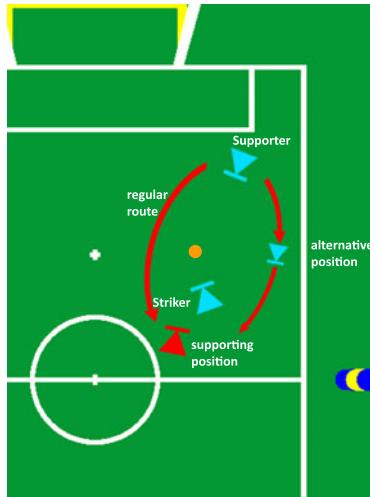


Fig. 2. Regular and alternative way to the desired supporting position

as shown in Sect. 5, this positioning leads to a high likelihood of the supporter standing between the ball and the own goal, and therefore being able to block opponent goal kicks. Additionally, if the striker fails (e. g. because of falling down, or the opponent dodges the striker), the supporter is likely to be in a good position to resume to the role of striker and scoring a goal. This is also shown in Sect. 5.

The special case when the supporter is closer to the opponent goal than the ball should be handled separately, in order to avoid the supporter crossing the line between the ball and the opponent goal on the way back, and therefore potentially blocking goal kicks of the teammate, as in Fig. 2. One possibility to handle this situation would be a distracting potential between the ball and the goal, and let the supporter walk along the potential field. However, this requires a high computation effort. A very simple, yet effective alternative is, to first send the supporter to an alternative position next to the ball, on the same side as the robot is currently positioned. The decision whether to select the left or right alternative position is based on the supporter's current position relative to the line between the current ball position and the middle of the opponent goal, as it can be seen in Fig. 3. The behavior control checks the signum of the angle $\gamma \in (-\pi/2; +\pi/2)$ between the vector from the ball to the supporter and the vector from the ball to the opponent goal. If $\gamma \geq 0$ the supporter is positioned to the left of the ball, else to the right.

Tactical decisions can be realized with two parameters x_{min} and x_{max} limiting the supporters longitudinal position. For example, for defensive behavior, the supporter should always stay in the own half of the playing field. This can be obtained by limiting the forward motion of the supporter to the middle line, i.e. setting $x_{max} = 0$. Conversely, to achieve an offensive behavior, the backward

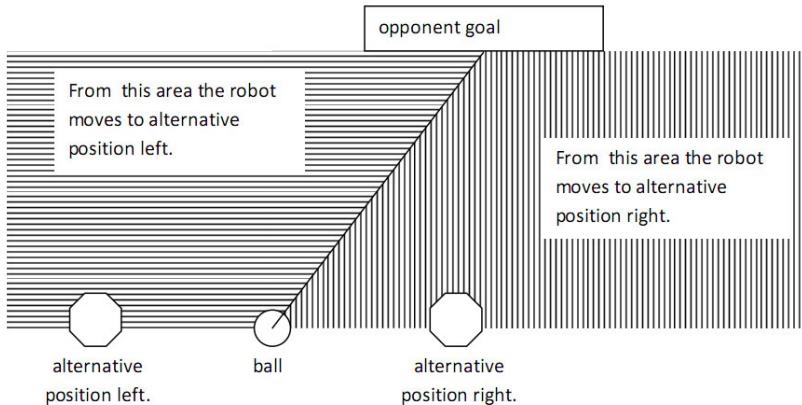


Fig. 3. Separation of walking left or right of the ball

motion of the supporter can be limited to the middle line by setting $x_{min} = 0$, which leads to the supporter always being positioned near the opponent goal. Furthermore, the longitudinal limits can be used to forbid the supporter to enter the goal areas, to avoid committing an illegal defense or illegal attack foul.

In summary, the developed behavior can be controlled by four parameters, namely the offset of the supporter position sideways (y -offset) and backwards (x -offset) to the ball, and the minimum and maximum limit x_{min} and x_{max} of the supporter position in the longitudinal axis. The behavior has been implemented as a XABSL [5] option, consisting of four states:

- Search Ball: Initially, and whenever the current ball position is unknown, the robot tries to find the ball.
- Position to Support Ball: In the most general case, the supporter goes to the position diagonally behind the ball, based on the current parameter settings.
- Position to Support near Middle: If the ball is near the longitudinal middle of the field, the behavior enters the hysteresis state to avoid frequently switching sides.
- Position to Support Alternative: If the supporter is closer to the opponent goal than the ball, it first goes to the alternative position next to the ball.

A visualization of the state machine, including the possible state transitions, is depicted in Fig. 4.

5 Results

The developed behavior has been evaluated in a simulation, because ground truth data of the robots' and ball's position is available. Furthermore, the videos of

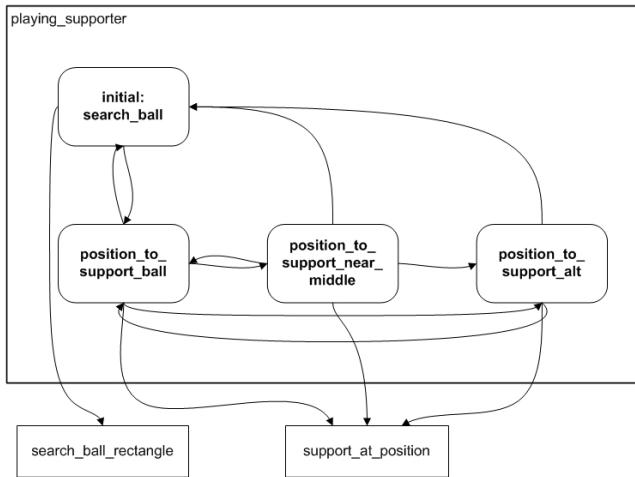


Fig. 4. The state machine for the supporter behavior

the relevant matches of RoboCup 2009 have been analyzed manually, to show the benefits of the new approach in a real situation.

For the experiments, a simulation based on [2] has been used. A screenshot of the simulation can be seen in Fig. 5. 44 halftimes of 10 minutes each have been simulated with one team playing with the new behavior against a team playing with the old behavior. For comparison, also 43 halftimes were simulated with both teams relying on the old behavior. The old behavior always positioned the supporter at a fixed distance in front of the own goal, away from the striker in y -direction. It can be assumed, that this behavior is not very effective. For the robots' motion simulation a kinematic motion algorithm was used, due to the limitations of computational power available. Therefore, no robot could fall over, which usually occurs frequently in real matches. However, the typical walking pattern was simulated, resulting in realistic shaking of the robots' cameras. The simulated camera framerate was 10 fps, also due to computing power. For a fluent game, independent of human influences, a script was used to replace the ball after it was kicked into the goal or out of bounds. During simulation, the position of the ball and of all robots was logged every second. Furthermore, the scored goals with associated timestamps were logged. Based on this data, it was possible to analyze different aspects that were influenced by the improved behavior. Over all halftimes for the reference data with both teams playing with the old behavior, the mean of the goal difference is 0.1395, while the mean for a team with new behavior against a team with old behavior is 1.7727. An unpaired t-test gives a P-value of 0.0001, which is highly significant. As a second benchmark, the occlusion of the goal by the robots was calculated. For the new supporter behavior, the mean occlusion is 0.7750 m, compared with a mean of 0.6496 m for the old behavior. The entire goal width is 1.5 m. The t-test indicates that also this improvement is highly significant. The results are summarized in Table 1.

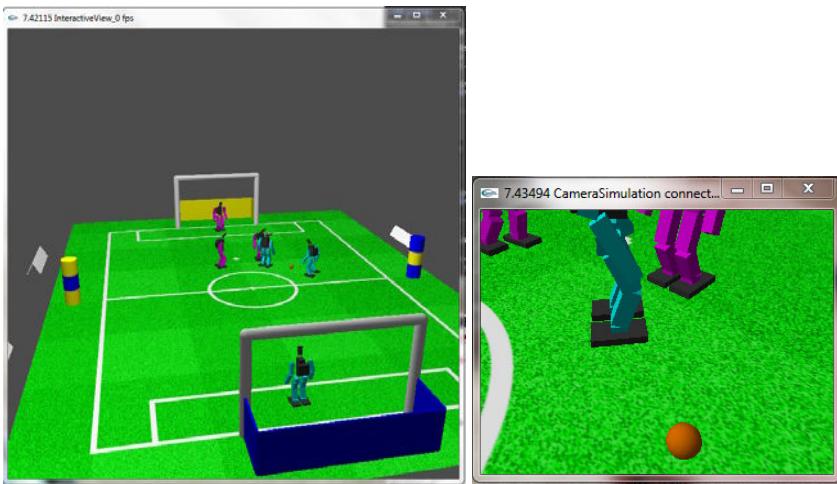


Fig. 5. Screenshot of the simulation experiments. Left: Match Overview. Right: Camera perspective of the cyan supporter.

Table 1. Overview of the simulation experiments

Behav. team 1	Behav. team 2	Sim. halftimes	Mean goal diff.	Mean goal occl.
old	old	43	0.1395	0.6496
new	old	44	1.7727 highly significant $P = 1.1010e-004$	0.7750 highly significant $P = 1.1623e-021$

The improved supporter behavior has been used at RoboCup 2009. Only two of the matches (i. e. the semifinal and the final) were noteworthy, in all other matches the opponents almost never reached the ball or attempted to score a goal. Therefore, all other matches are left out of the analysis. Videos of the matches can be found at <http://www.youtube.com/user/DarmstadtDribblers#grid/user/23B57AD2A2A1D6BD>.

In the semifinal, the opponent team had a goalkeeper and two field players. Frequently, both field players tried to reach the ball simultaneously and therefore blocked each other. Therefore, there were only few opponent kicks, instead the ball was only moved slightly by the opponents by walking against the ball. This usually moved the ball less than 1m, and if the ball passed our own striker, this distance was a good hint on where our supporter should be positioned along the x-axis relative to the ball. The y-offset was chosen small (0.5m), on the one hand to avoid that the ball can pass between both field players, on the other hand because no diagonal passes were observed at the team's previous matches. The match ended with 7:2.

In the final, the opponent team had three field players and no goalkeeper. The robot that was closest to the ball always tried to approach it to score a goal, the other two were positioned at free space, one in the own and the other in the opponent half, to have a good coverage of the field. The opponents were dribbling most of the time, moving the ball approximately 1-1.5m, which gave an evidence for the x-offset. Like in the semifinal, the y-offset was chosen small to prevent balls passing between both field players, also in this match no diagonal passes were expected. The final score of this match was 11:1.

In both matches, x_{min} and x_{max} were set to the borders of the goal areas, to prevent the supporter of committing an illegal defense or illegal attack foul.

For the two relevant matches, the interesting situations are summarized in Table 2. Overall, there were 28 situations when the striker failed (i.e. got dodged by an opponent or fell over), and therefore a well positioned supporter was needed to recover the situation. If the supporter was able to play the ball before an opponent did, the supporter was considered to be helpful (+). If the supporter could not reach the ball or could not move it towards the opponent goal, it was considered to be not helpful (-). If no second field player was available, the situation cannot be rated (o). Table 2 shows, that the team could benefit in many situations of the new supporter behavior.

Table 2. Analysis of the two relevant matches at RoboCup 2009. It can be seen, that the supporter frequently recovered a situation after the striker failed.

match	striker situation	supporter +	supporter -	supporter o
Semifinal	got dodged	12	7	7
	fell over	4	1	3
Final	got dodged	7	3	1
	fell over	5	3	1
Total		28	14	12

6 Conclusion

The development of humanoid robot technologies has reached a state where reasoning about high-level behavior development must be made. Indeed cognitive and mechanical abilities of humanoid robots are not enough developed to accurately reproduce human soccer tactics. However, with some adjustments key concepts of positioning and cooperation can be well transferred from human to humanoid robot soccer behavior. The proposed behavior is minimalistic with respect to required computing power, motion abilities and perception, and is therefore very well suited for humanoid soccer robots.

The behavior described in this paper was evaluated in simulations and has shown to yield significantly better results than the old behavior, regarding goal difference and goal occlusion. The results at RoboCup 2009 show that also in real matches with opponents with different tactics the improved supporter behavior

is efficient and contributes to the overall team success. During RoboCup 2009 it was possible to adjust the behavior to observable characteristics of the opponents robots and playing style. Behavior parameters were set mainly regarding the walking speed of the opponent robots and the range of their shots. It has shown that hand-tuned parameters worked already well in practice, however, for future work, reinforcement learning techniques can be applied to find better parameters.

Further, refinement of the behavior will come along with improvements of the mechanical abilities like passing and ball control, opponent robot modeling, a larger playing field and a rising number of field players.

Acknowledgment. Parts of this research have been supported by the German Research Foundation (DFG) within the Research Training Group 1362 “Cooperative, adaptive and responsive monitoring in mixed mode environments”.

References

1. Behnke, S., Stückler, J.: Hierarchical reactive control for humanoid soccer robots. *International Journal of Humanoid Robots (IJHR)* 5(3), 375–396 (2008)
2. Friedmann, M., Petersen, K., von Stryk, O.: Adequate motion simulation and collision detection for soccer playing humanoid robots. *Robotics and Autonomous Systems* 57, 786–795 (2009)
3. Phillips, M., Veloso, M.: Robust supporting role in coordinated two-robot soccer attack. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) *RoboCup 2008. LNCS*, vol. 5399, pp. 235–246. Springer, Heidelberg (2009)
4. Playne, D.: Knowledge-based role allocation in robot soccer. In: 10th International Conference on Control, Automation, Robotics and Vision, ICARCV 2008, December 2008, pp. 1616–1619 (2008)
5. Risler, M., von Stryk, O.: Formal behavior specification of multi-robot systems using hierarchical state machines in xabsl. In: *AAMAS 2008-Workshop on Formal Models and Methods for Multi-Robot Systems*, Estoril, Portugal, May 12-16 (2008)
6. Röfer, T., Laue, T., Müller, J., Bösche, O., Burchardt, A., Damrose, E., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Rieskamp, A., Schreck, A., Sieverdingbeck, I., Worch, J.-H.: B-human team report and code release (2009), http://www.b-human.de/media/coderelease09/bhuman09_coderelease.pdf
7. Work, H., Chown, E., Hermans, T., Butterfield, J., McGranaghan, M.: Player positioning in the four-legged league. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) *RoboCup 2008. LNCS*, vol. 5399, pp. 391–402. Springer, Heidelberg (2009)

Utilizing the Structure of Field Lines for Efficient Soccer Robot Localization

Hannes Schulz, Weichao Liu, Jörg Stückler, and Sven Behnke

University of Bonn, Institute for Computer Science VI,

Autonomous Intelligent Systems, Römerstr. 164,

53117 Bonn, Germany

{schulz, liu, stueckler, behnke}@ais.uni-bonn.de

Abstract. The rules in RoboCup soccer more and more discourage a solely color-based orientation on the soccer field. While the field size increases, field boundary markers and goals become smaller and less colorful. For robust game play, robots therefore need to maintain a state and rely on more subtle environmental clues. Field lines are particularly interesting, because they are hardly completely occluded and observing them significantly reduces the number of possible poses on the field.

In this work we present a method for line-based localization. Unlike previous work, our method first recovers a line structure graph from the image. From the graph we can then easily derive features such as lines and corners. Finally, we describe optimizations for efficient use of the derived features in a particle filter. The method described in this paper is used regularly on our humanoid soccer robots.

1 Introduction

On its way to realistic soccer environments, RoboCup started out with small-sized, color-coded scenarios. Gradually, artificial markers, colors and special lighting are removed and the soccer field size increases to encourage teams to build reliable vision systems which can compete under real-world conditions. While other leagues, like the MidSize league, already went a long way, the humanoid league is still at the beginning of this transition. Especially the small available computational power and noisy observations due to mostly unmodelled motion models prevented large steps so far. However, from the experience in MidSize-league we can learn that the removal of colored landmarks emphasizes the importance of field lines. For precise positioning, for example during setup phase of the game, use of lines and crossings is already mandatory since the uncertainties of other landmarks are prohibitively large. Finally, the restricted field of view in humanoid robots – as opposed to “omnivision” commonly used in other leagues – can be partially compensated for using field lines.

In this work we present a system for detecting field-lines and employing them for localization. This is by far not the first method presented for the purpose; however, our approach has several advantages. The line structure is determined using algorithms inspired from work on analysis of handwritten digits [1]. This method employs local and global cues to determine a graph which captures the essential structure of lines in an

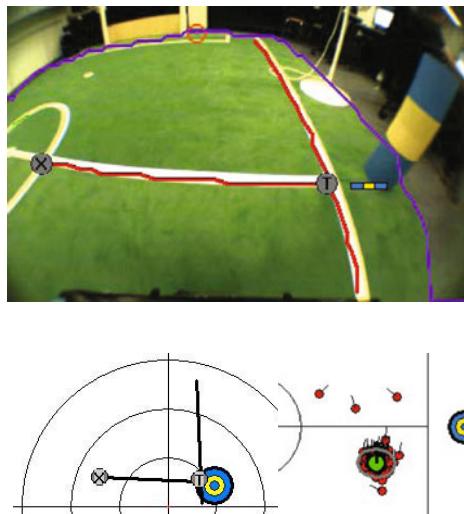


Fig. 1. Localization using line and corner features. The top-figure shows an image taken from the robot's front camera. The purple line denotes the detected field boundary, red (green) lines show field lines (not) used for localization. Detected corners are marked as "X" or "T". Bottom left: egocentric view with everything used for localization. Bottom right: resulting localization using the particle filter.

image taken by the robot's camera. The graph has a particularly nice structure: nodes are candidates for corners where the field lines meet with the branching factor determining the corner type. Edges in the graph correspond to field lines. Most importantly, the estimates of line parameters are not influenced by spurious segments or noisy locally estimated orientations.

Notably, the favorable properties of the algorithm come at little cost. We incorporated line and corner features into a particle filter which runs online on our humanoid soccer robots. This is possible, because the costly per-particle association decision of observed features to landmarks can be simplified.

The remainder of the paper is organized as follows. The next section reviews previous work on line-based localization. Section 3 and 4 describe preprocessing and feature extraction, respectively. In Section 5 we describe how line and corner features can be used in a particle filter. We provide qualitative and quantitative experimental results using motion capture as ground truth in Section 6.

2 Related Work

Work on field-lines for localization in RoboCup environments can be described on three axes. First, how candidate line segments are found; second, how the line segments are merged to lines; and third, how detected lines are used for localization. Naturally, the literature describes only combinations of all three approaches and this work is no exception. Nevertheless, this work contributes to all three areas.

Finding candidate line segments is usually performed using green-white-green transition filters [2] or scan-lines [3,4] on the image. However, convolving the image is an expensive operation and scan-lines ignore context. Our method simply makes use of all white pixels and rejects those which are not part of the field lines.

Candidate line segments are then processed to find actual lines. Hough-space techniques are commonly used for this purpose (e.g. [5], [6]). These methods need to calculate a large accumulator array and require double book-keeping of line-pieces for post-processing the found lines. Also, estimated line orientations of small segments tend to be quite noisy. The same is true for the approach followed by the NAO team NUManoid [2], where lines are determined by adding and removing points to candidate lines which requires tuning of numerous parameters. In the approach presented here, candidates for lines emerge naturally from the determined line structure graph.

Finally, candidate line segments or lines can be used for localization. Lauer et. al. [7] use all candidate line segments for performing a gradient descent from the currently estimated pose. This technique relies on the favorable properties of omnidirectional cameras, which are not allowed in the humanoid league, and on stable distance estimates, which are hard to determine for humanoid robots with unknown camera pitch and roll angles. Röfer et. al. [4] also make use of all candidate line segments using a pre-calculated lookup table for the observation model. With this method, segments which are observed on a line can be associated with different lines in the world, supporting improbable poses. In our experience, this approach also requires a large floor and standard deviations in the observation model such that spurious line segments do not completely destroy the belief. For precise localization, this approach is not helpful. Furthermore, the computational load is high if it's used in combination with particle filters, where each particle has to integrate information from all candidate line segments. Consequently, in humanoid and AIBO teams, field line extraction from candidate line segments prevails for localization purposes. The resulting long lines can then either be used as pose constraints [8] or directly used in a particle filter. Pose constraints rule out many possible poses and need to be relaxed iteratively in the case where no allowed poses are left. Particle filters can represent much more complex beliefs. However, to achieve real-time performance, speed optimization is critical. In this work, we describe optimizations used to acquire a high frame-rate even when many features are detected in an image.

Table 1. Timing of line-detection code in relation to other visual processing tasks

Program Part	Time (Standard Deviation) in μ s	
Color classification	1578.5	(33.2)
Pre-processing	421.6	(3.7)
Object detection	661.2	(94.5)
Lines preprocessing	128.3	(6.9)
Lines connect nodes	107.0	(30.3)
Split/merge lines	101.5	(35.4)
Verify crossings	35.0	(20.5)
Particle filter	3068.3	(1601.4)

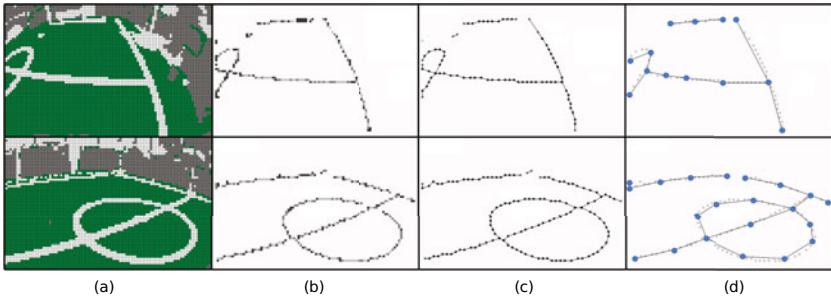


Fig. 2. Visualization of vectorization process. (a) subsampled image of pixels classified as “white” and “green”, with cluttered background. (b) skeleton inside field boundary. (c) all nodes with connections. (d) keynote structure graph.

3 Vectorization

The first step of our algorithm is the vectorization of the field lines. Our robots are equipped with three color cameras (IDS uEye 1226 LE) which capture images with a WXGA (752×480) resolution in YUV 4:2:2 color space. The individual pixels are classified to color classes as follows. First, the Y-component is compared to luminance thresholds for classification of black and white. For pixels with intermediate luminance, the color class is defined by a lookup-table for the U and V values. Color classes are described by ellipses in the UV plane. In addition, each color class is restricted to an interval in the Y dimension.

The pixels classified to a color class are counted in a separate 94×60 grid (one eighth of the our camera resolution in each dimension). Each pixel in this color class image represents the number of occurrences of its color in a 8×8 window of the original image. While this subsampling reduces spatial resolution, it allows for quick access to the density of the corresponding color in image regions. All algorithms below are performed on these subsampled color class images, albeit we use subpixel precision when possible.

In the next processing step, simple pre-processing operations are performed on the color images. We reduce unwanted effects such as Bayer-pattern induced orange and cyan colors next to sharp contrasts. In unreliable corner-regions of our wide-angle lens we delete all classified colors. In Fig. 2(a), we exemplarily show the result of classification for green and white.

The vectorization of lines is now implemented in two major steps: We first extract the field boundary, and second, determine the structure of lines within the field boundary. The following sections describe the two steps in detail.

3.1 Field Boundary Extraction

In robot soccer games, everything of interest is located on the green carpet. While the area outside the field is undefined and cluttered, objects on the field can be clearly distinguished by color and structure. Consequently, as a first step we segment the field region, thereby further reducing the area of interest.

We use a three-step boundary scanning algorithm. This algorithm first integrates the subsampled color-images into a gray-level image in a manner that it can better represent the field region. Then, it retrieves a histogram of field height in the image. In a final postprocessing step, we smooth and fill local gaps in the histogram.

Merging Color-Images. We want to determine the boundary of the green field. However, objects on the field might occlude some parts of the field. Therefore, we create a new 94×60 gray-level image which is composed by a weighted sum of previously classified colors according to their probability of being part of the field or occluding it.

Binarization. For a pixel to be part of the field, it must exceed a minimum threshold itself, be part of a largely green line or have enough field pixels below. After merging, we use three thresholds t_{pix} , t_{row} and t_{win} to determine whether a pixel g_{xy} is likely to be inside the field. The threshold tests are arranged in order of complexity to deal with the obvious cases first. First, we check the value of the pixel itself. If $g_{xy} \geq t_{pix}$, the pixel undergoes further examination. If the pre-calculated row-sum $\sum_{x'} \sum_{\{y'=-1,0\}} g_{x',(y+y')}$ is larger than some threshold t_{row} , the pixel is binarized to 1. The row-sum acts here as a prior which biases weak pixels in rows with many high-valued pixels to be part of the field. Pixels which do not pass t_{row} are examined by the most expensive test, by examining the sum s_{xy} of their neighbors below in an 8×4 neighborhood and comparing s_{xy} to t_{win} .

Retrieving Field Image Height Histogram. Assuming that the robot should always be located somewhere on the field, the field in the image always starts at the bottom and reaches up to some level. This simplified view can be captured in a histogram of image heights.

We count consecutive binary pixels with value 1 and 0 respectively in each column from bottom-up. Once we encounter more than four consecutive pixels with value 0, the algorithm sets the corresponding bin to be the y -coordinate of the last 1-valued pixel and proceeds to the next column.

Smoothing and Gap Filling. The histogram so far is a rough field boundary with gaps and peaks caused by uneven illuminations and imprecise color classification. We consequently smooth the histogram using a 1D Gaussian filter for slightly uneven bins and a median filter for small gaps and peaks.

Finally, we employ a local convex corner algorithm based on [9] to only fill in the remaining gaps but not include unwanted regions like the opponent robot's body or white goal posts attached to the field. This algorithm simply calculates the local convexity:

$$v_{xy} = (R_x - L_x) \cdot (P_y - L_y) - (P_x - L_x) \cdot (R_y - L_y)$$

where P_x is the number of the current bin and P_y is its value; L_x , L_y , R_x and R_y are the respective values of the neighboring left and right bins. If v_{xy} is 0, the top point P_y of the current x is collinear with its left and right neighbors; if v_{xy} is positive, the current bin is above the line segment between the top points of its left and right neighbors and it is below that line segment if v_{xy} is negative. The locally convex hull of the field is then determined by the heights of bins with associated non-negative results. Iterating this scheme will eventually lead to a globally convex hull, but for our purposes one iteration suffices. The result of the field boundary extraction is shown in Fig. 1 for an example image.

3.2 Preprocessing and Skeletonization

With the information of the field boundary, we only process the classified white pixels inside this boundary. Then, a low-pass filter is applied to make sure that each line cross-section has only a single maximum-valued pixel. Skeletonization [1] is used to reduce the line width to approximately one pixel. Unlike morphological methods which start from the border of the line and iteratively erode it, we use a ranking operator, which directly finds a skeleton in the middle of a line. The operator observes 3×3 pixel regions to decide if the central pixel belongs to the skeleton. Pixel having gray-level zero do not belong to the skeleton, but to the background. For all other pixels, the number $c(x, y)$ of neighboring pixels (8-neighborhood) having an equal or higher gray-level is computed and if this number is less than three the central pixel is added to the skeleton. Fig. 2(b) visualizes skeletons resulting from two soccer-field images.

3.3 Placement and Connection of Nodes

Nodes are placed starting at peaks of the skeleton ($c(x, y) = 0$). This emphasizes crossings, which tend to appear as peaks in the skeleton. Note however, that crossings detection does not depend on correct node placement at this stage. Then nodes are placed at least two pixels apart at pixels belonging to ridges ($c(x, y) = 1$ and $c(x, y) = 2$).

The nodes now need to be connected to represent field lines. First, the connection structure of the skeleton is reconstructed by inserting connections where 3×3 regions of nodes overlap or touch on the skeleton. In order to insert the few remaining connections necessary to recover the original field lines, more global information of adjacent connections is used. Lines are stretched by moving end-nodes to the last pixel of the skeleton and degree-0 nodes are split to fill in the gaps. Finally, candidate connections are created and evaluated according to continuity, closure and simplicity. Specifically, the distance between nodes should be small and the grayness on the line between nodes should be similar to grayness of nodes. Furthermore, we place restrictions based on the degree of nodes, ruling out crossings of degree greater than 4 and crossings which do not result in continuous lines. Examples are shown in Fig. 2(c); we refer the interested reader to [1] for details.

4 Feature Extraction

We can now easily extract features such as crossings or lines from the node structure and verify them in the image.

4.1 Line Crossing Detection

The node connections produced so far are the original structures of field lines in the image with many details and distortions. To extract the key structure, the lines are smoothed and nodes are removed at locations of low curvature. Short lines ending in junctions are eliminated and junctions that are close together and connected are merged to form a crossing. When merging and moving nodes, we make sure that the new nodes are placed on the skeleton to guarantee an undistorted world view. The result is shown in Fig. 2(d).

Crossing detection here is now dramatically simplified due to the key node structure representation. A degree-2 node with certain angle of edges connected to it is an L-crossing candidate, a degree-3 node is a T-crossing candidate, and a degree-4 node is an X-crossing candidate. To verify whether one candidate represents a real crossing, we first use a state machine to check out the green-white and white-green color transition along a circle centered at the crossing. Then, we check whether there is a white path from the crossing to the next neighbors in each direction. Both checks are performed in the sub-sampled color-images.

4.2 Field Line Extraction

Starting with the fine-grained, connected graph of observed line segments (Fig. 2(c)) we can extract field lines. Here, a field line is a connected set of nodes with degree two and nodes connected directly to the set with different degree.

The coordinates of the nodes can be approximately connected by a straight line. Consequently, we first traverse the graph to extract connected components of degree two nodes. Because we rely on wide-angle lenses, straight lines in the world do not result in straight lines in the image. Before proceeding, we therefore calculate undistorted coordinates of the nodes. Each component c_i is then processed using the split-and-merge algorithm: we fit a line l_i to c_i using least squares regression [10] on the coordinates of the nodes. The node $n^* = \arg \max_{n \in c_i} \text{dist}(n, l_i)$ with the largest distance to the fitted line defines a splitting point. We split c_i into two components $c_i^{1/2}$ if the node-line distance is sufficiently large and recursively process the resulting components. Components containing less than three nodes are discarded. During the merging phase, we merge components c_i and c_j if the parameters of l_i and l_j are sufficiently similar.

The final field line in image coordinates is determined by projecting the end-points of the component onto the fitted line. We do not use the end-points directly, since these can represent part of another line which barely did not pass the splitting threshold. If c_i contains many nodes, its (possibly wrongly matched) end-points will have limited influence on l_i , resulting in a better approximation. Lastly, by projecting both end-points to the floor, we can infer the line parameters (distance and angle) in a egocentric coordinate frame.

5 Integration of Features

As described in [6], we use Monte-Carlo localization (MCL, [11]) to estimate the current 3D pose of our soccer robots. The pose is a tuple (x, y, θ) , where (x, y) denotes the position on the field and θ is the orientation of the robot. The belief is updated recursively with:

$$p(x_t | z_{1:t}, u_{1:t-1}) = \eta \cdot p(z_t | x_t) \cdot \int p(x_t | x_{t-1}, u_{t-1}) \cdot p(x_{t-1} | z_{1:t-1}, u_{0:t-2}) dx_{t-1}, \quad (1)$$

where η is a normalization constant resulting from Bayes' rule, $u_{0:t-1}$ is the sequence of all motion commands executed by the robot up to time $t-1$ and $z_{1:t}$ is the sequence of all observations. The term $p(x_t | x_{t-1}, u_{t-1})$ is called motion model and denotes the

probability that the robot ends up in state x_t given it executes the motion command u_{t-1} in state x_{t-1} . The observation model $p(z_t|x_t)$ is the likelihood of making the observation z_t given the robot's current pose is x_t . MCL uses a set of random samples to represent the belief of the robot about its state at time t . Each sample consists of the state vector $x_t^{(i)}$ and a weighting factor $\omega_t^{(i)}$ that is proportional to the likelihood that the robot is in the corresponding state. The update of the belief is carried out according to the sampling importance resampling particle filter. First, the particle states are predicted according to the motion model. For each particle, a new pose is drawn given the executed motion command since the previous update. In the second step, new individual importance weights are assigned to the particles. Particle (i) is weighted according to the likelihood $p(z_t|x_t^{(i)})$. Finally, a new particle set is created by sampling from the old set according to the particle weights. Each particle survives with a probability proportional to its importance weight. This step is also called resampling.

In order to allow for global localization, e.g. in case of the “kidnapped robot problem”, a small amount of the particles is replaced by particles with randomly drawn poses. Additional particles are used if pose certainty suddenly drops (Augmented MCL, [12]).

5.1 Crossing Observations

We treat crossing observations similar to other point features on the field (e.g., center of goal, goal posts, marker poles). However, in contrast to the other point features, crossings are not unique. To calculate the likelihood of an observation in the particle filter, we have to make an association decision: for a line crossing observation o , each particle i and all crossings C of the same type (X/T/L), we must calculate the most likely association

$$o' := \arg \max_{c \in C} p(c | x_t^{(i)}). \quad (2)$$

While the result of the calculation can be re-used in the second step of the sampling importance resampling, evaluating the observation model repeatedly is expensive and limits the number of particles. However, we can considerably reduce the size of C by considering the egocentric orientation of T and L-crossings. Consider, for example, a particle at the centerline looking towards the yellow goal and observing an L-crossing oriented towards the particle. This particle is quite unlikely to observe the L-crossings next to the blue goal, which are oriented exactly opposite allocentrically. It is also unlikely to see the L-crossings next to the yellow goal which point towards the yellow goal. Consequently, we split the set \mathcal{L} of L-crossings into four sets $\mathcal{L}_{45^\circ}, \dots, \mathcal{L}_{315^\circ}$ containing two L-crossings of equal global orientation each. A pre-calculated, coarse lookup table $\mathbb{R}^2 \times \{45^\circ, 135^\circ, 225^\circ, 315^\circ\} \mapsto \mathcal{L}$ then associates an observation position on the field and an allocentric, oriented L-crossing observation with the closest L-crossing. We proceed similarly with the T-crossings, but since the distance between different T crossings is large, a lookup table mapping positions to closest T-crossings is sufficient. For X-crossings, we calculate the most likely crossing on a per-particle basis according to Eqn. (2) using the observation model. For an egocentric observation (d_o, β_o) and an

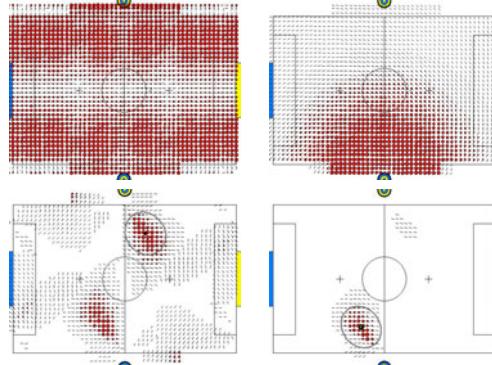


Fig. 3. Particle filter belief visualization based on observed landmarks. We used particles on a 3D-grid and show sizes and orientations based on the likelihood. The features used for localization are the same as in Fig. 1. Top left: only lines, top right: only pole, bottom left: only corners, bottom right: poles, corners and lines.

expected egocentric position ($d_e^{(i)}, \beta_e^{(i)}$) relative to the current particle i , we define the observation model to be

$$p_{\text{point}}(o|x_t^{(i)}) \propto \exp \left(-\frac{\|d_e^{(i)} - d_o\|^2}{2(\sigma_d + \lambda d_o)^2} - \frac{\|\beta_e^{(i)} - \beta_o\|^2}{2\sigma_\beta^2} \right), \quad (3)$$

where σ_d and σ_β represent the uncertainty of a range and bearing measurement, respectively. Note that the uncertainty of distance measures increases for far away objects to compensate for the unknown pitch and roll angle of our camera. Fig. 3 (top right) shows the belief resulting from the observation of a point-landmark.

5.2 Line Observations

In our current system, we ignore lines which are short and far away in terms of distance of the line to the origin. For each remaining line o represented by length of dropped perpendicular l_o , distance to closest observed point d_o , and expected angle β_e , we evaluate the observation model

$$p_{\text{line}}(o|x_t^{(i)}) \propto \exp \left(-\frac{d^2(l_e^{(i)}, l_o)}{2(\sigma_l + \lambda d_o)^2} - \frac{\|\beta_e^{(i)} - \beta_o\|^2}{2\sigma_\beta^2} \right). \quad (4)$$

Here, $d(\cdot, \cdot)$ depends on the oriented distance: in contrast to point landmarks discussed above, the orientation β_o represents the angle of the observed line, not the angle of a polar coordinate. As a result a simple observation model which does not take into account oriented distance, would assign equal likelihood to the situation where the robot observes the line behind itself and in front of itself, although the position of the line in front of or behind the robot can be directly inferred from the observation. We therefore set in Eqn. (4)

$$d(l_e, l_o) = \begin{cases} \|l_e - l_o\|_2 & \text{if } \langle l_e, l_o \rangle > 0 \\ \infty & \text{else,} \end{cases} \quad (5)$$

which eliminates high likelihood of the implausible situation. In contrast to corner observations, we cannot make a distinction between the seven long lines in our world. We use Eqn. (2) to determine the most likely match on a per-particle basis. Note, however, that due to monotonicity of \exp , for the arg max computation in Eqn. (2) it is not necessary to evaluate the likelihood in Eqn. (5) completely. Instead, we apply $\exp(\cdot)$ after the association has been made and rely on the minimum argument of \exp for the association decision itself. In Fig. 3 (top left) we visualize the belief resulting from the observation of the two lines in Fig. 1.

5.3 Combined Observation Model

For each observation o_j , we ensure that its observation likelihood is larger than some uniform threshold. We further incorporate confidence values c_j from our vision system such that unconfident observations o_j have less influence on the final distribution than confident observations in the same frame:

$$p(o_j | x_t^{(i)}) = \alpha_{\text{uni}} p_{\text{uni}}(o_j | x_t^{(i)}) + \alpha_{\text{normal}} p(o_j | x_t^{(i)}),$$

where $\alpha_{\text{uni}} = \alpha_{\text{base}} + (1 - \alpha_{\text{base}}) \cdot (1 - c_j)$ and $\alpha_{\text{base}} \in]0, 1[$ is a uniform floor. We further set $\alpha_{\text{uni}} + \alpha_{\text{normal}} = 1$ and $p_{\text{uni}}(o_j | x_t^{(i)})$ to be the Lebesgue measure of the observation range. Assuming independence of observations, as typically done in MCL, the observation likelihood of a particle then amounts to the product of all single observations

$$p_{\text{comb}}(z_t | x_t^{(i)}) \propto \prod_{l \in \mathcal{L}} p_{\text{line}}(l | x_t^{(i)}) \prod_{o \in \mathcal{P}} p_{\text{point}}(o | x_t^{(i)}), \quad (6)$$

where \mathcal{P} includes corners and other point landmarks such as goal centers and poles marking the end of the center line. The combined belief resulting from observations of point and line landmarks is shown in Fig. 3 (bottom right).

6 Results

Our 1.3 GHz system runs at about 24 frames per second using between 250 and 1000 particles (depending on the certainty of the current pose). Tab. 1 shows the relative timing results. Soccer robots profit enormously from line-based localization. Consider the pose certainty of the robot in our running example. We illustrate the beliefs in Fig. 3 by equally distributing particles in a regular 3D-grid. Besides lines and corners, the robot only observes a pole at the side of the field. With color-based localization only, the position on the field is not at all clear from the image. Using only corners, the belief is reduced to two positions. The only colored landmark observed is then enough to further reduce the pose belief to an almost unimodal distribution. For a given frame, we run a particle filter update and select the most likely orientation z for each planar position p_{xy} . We then show this particle sized in proportion to its likelihood at position (x, y) .

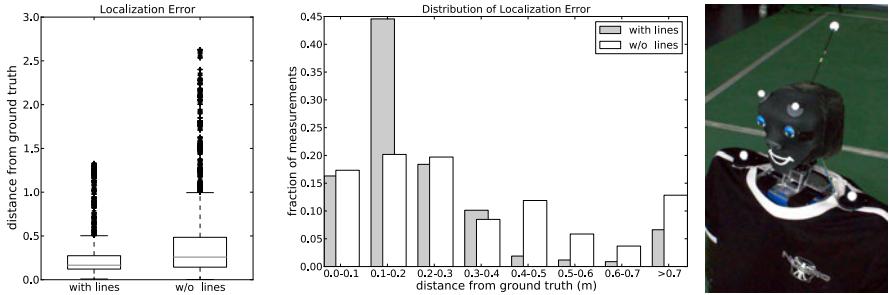


Fig. 4. Left: Localization error with lines (without lines). The median accuracy is 17 cm (26 cm). Center: Distribution of distances between estimated and ground truth pose. Right: TeenSize Robot “Dynaped” with attached infrared-reflective markers.

To further quantify the performance of our algorithm, we let our TeenSize robot “Dynaped” walk for about 5 minutes across the field using remote-control. Robot and field are configured to be rule conformant to the rules of 2010, namely, the robot only uses a single camera, only the posts of the goal are colored and center line pole sizes are reduced. We fix the movable head of the robot to look straight at all times to eliminate effects due to active vision. In addition to estimated poses we record ground truth positions using an Optitrack motion capture system and infrared-reflective markers attached to the robot’s torso and head. The experiment was performed twice, once using line and crossing detection and once without, with similar trajectories. Both trajectories contain about 10000 poses. The average estimated speed was 20 cm/s (with lines) and 22 cm/s (without lines); we removed 10% (7%) of the recorded frames where the motion capturing tracking failed. We then determined the percentage of recorded frames for which the distance between the robot’s estimated pose to ground truth was in some interval. The result is depicted in Fig. 4. Note that without lines only 66% of the recorded poses are within 40 cm of the ground truth while with line and crossing detection enabled 89% are in this range. With lines, the median localization error was 17 cm, which is less than the average step length. This error is also within the range of the robot’s trunk movements which are due to walking-based weight shifts, which can be seen as an error of the motion capturing process.

In the TeenSize league, robots play on a field of the same size as the KidSize field, but the robots are – by definition – much larger. The structural information which is visible from the perspective of a TeenSize robot is consequently very informative. Our TeenSize robot Dynaped, the champion of the Graz 2009 world cup could therefore mainly rely on allocentric information from the particle filter for its decisions instead of previously used egocentric information.

7 Conclusion

In this work, we introduced a line-detection method based on methods developed for the identification of structure in handwritten digits. We first described how to find the boundary of the soccer field. White points within this region are then skeletonized and

a simple graph structure is retrieved from the skeleton. The graph structure has advantageous properties: field-line corner candidates are represented by nodes and field-lines are represented by edges. This graph structure is then used to verify linear components and crossings and to determine their parameters. Due to the graph representation, clustering or averaging of noisy locally estimated line parameters can be avoided. Finally, we showed how lines and crossings can be used in Monte-Carlo localization. The paper describes observation models and optimizations to speed up association decisions. Our observation models were exemplarily demonstrated using visualizations of resulting particle filter beliefs. Evaluation of our localization on the real robot showed smaller differences to ground truth when lines and corners were used. The algorithms described were used on our KidSize and (winning) TeenSize soccer robots during RoboCup 2009.

Acknowledgements. This project is supported by the German Research Foundation (DFG), grant BE2556/2-2 and BE2556/4.

References

1. Behnke, S., Pfister, M., Rojas, R.: Recognition of handwritten digits using structural information. In: Proc. of ICNN (1997)
2. Henderson, N., Nicklin, P., Wong, A., Kulk, J., Chalup, K., King, R., Middleton, H., Tang, S., Buckley, A.: The 2008 NUmroids Team Report (2008)
3. Sridharan, M., Stone, P.: Real-time vision on a mobile robot platform. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 2148–2153 (2005)
4. Röfer, T., Jüngel, M.: Fast and robust edge-based localization in the sony four-legged robot league. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 262–273. Springer, Heidelberg (2004)
5. Bais, A., Sablatnig, R., Novak, G.: Line-based landmark recognition for self-localization of soccer robots. In: Proceedings of the IEEE Symposium on Emerging Technologies, 2005, pp. 132–137 (2005)
6. Strasdat, H., Bennewitz, M., Behnke, S.: Multi-cue localization for soccer playing humanoid robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, p. 245. Springer, Heidelberg (2007)
7. Lauer, M., Lange, S., Riedmiller, M.: Calculating the perfect match: An efficient and accurate approach for robot self-localization. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, p. 142. Springer, Heidelberg (2006)
8. Borisov, A., Ferdowsizadeh, A., Gohring, D., Mellmann, H., El Bagoury, M., Kohler, F., Welter, O., Xu, Y.: NAO-Team Humboldt (2009)
9. Graham, R.: An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set. In: Information Processing Letters, pp. 132–133. Elsevier, Amsterdam (1972)
10. Kenney, J., Keeping, E.: Mathematics of Statistics, Linear Regression and Correlation, ch. 15. Van Nostrand (1962)
11. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: Proc. of ICRA, pp. 1322–1328 (1999)
12. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. The MIT Press, Cambridge (2005)

Robot Detection with a Cascade of Boosted Classifiers Based on Haar-Like Features

F. Serhan Daniş, Tekin Mericli, Çetin Mericli, and H. Levent Akın

Boğaziçi University, Department of Computer Engineering, 34342, Bebek, Istanbul, Turkey
`{serhan.danis, tekin.mericli, cetin.mericli, akin}@boun.edu.tr`

Abstract. Accurate world modeling is important for efficient multi-robot planning in robot soccer. Visual detection of the robots on the field in addition to all other objects of interest is crucial to achieve this goal. The problem of robot detection gets even harder when robots with only on board sensing capabilities, limited field of view, and restricted processing power are used. This work extends the real-time object detection framework proposed by Viola and Jones, and utilizes the unique chest and head patterns of Nao humanoid robots to detect them in the image. Experiments demonstrate rapid detection with an acceptably low false positive rate, which makes the method applicable for real-time use.

1 Introduction

When we consider multiple autonomous robots operating in the same environment, the robots should be aware of each other in order to plan their actions accordingly and maximize the overall utility. Accurate world modeling through detection of other robots becomes even more important for a robot operating in environments which are both adversarial and collaborative such as the game of soccer.

In soccer, the players make their plans considering the positions and actions of the teammates and the opponent players. The ability to detect the other players and keep track of their locations on the field paves the way for more efficient utilization of multiple agents through higher level plans, team play, and tactics. Therefore, in order to improve the quality of the games and increase the chance of scoring by means of proper positioning and passing, detection of the robots on the field as a part of the world modeling process becomes essential. In this paper, we present a robot detection method based on Haar-like features specifically for the RoboCup Standard Platform League (SPL) [1], where Aldebaran Nao humanoid robots [2] with on board sensing capabilities, a limited field of view, and a 500 MHz processor are used.

Nao robots in the 2009 versions come with red and blue colored patches in various shapes on some parts of their bodies that are used as the “uniforms” to distinguish between the robots of different teams during SPL games. Their human-like faces and chest patches with a unique shape are the two main features we utilize in detection. Fig. 1 gives an idea about the physical appearance of the Nao.

Although there is a broad literature on feature-based object detection and recognition, the most relevant work is the one by Fasola and Veloso [3], where a similar approach is used to detect the uniforms of the Sony Aibo robots [4], the previously



Fig. 1. Aldebaran Nao humanoid robot with colored parts

used robot platforms of the SPL. The main motivation of that study was prediction of whether the ball is occluded by a robot or not. They also compared this approach with a method that uses colored blobs in a color segmented image to detect the uniforms. However, as opposed to the Aibo case, there are more colored patches on the Nao and they are so far apart from each other, which makes it difficult to use colored blob based algorithms for robot detection purposes.

The rest of this paper is organized as follows. Section 2 elaborates on the methodology we followed. The experiments are explained and the obtained results are discussed in Section 3. Section 4 summarizes and concludes the paper while pointing to some possible future extensions.

2 Methodology

Haar-like features are image features used for generic object recognition purposes. They are named after Haar wavelets for their similarity. The use of a cascade of boosted Haar-like features for object detection was proposed by Viola and Jones, and a robust face detector based on this method has become nearly the industry standard [5,6].

The feature extraction method proposed by Viola and Jones, scans the images at 11 different scales and uses an intermediate transformation of the image, called the *integral image*. In contrast to the conventional approach that requires computation of pyramids of 11 images, they use a cascading property that reduces the computation time and increases the detection performance. A variant of AdaBoost, an adaptive meta-algorithm for improving the learning performance by adjusting the strong classifier threshold to minimize the false negatives, is used for deciding which of the extracted features the weak classifiers contribute the most to form the final strong classifier.

The whole system is packed under the name “The Haartraining” and used as a generic object detection training algorithm in the literature [7].

OpenCV is a very popular computer vision software library that provides a wide range of operators, classifiers, and detectors [8]. It implements *Haartraining* [7].

The training of such classifiers requires both positive image samples containing only the patterns of the object of interest, and negative image samples which are known for certain to not contain the object of interest.

2.1 Training

The object detection procedure employs simple features, reminiscent of the Haar basis functions [9]. In [5], three types of such Haar-like features are utilized; namely, two, three, and four rectangle features, which help in computing the difference between the sum of pixels within rectangular areas in the image. These features are shown in Fig. 2.

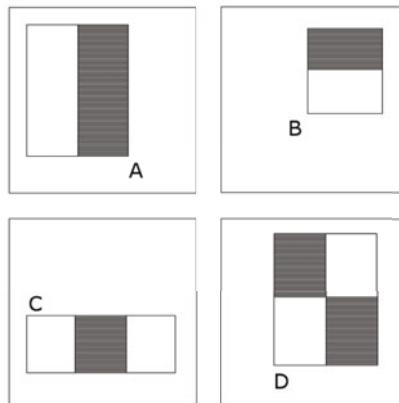


Fig. 2. Rectangular Haar-like features employed in [5]. A and B are the two-rectangle features. C is the three-rectangle feature and D is the diagonal four-rectangle feature.

The data collection and training procedures we followed for robot detection task can be summarized as follows:

1. Crop images to generate positive images that contain only the pattern to be detected
2. Pick some negative images that do not contain the patterns of interest
3. Generate M random training samples from the set of positive images
4. Generate N random test samples from the set of negative and positive images
5. Apply Haartraining to M training samples to generate an XML file of the Haar cascade descriptor
6. Evaluate the performance of the descriptor on the test samples

3 Experiments and Results

3.1 Training Data Collection and Generation

We treated different parts of the Nao's body as different objects of interest, and images containing those parts of the robot were fed to the training system. We tried training the system using the following parts of the robot, as indicated in Fig. 3:

- The face
- The chest pattern
- The whole body as a single object

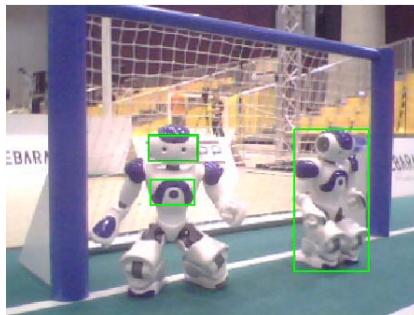


Fig. 3. Two Aldebaran Nao humanoid robots on the field, in front of the blue goal. The face and the chest parts are highlighted with rectangles. Notice the similarity between the color of the goal and the color of the robot uniforms. It is especially challenging to recognize robots accurately in such cases.

The performance of the resulting classifier heavily depends on the variety and the quality of the training data set. To increase the variety of the images in the training set, we used artificially modified images populated from the original images in the training set. The following distortions were applied:

- Adding random skew to the image to the left or to the right
- Rotating the image clockwise or counterclockwise by a random amount
- Shearing the image randomly

These distortions allow us to bootstrap a relatively small training set to a much bigger and a more diverse training set.

We generated two different training data sets per robot part to be recognized:

- Artificial images generated from a simulated environment
- Actual images from RoboCup 2009 competitions containing real robots on the field with realistic lighting and unpredictable noise sources on the competition site (spectators, advertisement boards, etc.)

We used the Webots robot simulator [10] for capturing images from a simulated environment, as shown in Fig. 4(a). With Webots we can simulate the competition environment realistically; however, the obtained images from the simulator are better compared to the real images due to the advanced rendering capabilities of the simulator. The quality of the obtained images is between a color quantized image and a normal image captured from the robot's camera.

The real world images used are extracted from the extensive log database of Cerberus team, and in particular the images captured during the RoboCup 2009 competitions are used, as shown in Fig. 4(b). Notice that the image from the simulated environment lacks the noise that exists in the real world image. Most images in the database were captured from the head camera of a Nao. We specifically used these logs instead of taking a set of pictures in the laboratory to keep the various sources of noise and uncertainty (lighting changes, uncontrollable objects such as people around

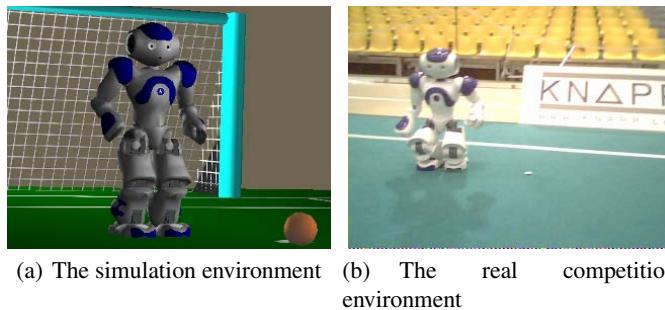


Fig. 4. Training data set images

the fields, etc.) that do exist at the competition site but usually not in isolated laboratory environments.

Different training sets were generated for the face, the chest, and the whole body of the robot using the aforementioned images. In each training set, a subset of negative images that do not contain the object of interest are also included, an example of which is given in Fig. 5(b), 5(a). These negative images are used to check for false positives once the training is completed. Several variations of positive training data with different number of images were populated per object of interest.

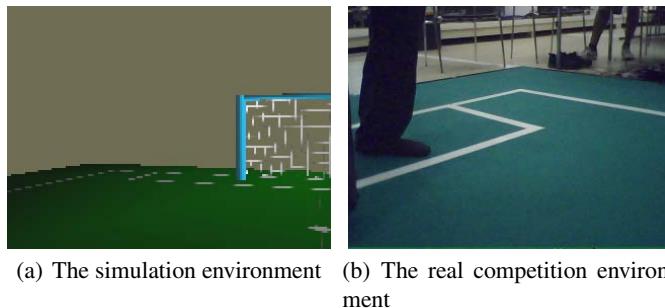


Fig. 5. Negative images that do not contain objects of interest

Artificial Images

Body: We started by training a classifier for the whole body of the robot. We prepared a set of positive training images captured from the simulator consisting of frontal and lateral views of the Nao robot, as illustrated in Fig. 6. The result of this training was quite unsatisfactory; therefore, we decided to exclude the whole body features from the process of training using real world images.

Face: To train a classifier for detecting the face patterns of Nao, we first prepared a set consisting of images captured from the simulator that include different views of the head of the robot from different angles, examples of which are depicted in Fig. 7.

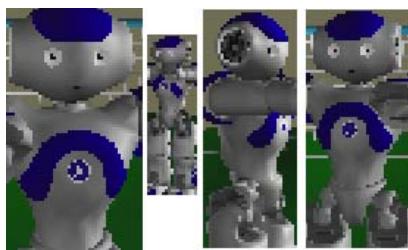


Fig. 6. Positive images of the body from the simulation world

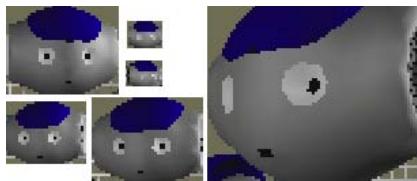


Fig. 7. Positive images of the face from the simulation world

Real World Images

Face: To obtain the training images for the face pattern from the real world logs, we cropped positive images in such a way that the resulting sub-images would contain only the frontal view of the robot's head. The two variations of frontal view used were the frontal view of the whole head (Fig. 8(a)), and the frontal view of the eye region only (Fig. 8(b)).

Chest: The specific pattern on the chest area of the Nao, as indicated in Fig. 9, is unique; that is, under normal conditions only a Nao robot has this pattern on the field. Therefore, that makes the chest region an important part for detection of the robots.

The experiments were repeated in an iterative manner using first the artificial world dataset and then the real world dataset, making hypothesis on the causes of ghost images or lack of hits, changing the number of training set elements and noisy positive

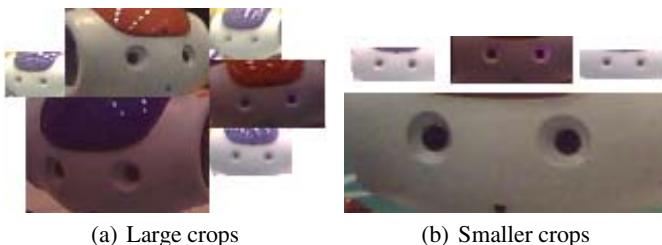


Fig. 8. Facial positive images



Fig. 9. Positive images of the chest pattern

images in the training set accordingly to arrive at a higher hit rate and less number of ghost robots. An example of noisy and clear positive images are given in Fig. 10. Noisy positive images are the ones affected by both poor illumination and perspective view, as shown in Fig. 10(a), whereas the clear images are the ones viewed frontally with the colored areas being well separated from the white areas, as shown in Fig. 10(b).

3.2 Results

To standardize the testing procedure, we kept the number of distorted images generated from the original positive images at 1000, and generated 200 test images for each trial. Table 1 summarizes the experiments according to the pattern, the original number of positive images containing the pattern, and the pattern properties in each trial.

The artificial dataset was used mostly for exploration purposes. The performance evaluation after the training using the artificial dataset yielded unacceptable results. In the case of the robot body detection, the algorithm detected 1200 “ghost” robots in addition to 20 hits, which was not desirable. Obtaining a hit ratio of 10 percent (20 out of 200) indicated that using full body of the robot as the positive patterns was not a good idea. The hit ratio was 86 percent for the artificial robot face detection; however, the false positive rate was 93 percent, which is also very high. Both results obtained in the simulation environment directed us to use more specific patterns for detection, such

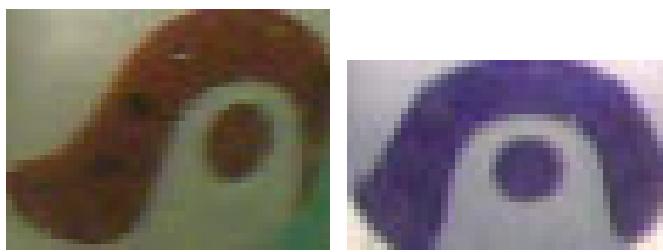


Fig. 10. Example of image types

Table 1. Comparison of different detection setups

The pattern	training images	image properties	hits	missed	false
Robot body (simulation)	30	noisy	20	180	1200
Robot face (simulation)	10	noisy	173	27	180
Robot face	27	noisy	176	24	844
Robot face	4	clear	170	30	365
Robot chest	33	noisy	169	31	113
Robot chest	30	noisy	144	56	16
Robot chest	4	clear	157	43	5

as the face or the chest pattern instead of the whole body pattern. Because of these poor results obtained in the simulation environment, the robot body detection is not repeated using the real world images.

In the first attempts, the positive sets included many noisy images. Compared to the robot face detection results in the artificial dataset, the results of robot face detection in the real world data set were worse, with a hit ratio of 88 percent and a false positive rate of 400 percent, which can be interpreted as an average of four “ghost” robots were detected in each image. Using less noisy and clearer positive images, the false positive rate was reduced down to 200 percent, which is still not acceptable since it would result in finding many “ghost” robots on the field.

The results of the chest pattern were quite acceptable compared to the previous results. As the chest pattern is specific to the Nao humanoid robot, the success of the results were expected. The false positive rate in the chest detection was only 2.5 percent, and the hit rate was 79 percent, which are highly acceptable results as the robot would not see many “ghosts” on the field, and would successfully detect approximately four of the five robots within its field of view. The required time to process an image frame for detection is 10 ms on a conventional computer. This corresponds to about 40 ms on the processor of the Nao robot ($\sim 20\text{fps}$), which is also reasonable for real-time robot detection implementation on the actual robot.

4 Conclusions and Future Work

In this work, we introduced a robust robot detection algorithm utilizing a cascade of boosted classifiers based on Haar-like features for quickly and efficiently detecting Aldebaran Nao humanoid robots during the soccer games of RoboCup Standard Platform League. We treated the robot both as a whole and as a collection of parts, and trained separate classifiers for each part. To improve the classifier performance, we investigated the possibility of augmenting the training set using artificial images captured from the simulation environment. Furthermore, we used a variety of distortions like rotation, skew, and shear to create modified versions of the training images, enlarging the training set both in size, and in variety of features. Experimental results show that contrary to the expectations of the face detection of the robot to yield better results, the chest pattern was the most distinguishable and easy to recognize part of the robot. The

resulting system has a 79 percent hit rate and runs at around 20 fps, which makes it suitable for use in actual games.

Investigation of preparing better stratified training sets, developing a method for making the selection process easy for extensive actual competition logs, and trying to use Haar features on the image channels Cb and Cr along with the channel Y are among the possible future extensions.

Acknowledgments

This project was supported by Boğaziçi University Research Fund project 09M105.

References

1. The RoboCup Standard Platform League, <http://www.tzi.de/spl>
2. Aldebaran-Nao, <http://www.aldebaran-robotics.com/eng/Nao.php>
3. Fasola, J., Veloso, M.: Real-Time Object Detection using Segmented and Grayscale Images. In: Proceedings of ICRA 2006, May 2006, pp. 4088–4093 (2006)
4. The Sony Aibo robots, <http://support.sony-europe.com/aibo/>
5. Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple Features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, p. 511. IEEE Computer Society, Los Alamitos (2001)
6. Viola, P.A., Jones, M.J.: Robust Real-Time Face Detection. International Journal of Computer Vision 57(2), 137–154 (2004)
7. Seo, N.: Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-Like Features) (2007)
8. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
9. Papageorgiou, C.P., Oren, M., Poggio, T.: A general framework for object detection. In: ICCV 1998: Proceedings of the Sixth International Conference on Computer Vision, Washington, DC, USA, p. 555. IEEE Computer Society, Los Alamitos (1998)
10. Webots Mobile Robot Simulation Environment, <http://www.cyberbotics.com>

LearnPNP: A Tool for Learning Agent Behaviors

Matteo Leonetti and Luca Iocchi

Sapienza University of Rome,
Department of Computer and System Sciences,
via Ariosto 25, 00185
Rome, Italy

Abstract. High-level programming of robotic agents requires the use of a representation formalism able to cope with several sources of complexity (e.g. parallel execution, partial observability, exogenous events, etc.) and the ability of the designer to model the domain in a precise way. Reinforcement Learning has proved promising in improving the performance, adaptability and robustness of plans in under-specified domains, although it does not scale well with the complexity of common robotic applications. In this paper we propose to combine an extremely expressive plan representation formalism (Petri Net Plans), with Reinforcement Learning over a stochastic process derived directly from such a plan. The derived process has a significantly reduced search space and thus the framework scales well with the complexity of the domain and allows for actually improving the performance of complex behaviors from experience. To prove the effectiveness of the system, we show how to model and learn the behavior of the robotic agents in the context of Keepaway Soccer (a widely accepted benchmark for RL) and the RoboCup Standard Platform League.

Keywords: Plan Representation, Reinforcement Learning, Agent Programming.

1 Introduction

High-level programming of mobile robots in complex, dynamic, partially observable and stochastic environments, like the ones provided by the RoboCup soccer competitions, is still a fundamental step for effective cognitive robots. This is an extremely difficult task, because the human developer is not able to predict the many possible situations that the robot will encounter, thus limiting its adaptivity to the real environment.

Machine learning techniques can help to create more robust behaviors and reduce the need for defining complex models of the systems in use. RoboCup teams have successfully experimented many different kinds of machine learning methods, both for real soccer robots (e.g., four legged AIBOs [3]) and for multi-agent systems [16]. Most of the approaches applied on real robots, however, target specific sub-tasks, like object recognition and walk optimization. The use of learning for improving high-level robot programming is mostly limited to

simulated systems, where the difficulties about sensor noises, non-deterministic effects of actions, time and resource constraints and real-time requirements are limited. Learning approaches have still many limitations due to the difficulties for the designer to shape the search space, even when his knowledge could reduce the learning burden significantly. These approaches do not traditionally scale well with the complexity of the task.

In this paper, we present a framework that integrates high-level robot programming and Reinforcement Learning (RL). With this framework it is possible to both design a partially specified program (or plan) for describing a complex task of a mobile robot and run reinforcement learning to improve its performance from experience. Hence, the designer can freely decide at which level learning can improve a partially specified plan, without the need of providing a full model of a complex domain, which will introduce many sources of uncertainties. We combine an approach we formerly proposed [7] with a high-level robot programming language, casting the learning problem in the specific stochastic processes that such a language can generate. Among the many available languages we have chosen Petri Net Plans (PNP), because of its high expressiveness and its successful use in many robotic applications including multi-robot systems [19]. This choice allows for naturally extending the proposed approach to multi-robot learning. As for the learning framework, we use Reinforcement Learning (RL), which is nowadays a strongly established field presenting many remarkable results. RL researchers have faced the problem of computing a *policy* from experience with or without a model of the environment, and RL techniques have been used for heuristic improvement and to plan in both state and plan spaces. Controllable stochastic processes can effectively model the uncertainty in the domain, but it must also cope with the *curse of dimensionality* for whatever the input space it always requires a form of function approximation.

The novelty of the framework presented in this paper is in the application of a recent approach to apply RL theory and methods in a novel learning state space. Combining such an approach with Petri Net Plans provides a tool to compactly represent and learn complex concurrent plans with interrupts and sensing. The search space is considerably smaller with respect to standard formulations, thus it does not require any function approximation and it scales well with the complexity of the domain. We provide the source code of the PNP executor and the learning algorithms at <http://pnp.dis.uniroma1.it>

To show the effectiveness of our solution we modeled several behaviors both in the *Keepaway* domain [15,6] and on NAO humanoid soccer robots. We chose the *Keepaway* domain because it is a well-known benchmark for learning algorithms at the edge of what RL can currently face, that also allows for a direct comparison with previous results. We are testing the system also on NAO humanoid robots to demonstrate the feasibility of using the method on real soccer robots.

2 Related Work

Petri Nets have a well established tradition as a modeling tool, and the use of stochastic transitions together with machine learning has been previously

suggested [18,8]. In this paper, we limit the search to deterministic policies and don't make use of probabilities directly in the model. Differently from the previous approaches we associate a value function to the marking of the Petri Net and try to estimate the value for the optimal policy defined over specific non-deterministic choice points. The work most related to our approach lies in the field of Hierarchical Reinforcement Learning (HRL). The overall idea of HRL is the ability of expressing constraints on the behavior of the agent so that knowledge about the task and the environment can be exploited to shrink the search space. The optimal policy in this setting is the best one among those compatible with the constraints. The approaches closest to ours are Parr and Russell's HAM [1] and Andre and Russell's ALisp [9]. A similar approach can also be found in the field of symbolic agent programming, as this is the case of Decision Theoretic Golog (DTGolog) [2]. All of the mentioned works allow to partially define the agent behavior in a high-level language (hierarchical state machines, Lisp and Golog respectively) and learn (or compute, in the case of DTGolog) the best behavior when this is not specified. Instead of a Lisp program or a state machine we exploit the expressive capabilities of Petri Nets and don't require the existence of an underlying Markovian process. This allows us to account for partial observability applying RL to a structure (the plan) that can do anything between retaining full memory of the previous choices (unwinding in a tree), or be completely reactive to the observations. Since the reward and the dynamics of the network (through the definition of conditions) depend on the environment and so on hidden variables the process might not be Markovian.

Applying direct RL to Non-Markovian processes has been considered by Pendrith and McGarity [11], Singh [14], and Perkins [13,12] but is still a subject that requires further investigation. More Recently Crook has studied the topic in conjunction with active perceptions [4] and has underlined the importance of the exploration policy in his work on consistent exploration [5].

3 Background

Our work is based on the combination of Petri Net Plans with a Reinforcement Learning algorithm in order to make learning possible and effective in a small state space related to the plan. Before we elaborate on the formal definition of the controllable stochastic process to which learning is applied, we briefly recall the plan representation formalism of Petri Net Plans.

3.1 Petri Net Plans

Petri Net Plans (PNP) [19] are a compact way to represent reactive, conditional, parallel plans. The formalism is built on top of Petri Nets (PN) [10], an established tool for modeling systems' behavior.

A Petri Net is a bipartite graph, in which nodes can be either *places* or *transitions*. Places have an integer number associated to them which represents their state and is said to be the number of *tokens* in that place. The number of

tokens in a place is also called the *marking* of that place, while the marking of the network is the vector of the numbers of tokens of each place in the network.

Petri Net Plans are particular Petri Nets whose operational semantics is enriched with the use of conditions verified at run-time over an external Knowledge Base. No restriction is imposed on the KB which is supposed to be updated by other modules according to the agent's perceptions.

The basic actions of a PNP are: *ordinary actions*, which represents a durative action; and *sensing actions*, representing procedures whose outcomes reflect the truth of one or more conditions actually implementing an if-statement.

A Petri Net Plan is the composition of the aforementioned basic actions by means of a set of operators such as *sequence*, *loop*, *fork* and *join*, which have the usual meaning of the corresponding operators for programming languages. There are also operators for multi-agent plans but in this paper we limit ourselves to single-agent plans. If a PN is a PNP then it is *safe*, that is the number of tokens in each place is at any moment in $\{0, 1\}$, and each token models the status of a thread in execution. Therefore, the maximum number of states (*markings*) in a Petri Net Plan with $|P|$ places is $2^{|P|}$ and although there is always a state machine equivalent to any PNP, Petri Net Plans can be exponentially more compact than state machines. This property turns out to be particularly relevant when parallelism among actions would induce a number of states that makes agent programming extremely cumbersome to the designer.

In addition to the basic actions and operators defined in [19], in this paper we also make use of *non-deterministic choices*, that are structures similar to sensing actions, but not labeled with conditions. Therefore, when the transitions of a non-deterministic choice are enabled, the path chosen by the executor is not determined by the operational semantics, thus leaving the possibility of making an informed choice based on past experience. An example of non-deterministic choice is reported in Figure 1. In the following we will refer to this structure also as *choice operator*.

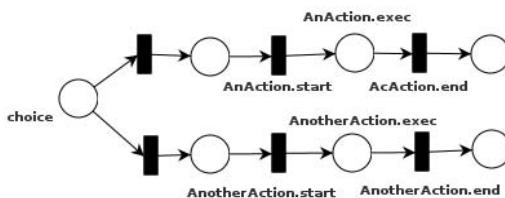


Fig. 1. An example of a simple choice point between two ordinary actions

Summarizing, a Petri Net Plan is formally a tuple

$$PNP = \langle P, T, W, M_0, G \rangle$$

in which $\langle P, T, W, M_0 \rangle$ is a Petri Net, and where:

- P is a set of *places* that represent the execution phases of actions. Each action is described by three places: the *initiation*, *execution* and *termination* place. Those places represent respectively the time before the action starts, during its execution and after the action terminates.
- T is a set of transitions that represent events and are grouped in different categories: action starting transitions, actions terminating transitions, interrupts and control transitions (which are part of an operator). Transitions may be labeled by conditions to control their firing.
- $W(p_i, t_j) \in \{0, 1\}$ and $W(t_j, p_i) \in \{0, 1\}$ for each $p_i \in P$ and $t_j \in T$ is the weight function. In PNP an arc either does not exist (zero weight) or has weight one.
- G is a set of *goal* markings reaching when the plan terminates. It must be a proper subset of the possible markings a PNP can be in.

We report here the definitions of a *possible* and an *executable* transition making the role of time more explicit with respect to their original formulation. We consider time in discrete timesteps and actions can take multiple timesteps to complete. We use the following notation:

- t_k : the time of occurrence of the k^{th} marking transition. By convention we denote $t_0 = 0$
- $M_k = M(t_k)$ where $M(t) = M_k$ for $t_k \leq t \leq t_{k+1}$ is the marking at time t
- $KB_k = KB(t_k)$ where $KB(t) = KB_k$ for $t_k \leq t \leq t_{k+1}$ is the state of the Knowledge Base at time t

Definition 1. Possible transition in a PNP. *Given two markings M_i , M_{i+1} , a transition from M_i to M_{i+1} is possible iff $\exists t \in T$ such that (i) $\forall p' \in P$ s.t $W(p', t) = 1$ then $M_i(p') = 1$; (ii) $M_{i+1}(p') = M_i(p') - 1 \forall p' \in P$ s.t. $W(p', t) = 1$; (iii) $M_{i+1}(p'') = 1 \forall p'' \in P$ s.t. $W(t, p'') = 1$.*

Definition 2. Executable transition in a PNP. *Given two markings M_i , M_{i+1} and a Knowledge Base K_i at time t_i , a transition from M_i to M_{i+1} is executable iff $\exists t \in T$, such that a transition from M_i to M_{i+1} is possible and the event condition ϕ labeling the transition t (denoted with $t.\phi$) holds in K_i (i.e. $K_i \models \phi$).*

4 Learning in Petri Net Plans

The central idea of learning in PNP is considering the stochastic process over markings that derives from the semantics of a Petri Net Plan, and learn how to control it in those choice points in which the behavior is partially specified. Before moving on and considering what algorithms can be applied to this problem we first define and analyze such a stochastic process to clarify what we are actually controlling.

4.1 Learning Problem Definition

Given a $PNP = \langle P, T, W, M_0, G \rangle$ we define a decision process

$$DP = \langle S, A, Tr, r \rangle$$

where:

- $S = \{M_i\}$ is the set of reachable markings executing a sequence of possible transitions from T . This is the state space of our controllable process.
- A is the set of actions. We define an action for each transition introduced by a choice operator, plus one *unnamed* action to account for all the other transitions that there is no need for the controller to distinguish.
- $Tr(s', a, \tau, s) = Pr(t_{k+1} - t_k \leq \tau, s_{k+1} = s' | s_k = s, a_k = a)$ is the probability for the action a to take τ time steps to complete, and to reach state s' from state s . This probability is defined to be 0 unless the transition corresponding to a is *possible* in the marking corresponding to s and when it fires it transforms such a marking in the marking corresponding to s' . In this latter case, the probability is in general unknown (which accounts for the necessity of learning, otherwise the problem would be solvable by dynamic programming).
- $r(s', a, k, s_t)$ is the reward function. We define it to be $r(s', a, k, s_t) = \sum_{i=1}^k \gamma^{i-1} r_{t+i}$ where the r_{t+i} are the instantaneous rewards collected during the action execution, and γ such that $0 \leq \gamma \leq 1$ is the discount factor. Instantaneous rewards are defined over perceptions, that is they are a function of the state of the knowledge base.

Notice that the states for which we learn a policy are a small subset of the reachable markings, being only the markings that make *possible* the few transitions used in choice operators. Both the transition and the reward function depend on the chosen action which is derived from a transition of a Petri Net Plan. If such a transition is not labeled, there is nothing that prevents it from firing when it becomes *possible* and its probability to complete in one time step is 1. On the other hand, if the transition is labeled by a condition the probability of that condition to be true at a specific time step depends on the state of the Knowledge Base and of the environment. The reward might thus depend on the hidden variables of the environment that cause (through the conditions) the transitions to fire. The dependency of the reward from a hidden state can make the process non-Markovian.

In order to define a decision problem, we establish a performance criterion that the controller of the stochastic decision process tries to maximize. As such, we consider the expected discounted cumulative reward, defined for a stochastic policy $\pi(s, a)$ and for all $s \in S$ and $a \in A$ as:

$$\begin{aligned}
 Q_\pi(s, a) &= E\left\{\sum_{i=1}^{\infty} \gamma^{i-1} r_i\right\} \\
 &= \sum_{s' \in S} \sum_{\tau=0}^{\infty} \pi(s, a) Tr(s', a, \tau, s) \left(r(s', a, \tau, s) + \right. \\
 &\quad \left. + \gamma^\tau \sum_{a' \in A(s')} \pi(s', a') Q_\pi(s', a') \right) \tag{1}
 \end{aligned}$$

where $A(s)$ is the set of actions for which $Tr(s', a, \tau, s) > 0$ for some value of τ .

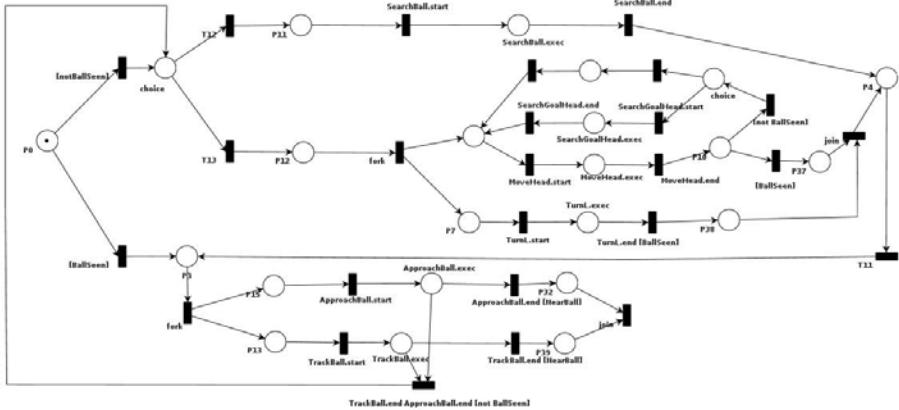


Fig. 2. The PNP for a robot in the RoboCup Standard Platform League

4.2 Learning Algorithms

Since the stochastic process is in general non Markovian we preferred a first-vist MonteCarlo algorithm [17] to TD methods. An alternative to yet be explored in this context might be Perkins's MCESP [13] or policy search methods. The exploitation policy proved to be fundamental in our experiments, confirming previous results available in the literature. Consistent exploration [5], in particular, dramatically improved the reliability of the estimation of the expected reward for a specific policy. In this paper we focus on the modeling aspect of the learning problem and this leaves little room for an extended experimental evaluation. Nevertheless, with respect to our preliminary evaluation [7] some recent result suggests new directions for future work and will be discussed in the following section.

5 Modeling

In the following we show two examples of how to model agent behaviors and learn with Petri Net Plans.

5.1 RoboCup Standard Platform League

The Standard Platform League lets us show an important feature of our method that is the ability to learn which *combination* of actions performs better on the long term, over several plan branches. In Figure 2 we show a portion of a plan drawn from the SPL.

The upper part of the figure represents the behavior of the agent when it does not see the ball. The agent has a choice between two possible branches: *SearchBall* which is a predefined procedure that moves the head and the body together; and *MoveHead* plus *TurnLeft* as an alternative. This second branch has a fork that creates two threads, one for the legs (*TurnLeft*) and one for the head

(*MoveHead*). For what concerns the head, the designer was not sure whether interleaving a look at the goal (to help the localization) and the movement of the head to search for the ball was more convenient than just the latter. Therefore, another choice point is added only on the head branch that will learn the best behavior while the legs are turning left. Recall that the system discerns among *markings*, so the alternative in this last choice point are between the marking with a token in TurnLeft and one in MoveHead and the marking with a token in TurnLeft and one in the empty action that just returns to the initial place of the branch. When the agent sees the ball the two threads join and the rightmost place collects the token when it comes either from SearchBall or from the join. The execution then moves to the lower part of the network. Here ApproachBall and TrackBall are executed at the same time. In the case in which the robot loses the ball an interrupt is issued, both actions are terminated and the control goes back to the place where the ball has to be searched. Otherwise, when the agent is close to the ball the two threads join. Beyond the join the would be the part of the plan related with the kick with another choice point but we took that part off the picture for simplicity.

The implementation on the NAO robots will provide full evidence of the effectiveness of the proposed framework. While a quantitative analysis about learning on NAO robots is still under progress, we have verified in simulation the capability of the system to learn any probability distribution associated to the non-deterministic effects of the actions in the plan and consequently the best actions to perform.

5.2 Keepaway Soccer

Keepaway is a domain proposed by Stone and Sutton [15] as a benchmark for Reinforcement Learning. It is a subtask of RoboCup Soccer in which one team, the *keepers*, must keep possession of the ball in a limited region as long as possible while another team, the *takers*, tries to gain possession. The task is *episodic* and one episode ends whenever the takers manage to catch the ball or the ball leaves the region. We conducted our studies on the 3 vs 2 task, i.e. with three keepers and two takers.

In our work, we focus on the keepers and leave the takers' behavior to their predefined policy that consists in both of them following the ball. We refer to Stone et al. [15] and especially to the more recent work by Kalyanakrishnan and Stone [6] as representatives of the “RL way” to face Keepaway Soccer and we show our methodology applied to this task.

The first step is devising a proper set of actions. We consider three actions for the agent closer to the ball and three for the other two agents. The actions available to an agent close enough to kick are *holdBall* that just keeps possession of the ball, *passToCloser* that passes the ball to the agent that is closer to the kicker and *passToFree* that passes the ball to the agent whose line of pass is further from the takers. The first action is clearly wrong since a player cannot hold the ball indefinitely without being reached by the takers but we added it as a control, to make sure that our algorithm assigns the correct value to it and never chooses

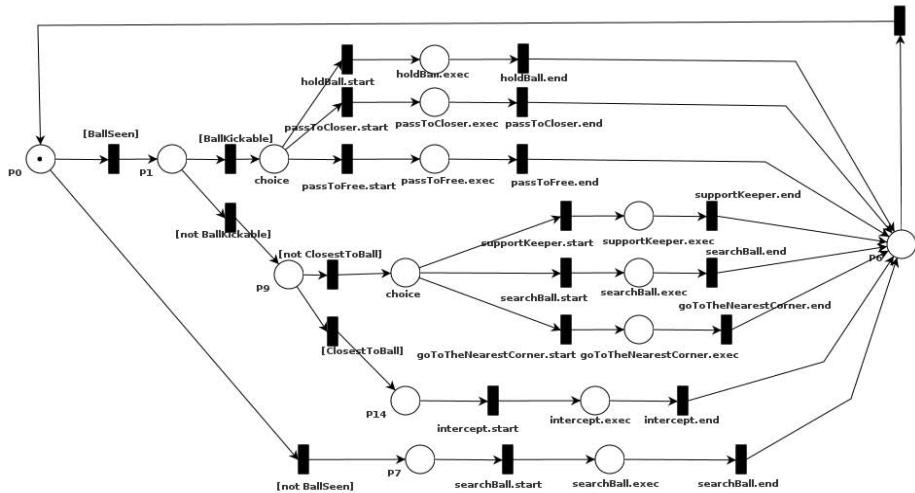


Fig. 3. The PNP for a keeper with choice points on the passing and positioning behaviors

that action after convergence. The actions available to the agents far from the ball are *searchForBall* that just turns in place, *getOpen* that is the default hand-coded behavior provided by the framework described as “move to a position that is free from opponents and open for a pass from the ball’s current position”, and *goToTheNearestCorner* that goes to the corner closer to the agent.

After the definition of the available actions we create a PNP to accommodate our choice points. The entire PNP used in these experiments is shown in Figure 3. We implemented as a PNP the plan used in our previous work [7] but we provide here an experimental evaluation of a different setting, and show how the value function changes at the choice points. We had previously obtained an average hold time of 31 seconds, but the Keepaway implementation turned out to be quite dependent on the system and the same behavior in these experiments holds the ball for an average of 25 seconds. The actual value is not the focus of the experiments that are rather aimed at providing an initial analysis of the evolution of the value function on the non-Markovian system, but we keep this value as a baseline.

The leftmost node is the initial state, the control flows from left to right and it reaches the rightmost node within a simulation server cycle. In each cycle the agent must send a command to the server, thus performing an action, therefore every path from the leftmost to the rightmost nodes contains exactly one action.

As mentioned before these experiments were conducted using first-visit MonteCarlo with $\alpha = \max(1/n(m), 0.02)$ and $\gamma = 1$ where $n(m)$ is the number of episodes in which the marking m has been visited. Due to the consistent exploration the choice is made only the first time that a choice point is hit and is remembered till the end of the episode. As with the exploration policy we used ϵ -greedy decreasing ϵ linearly from 1 to 0 in 300 episodes (after which the policy

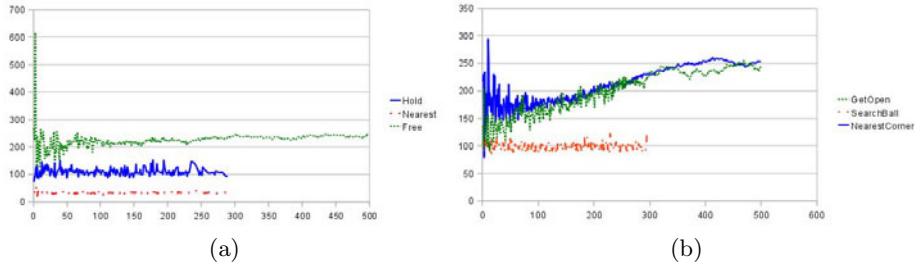


Fig. 4. Value function at choice points. (a) is the value for the choice when the agent is close to the ball while (b) when it is far from it. The x axis is the number of episodes in the run while y axis is the hold time in tenths of seconds.

just exploits). The reward signal is given by the duration of the episode: at every server cycle each agent receives a reward of $1/10$ of second. The values of the three choices in both choice points are plotted in figure 4 (a) and (b), averaged over 25 runs. The optimal policy consists in choosing *passToFree* and *goToTheNearest-Corner* whose values can be seen to converge to about 25 seconds. While at the first choice point (a) the value of *passToFree* clearly outperforms its alternatives from the very first experiments, at the second choice point (b) some times (4 out of 25) the algorithm converged to *getOpen* which on average performs slightly worse than *goToTheNearestCorner*. Despite this can happen in a non-Markovian setting, notice how close the two policy actually are, so the loss in performance is not really significant. It is still to investigate whether a slower decrease of ϵ might help in this respect. Having ϵ decreasing is of the utmost importance since the method is on-policy, and it would just converge to the value of the $(\epsilon-)$ random behavior which in this case can be arbitrarily worse than the optimal one.

Besides the considerations on the convergence of the usual RL algorithms, notice what a good estimate of the value function can be obtained in the first hundred of episodes which is orders of magnitude fewer than the results in the literature. Of course we are not solving the full RL problem, but we are limiting the search in the tiny policy space left by the plan specification. If on the one hand this precludes the convergence to the global optimal policy, which is unlikely to be admitted by the constraints of the plan, on the other hand the designer can easily cooperate with the learning algorithm, being able to improve the plan using an expressive high-level formalism he can easily access.

As a further example of a behavior not easily specifiable with traditional methods consider the case in which, right after having passed the ball, the agent is able to move to the corner not occupied by any teammate. The designer might want the algorithm to learn whether going to the corner is actually convenient, resulting in the PNP in figure 5.

In addition to the new choice point, the figure also shows (in blue) an *interrupt*. If during *goToFreeCorner* the agent becomes the closest to the ball, it stops moving to the corner and executes the part of the plan that pertains to that situation. Adding this type of time constraints (*goToFreeCorner* must happen after a kick) to traditional RL would expand the state space even further.

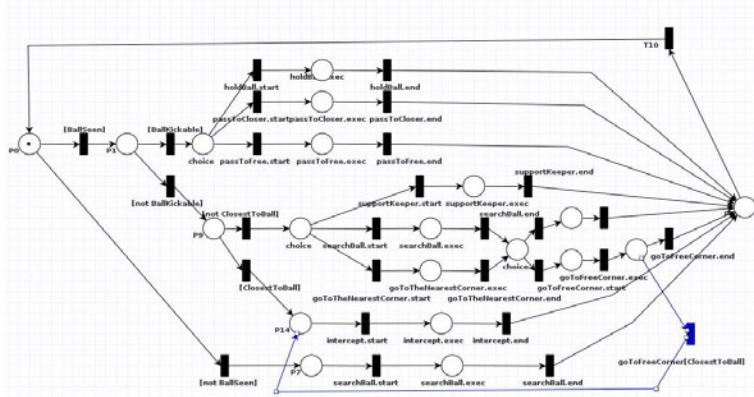


Fig. 5. The PNP with a new choice point after the kick. In blue is shown the interrupt.

6 Conclusions

In this paper we have presented a framework for learning complex high-level robot behaviors, based on integrating a high-level programming language PNP with reinforcement learning.

To the best of our knowledge no other system can effectively learn parallel plans on continuous domains. This is thanks to the fact that in this formalism the designer can shrink the search space providing only the choices he considers reasonable in different parts of the plan, and restricting the attention to the environment variables through the use of appropriate conditions. The solution found in such a state space is the optimal one among those compliant with the constraints imposed by the plan on the available policies. Such a solution could be not the optimal one in general, when the whole state space of both the plan and the environment is considered. Nonetheless, even if RL is supposed to compute the optimal policy on complex domains it fails to do so in a time of practical interest (as we have shown on keepaway where our policy outperformed the policy learned with traditional RL). In those cases, agent programming is usually performed by developers themselves at the level of plans and being able to apply learning algorithms at this level can prove to be a powerful tool for autonomous agents.

References

1. Andre, D., Russell, S.: Programmable reinforcement learning agents. In: Advances in Neural Information Processing Systems, pp. 1019–1025 (2001)
2. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: Decision-theoretic, high-level agent programming in the situation calculus. In: Proceedings of the National Conference on Artificial Intelligence, pp. 355–362. AAAI Press / The MIT Press (2000)
3. Chalup, S., Murch, C., Quinlan, M.: Machine learning with AIBO robots in the four-legged league of robocup. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(3), 297–310 (2007)

4. Crook, P., Hayes, G.: Learning in a state of confusion: Perceptual aliasing in grid world navigation. *Towards Intelligent Mobile Robots* 4 (2003)
5. Crook, P.A.: Consistent exploration improves convergence of reinforcement learning on pomdps (2007)
6. Kalyanakrishnan, S., Stone, P.: Learning complementary multiagent behaviors: A case study. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) *RoboCup 2009. LNCS*, vol. 5949, pp. 153–165. Springer, Heidelberg (2010)
7. Leonetti, M., Iocchi, L.: Improving the Performance of Complex Agent Plans Through Reinforcement Learning. In: *Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (2010)
8. Lima, P.U., Saridis, G.N.: Design of Intelligent Control Systems Based on Hierarchical Stochastic Automata. World Scientific Publ., Singapore (1996)
9. Marthi, B., Russell, S.J., Latham, D., Guestrin, C.: Concurrent hierarchical reinforcement learning. In: Kaelbling, L.P., Saffiotti, A. (eds.) *IJCAI*, pp. 779–785. Professional Book Center (2005)
10. Murata, T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4), 541–580 (1989)
11. Pendrith, M.D., McGarity, M.: An analysis of direct reinforcement learning in non-markovian domains. In: Shavlik, J.W. (ed.) *ICML*, pp. 421–429. Morgan Kaufmann, San Francisco (1998)
12. Perkins, T., Pendrith, M.: On the existence of fixed points for Q-learning and Sarsa in partially observable domains. In: *Proceedings of the Nineteenth International Conference on Machine Learning*, p. 497. Morgan Kaufmann Publishers Inc., San Francisco (2002)
13. Perkins, T.J.: Reinforcement learning for pomdps based on action values and stochastic optimization. In: *Eighteenth National Conference on Artificial Intelligence*, Menlo Park, CA, USA, pp. 199–204. American Association for Artificial Intelligence (2002)
14. Singh, S., Jaakkola, T., Jordan, M.: Learning without state-estimation in partially observable Markovian decision processes. In: *Proc. of 11th International Conference on Machine Learning* (1994)
15. Stone, P., Sutton, R.S., Kuhlmann, G.: Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior* 13(3), 165–188 (2005)
16. Stone, P., Veloso, M.: Multiagent systems: A survey from a machine learning perspective. *Autonomous Robotics* 8(3) (July 2000)
17. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (1998)
18. Wang, F.-Y., Saridis, G.N.: Task translation and integration specification in Intelligent Machines. *IEEE Transactions on Robotics and Automation RA-9(3)*, 257–271 (1993)
19. Ziparo, V.A., Iocchi, L., Nardi, D., Palamara, P.F., Costelha, H.: Petri net plans: a formal model for representation and execution of multi-robot plans. In: *AAMAS 2008: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, Richland, SC, pp. 79–86. International Foundation for Autonomous Agents and Multiagent Systems (2008)

Author Index

- Abreu, Pedro 242
Aamir, Ahmad 350
Akin, H. Levent 72, 362, 409
Andriluka, Mykhaylo 180

Behnke, Sven 97, 121, 157, 374, 397
Burchardt, Armin 145

Castelão, Daniel 242
Correa, Mauricio 13
Costa, Israel 242
Czarnetzki, Stefan 48, 266

DallaLibera, Fabio 218
Danış, F. Serhan 409
Decker, Peter 169
Dijkshoorn, Nick 336
Dillenberger, Denis 303
Droeschel, David 121

Eckel, Gerhard 133
Edalat, Mohammad Ehsan 206

Ferrein, Alexander 133
Formsma, Okke 336
Frias, Diego 82
Fujii, Takashi 60

Gabel, Thomas 36
Garganta, Júlio 242
Gökçe, Barış 362
Gossow, David 169, 303, 314

Hausknecht, Matthew 254
Hegele, Maximilian 48
Hermosilla, Gabriel 25
Holz, Dirk 121

Ikemoto, Shuhei 218
Iocchi, Luca 278, 418
Ishiguro, Hiroshi 218

Kellner, Tobias 133
Kerner, Sören 48, 266
Kheirikhah, Mohammad Mahdi 206

Klingauf, Uwe 180
Kohlbrecher, Stefan 180
Kruse, Michael 266

Lau, Nuno 324
Laue, Tim 1, 109, 145
Leonetti, Matteo 418
Lima, Pedro U. 350
Liu, Weichao 397
Loncomilla, Patricio 25
Lu, Huimin 291

Marchetti, Luca 278
Marino, Francesco Antonio 278
Masutani, Yasuhiro 60
Menegatti, Emanuele 218
Meriçli, Çetin 194, 409
Meriçli, Tekin 409
Meyer, Johannes 180
Minato, Takashi 218
Missura, Marcell 97, 374
Mitsunaga, Noriaki 60
Müller, Judith 109

Nagasaki, Yasunori 60
Naito, Osamu 60
Nakagawa, Yukiko 60
Naruse, Tadashi 60
Neuhaus, Frank 303
Niemüller, Tim 133
Noort, Sander van 336

Oliveira, Ricardo 350
Özkucur, N. Ergin 72

Pagello, Enrico 218
Parra-Tsunekawa, S. Isao 13
Paulus, Dietrich 169, 303, 314
Pellenz, Johannes 303
Petersen, Karen 180, 386
Pimentel, Fagner de Assis Moura 82
Pirro, David 133
Podbregar, Patrick 133

Rabiee, Samaneh 206
Randelli, Gabriele 278

- Rath, Christof 133
Reis, Luís Paulo 242, 324
Riedmiller, Martin 36
Röfer, Thomas 1, 109, 145
Roth, Stefan 180
Ruiz-del-Solar, Javier 13, 25
- Santos, João 350
Santos, Pedro 350
Schiele, Bernt 180
Schmitz, Andreas 97
Schnabel, Ruwen 121
Schnitzspan, Paul 180
Schulz, Hannes 397
Schwahn, Oliver 180
Seekircher, Andreas 1
Seib, Viktor 314
Shafii, Nima 324
Simões, Marco A.C. 82
- Souza, Josemar Rodrigues de 82
Steinbauer, Gerald 133
Stoll, Georg 386
Stone, Peter 254
Stryk, Oskar von 180, 386
Stückler, Jörg 121, 157, 397
- Veloso, Manuela 194
Verschae, Rodrigo 13
Vetter, Sebastian 314
Visser, Arnoud 336
Visser, Ubbo 230
- Watanabe, Masato 60
Wilken, Tobias 374
- Zhang, Hui 291
Zheng, Zhiqiang 291