

Lecture Notes in Artificial Intelligence 2019

Subseries of Lecture Notes in Computer Science

Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Peter Stone Tucker Balch
Gerhard Kraetzschmar (Eds.)

RoboCup 2000: Robot Soccer World Cup IV



Springer

Series Editors

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

Volume Editors

Peter Stone
AT & Labs - Research
180 Park Ave., room A273, Florham Park, NJ 07932, USA
E-mail: pstone@research.att.com

Tucker Balch
Carnegie Mellon University
The Robotics Institute
5000 Forbes Avenue, Pittsburgh, PA 15213-3891, USA
E-mail: trb@ri.cmu.edu

Gerhard Kraetzschmar
University of Ulm
Neural Information Processing Department
Oberer Eselsberg, 89069 Ulm, Germany
E-mail: gkk@neuro.informatik.uni-ulm.de

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

RoboCup <4, 2000, Melbourne>
Robot Soccer World Cup IV / RoboCup 2000. Peter Stone ... (ed.). - Berlin ;
Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ;
Singapore ; Tokyo : Springer, 2001

(Lecture notes in computer science ; Vol. 2019 : Lecture notes in
artificial intelligence)
ISBN 3-540-42185-8

CR Subject Classification (1998): I.2, C.2.4, D.2.7, H.5, I.5.4, I.6, J.4

ISBN 3-540-42185-8 Springer-Verlag Berlin Heidelberg New York

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-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author
Printed on acid-free paper SPIN 10782256 06/3142 5 4 3 2 1 0

Preface

RoboCup 2000, the Fourth Robot World Cup Soccer Games and Conferences, was held from August 27th to September 3rd, 2000, at the Melbourne Exhibition Center in Melbourne, Australia. Like the previous international RoboCup events – RoboCup 97 in Nagoya, Japan; RoboCup 98 in Paris, France; and RoboCup 99 in Stockholm, Sweden – RoboCup 2000 included a technical workshop as well as several robotic soccer competitions. RoboCup 2000 introduced the first RoboCup Jr. competition for children, as well as demonstrations of humanoid robots and of the RoboCup-Rescue disaster rescue simulator.

This book documents RoboCup 2000. It consists of (i) an overview; (ii) championship papers by the winners of the competitions; (iii) the finalist papers for the RoboCup challenge awards; (iv) the papers and posters presented at the workshop; and (v) descriptions of the teams that competed.

The book begins with an overview article introducing the competitions and demonstrations and including the scores of all of the games in the four competition leagues: the simulation league, the small-size robot (F180) league, the middle-size robot (F2000) league, and the Sony legged robot league. The following section presents the championship papers from the winners of these leagues.

The RoboCup challenge awards are distinctions that are given annually to the RoboCup-related researchers with the greatest potential to advance their respective fields. In RoboCup 2000, the challenge award finalists were selected from among the workshop papers. The four selected finalist papers appear in the next section.

The annual RoboCup workshop provides a forum for RoboCup researchers to exchange ideas that are generally applicable across the different RoboCup leagues and/or that are of general scientific interest. The RoboCup 2000 workshop received more than 60 submissions, from which 20 were selected for full presentation and an additional 20 were selected for poster presentation. These research papers form the main body of this book.

The book concludes with descriptions of most of the more than 80 teams that competed in RoboCup 2000. These team descriptions serve to catalog the full range of researchers and approaches that have been applied to the challenges put forth by RoboCup.

The next international RoboCup events will be held in Seattle, USA (2001) and in Fukuoka, Japan (2002). In addition to all existing RoboCup events, they are scheduled to introduce (i) RoboCup-Rescue disaster rescue competitions for the transfer of ideas and techniques developed in the soccer domain to a related task, and (ii) a humanoid robot competition as a step towards the long-term goal of creating a full team of humanoid robots that compete on a real soccer field. We look forward to continuing research innovations and exciting demonstrations of robotics and AI technology in these and other future RoboCup events.

RoboCup Federation

The RoboCup Federation, the governing body of RoboCup, is an international organization that promotes science and technology using soccer games by robots and software agents.

President:

Hiroaki Kitano, Japan Science and Technology Corporation, Japan

Vice-Presidents:

Minoru Asada, Osaka University, Japan

Enrico Pagello, University of Padua, Italy

Manuela Veloso, Carnegie Mellon University, USA

Trustees:

Hans-Dieter Burkhardt, Humboldt University, Germany

Silvia Coradeschi, Orebro University, Sweden

Dominique Duhaut, University of Southern Bretagne, France

Frans Groen, University of Amsterdam, The Netherlands

Andrew Jennings, Royal Melbourne Institute of Technology, Australia

Milind Tambe, University of Southern California, USA

Executive Committee:

Tucker Balch, Carnegie Mellon University, USA

Andreas Birk, Free University of Brussels, Belgium

Masahiro Fujita, Sony Corp., Japan

Gerhard Kraetzschmar, University of Ulm, Germany

Paul Levi, University of Stuttgart, Germany

Pedro Lima, ISR/IST, Technical University of Lisbon, Portugal

Henrik Lund, University of Southern Denmark, Denmark

Daniele Nardi, University of Rome “La Sapienza”, Italy

Itsuki Noda, ETL, Japan

Daniel Polani, Lübeck University, Germany

Raúl Rojas, University of Berlin, Germany

Peter Stone, AT&T Labs – Research, USA

Satoshi Tadokoro, Kobe University, Japan

Gordon Wyeth, University of Queensland, Australia

Advisory Committee:

Luigia Carlucci Aiello, Universitá di Roma “La Sapienza”, Italy

Daniel Bobrow, XEROX PARC, USA

Michael Brady, Oxford University, UK

Toshio Fukuda, Nagoya University, Japan

Michael P. Georgeff, Australian AI Institute, Australia

Alan Mackworth, University of British Columbia, Canada

David Waltz, NEC Research, USA

Wolfgang Wahlster, DFKI, Germany

RoboCup 2000 Organization and Support

General Chair:

Andrew Jennings, Royal Melbourne Institute of Technology, Australia

Deputy General Chair:

Jeremy Samuel, Australian AI Institute, Australia

Organizing Co-chairs:

Enrico Pagello, University of Padua, Italy

Manuela Veloso, Carnegie Mellon University, USA

Minoru Asada, Osaka University, Japan

Simulator League Committee:

Paul Scerri (chair), Linköping University, Sweden

Tomoichi Takahashi, Chubu University, Japan

Gal Kaminka, University of Southern California, USA

Mikhail Prokopenko, CSIRO, Australia

Ryszard Kowalczyk, CSIRO, Australia

Small-Size Robot League (F180) Committee:

Gordon Wyeth (chair), University of Queensland, Australia

Raúl Rojas, Free University of Berlin, Germany

Yuki Nakagawa, Kitano Symbiotic Systems, Japan

Andrew Howard, University of Melbourne, Australia

Middle-Size Robot League (F2000) Committee:

Steffen Gutmann (chair), University of Freiburg, Germany

Yong Fook Seng, Ngee Ann Polytechnic, Singapore

Wei-Min Shen, University of Southern California, USA

Paul Parisi, Australian AI Institute, Australia

Sony Legged Robot League Committee:

Masahiro Fujita (chair), Sony Corp., Japan

Minoru Asada, Osaka University, Japan

Manuela Veloso, Carnegie Mellon University, USA

Dominique Duhaut, University of Southern Bretagne, France

Workshop Program Committee:

Peter Stone, (co-chair), AT&T Labs – Research, USA

Tucker Balch (co-chair), Carnegie Mellon University, USA

Gerhard Kraetzschmar (co-chair), University of Ulm, Germany

Giovanni Adorni, University of Parma, Italy

David Andre, U.C. Berkeley, USA

Ron Arkin, Georgia Institute of Technology, USA

Andreas Birk, University of Brussels, Belgium

Andrea Bonarini, Politecnico di Milano, Italy

Michael Bowling, Carnegie Mellon University, USA

Ian Frank, Electrotechnical Laboratory, JAPAN

Doug Gage, Space and Naval Warfare Systems Center, USA

Wiebe van der Hoek, Utrecht University, The Netherlands

VIII Organization and Support

Huosheng Hu, University of Essex, UK
Kurt Konolige, SRI International, USA
Robin Murphy, University of South Florida, USA
Lynne Parker, Oak Ridge National Laboratory, USA
Jukka Riekki, University of Oulu, Finland
Wei-Min Shen, University of Southern California, USA
Sho'ji Suzuki, Osaka University, Japan
Masayuki Ohta, Tokyo Institute of Technology, Japan
Gordon Wyeth, University of Queensland, Australia

Supported by:

Fuji Xerox
Australon
RMIT University

RoboCup World Wide Sponsors:

Sony Corporation
SGI

Table of Contents

Overview of RoboCup-2000	1
<i>Edited by Peter Stone with Minoru Asada, Tucker Balch, Masahiro Fujita, Gerhard Kraetzschmar, Henrik Lund, Paul Scerri, Satoshi Tadokoro, and Gordon Wyeth</i>	
Champion Teams	
FC Portugal Team Description: RoboCup 2000 Simulation League Champion	29
<i>Luís Paulo Reis and Nuno Lau</i>	
The Cornell RoboCup Team.....	41
<i>Raffaello D'Andrea, Tamás Kalmár-Nagy, Pritam Ganguly, and Michael Babish</i>	
CS Freiburg: Doing the Right Thing in a Group	52
<i>Thilo Weigel, Willi Auerbach, Markus Dietl, Burkhard Dümler, Jens-Steffen Gutmann, Kornel Marko, Klaus Müller, Bernhard Nebel, Boris Szerbakowski, and Maximilian Thiel</i>	
The UNSW RoboCup 2000 Sony Legged League Team.....	64
<i>Bernhard Hengst, Darren Ibbotson, Son Bao Pham, John Dalgliesh, Mike Lawther, Phil Preston, and Claude Sammut</i>	
Challenge Award Finalists	
Adaptive Path Planner for Highly Dynamic Environments	76
<i>Jacky Baltes and Nicholas Hildreth</i>	
Communication and Coordination Among Heterogeneous Mid-Size Players: ART99	86
<i>Claudio Castelpietra, Luca Iocchi, Daniele Nardi, Maurizio Piaggio, Alessandro Scalzo, and Antonio Sgorbissa</i>	
A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor	96
<i>Carlos F. Marques and Pedro U. Lima</i>	
Behavior Classification with Self-Organizing Maps	108
<i>Michael Wünstel, Daniel Polani, Thomas Uthmann, and Jürgen Perl</i>	

Technical Papers

Full Papers

Flexible Synchronisation within RoboCup Environment: A Comparative Analysis	119
<i>Marc Butler, Mikhail Prokopenko, and Thomas Howard</i>	
Extended Q-Learning: Reinforcement Learning Using Self-Organized State Space	129
<i>Shuichi Enokida, Takeshi Ohasi, Takaichi Yoshida, and Toshiaki Ejima</i>	
And the Fans Are Going Wild! SIG plus MIKE	139
<i>Ian Frank, Kumiko Tanaka-Ishii, Hiroshi G. Okuno, Junichi Akita, Yukiko Nakagawa, K. Maeda, Kazuhiro Nakadai, and Hiroaki Kitano</i>	
Fast and Accurate Robot Vision for Vision Based Motion	149
<i>Pieter Jonker, Jurjen Caarls, and Wouter Bokhove</i>	
Design of RoboCup-Rescue Viewers – Towards a Real World Emergency System –	159
<i>Yoshitaka Kuwata and Atsushi Shinjoh</i>	
From Multiple Images to a Consistent View	169
<i>R. Hanek, T. Schmitt, M. Klupsch, and S. Buck</i>	
Omni-Directional Vision with a Multi-part Mirror	179
<i>Fabio M. Marchese and Domenico G. Sorrenti</i>	
Observation Strategy for Decision Making Based on Information Criterion	189
<i>Noriaki Mitsunaga and Minoru Asada</i>	
Towards a Logical Approach for Soccer Agents Engineering	199
<i>Jan Murray, Oliver Obst, and Frieder Stolzenburg</i>	
Bridging Gap between the Simulation and Robotics with a Global Vision System	209
<i>Yukiko Nakagawa, Hiroshi G. Okuno, and Hiroaki Kitano</i>	
Real-Time Estimating Spatial Configuration between Multiple Robots by Triangle and Enumeration Constraints	219
<i>T. Nakamura, M. Oohara, A. Ebina, M. Imai, T. Ogasawara, and H. Ishiguro</i>	
Framework of Distributed Simulation System for Multi-agent Environment	229
<i>Itsuki Noda</i>	

Robust Real Time Color Tracking	239
<i>Mark Simon, Sven Behnke, and Raúl Rojas</i>	

Reinforcement Learning for 3 vs. 2 Keepaway	249
<i>Peter Stone, Richard S. Sutton, and Satinder Singh</i>	

Fault-Tolerant Self Localization by Case-Based Reasoning	259
<i>Jan Wendler, Steffen Brüggert, Hans-Dieter Burkhard, and Helmut Myritz</i>	

PINO The Humanoid: A Basic Architecture	269
<i>Fuminori Yamasaki, Tatsuya Matsui, Takahiro Miyashita, and Hiroaki Kitano</i>	

Poster Descriptions

Team/Goal-Keeper Coordination in the RoboCup Mid-Size League	279
<i>Giovanni Adorni, Stefano Cagnoni, Monica Mordonini, and Maurizio Piaggio</i>	

RescueModel: A Multi-Agent Simulation of Bushfire Disaster Management	285
<i>Gary Au, Simon Goss, Clint Heinze, and Adrian R. Pearce</i>	

Vision-Based Localization in RoboCup Environments	291
<i>Stefan Enderle, Marcus Ritter, Dieter Fox, Stefan Sablatnög, Gerhard Kraetzschmar, and Günther Palm</i>	

Collaborative Emergent Actions between Real Soccer Robots	297
<i>M. Ferraresto, C. Ferrari, E. Pagello, R. Polesel, R. Rosati, A. Speranzon, and W. Zanette</i>	

The Statistics Proxy Server	303
<i>Ian Frank, Kumiko Tanaka-Ishii, Katsuto Arai, and Hitoshi Matsubara</i>	

Using Simulated RoboCup in Undergraduate Education	309
<i>Fredrik Heintz, Johan Kummeneje, and Paul Scerri</i>	

Path Planning of a Mobile Robot as a Discrete Optimization Problem and Adjustment of Weight Parameters in the Objective Function by Reinforcement Learning	315
<i>Harukazu Igarashi</i>	

Simulator Complex for RoboCup Rescue Simulation Project — As Test-Bed for Multi-Agent Organizational Behavior in Emergency Case of Large-Scale Disaster	321
<i>Toshiyuki Kaneda, Fumitoshi Matsuno, Hironao Takahashi, Takeshi Matsui, Masayasu Atsumi, Michinori Hatayama, Kenji Tayama, Ryousuke Chiba, and Kazunori Takeuchi</i>	

A Quadratic Programming Formulation of a Moving Ball Interception and Shooting Behaviour, and Its Application to Neural Network Control	327
<i>Frederic Maire and Doug Taylor</i>	
Keeping the Ball from CMUnited-99	333
<i>David McAllester and Peter Stone</i>	
Potential Tasks and Research Issues for Mobile Robots in RoboCup Rescue	339
<i>Robin R. Murphy, Jenn Casper, and Mark Micire</i>	
Potential Field Approach to Short Term Action Planning in RoboCup F180 League	345
<i>Yasunori Nagasaka, Kazuhito Murakami, Tadashi Naruse, Tomoichi Takahashi, and Yasuo Mori</i>	
RoboCup-Rescue Simulation: In Case of Fire Fighting Planning	351
<i>Masayuki Ohta, Tomoichi Takahashi, and Hiroaki Kitano</i>	
On Emergence of Scalable Tactical and Strategic Behavior	357
<i>Mikhail Prokopenko, Marc Butler, and Thomas Howard</i>	
Karlsruhe Brainstormers — A Reinforcement Learning Approach to Robotic Soccer	367
<i>M. Riedmiller, A. Merke, D. Meier, A. Hoffman, A. Sinner, O. Thaté, and R. Ehrmann</i>	
Preliminary Studies of Dynamics of Physical Agent Ecosystems	373
<i>Josep Lluís de la Rosa, Israel Muñoz, Bianca Innocenti, Albert Figueras, Miquel Montaner, and Josep Antoni Ramon</i>	
RoboCup Rescue Disaster Simulator Architecture	379
<i>Tomoichi Takahashi, Ikuo Takeuchi, Tetsuhiko Koto, Satoshi Tadokoro, and Itsuki Noda</i>	
Improvement Continuous Valued Q-learning and Its Application to Vision Guided Behavior Acquisition	385
<i>Yasutake Takahashi, Masanori Takeda, and Minoru Asada</i>	
Recognizing Formations in Opponent Teams	391
<i>Ubbo Visser, Christian Drücker, Sebastian Hübner, Esko Schmidt, and Hans-Georg Weland</i>	
Team Descriptions	
Simulation League	
SBC++ Simulator Team	397
<i>Amin Abbaspour, Mahmood Rahmani, Bahman Radjabalipour, and Eslam Nazemi</i>	

A Team	401
<i> Hidehisa Akiyama</i>	
AT Humboldt 2000 (Team Description)	405
<i> Hans-Dieter Burkhard, Joscha Bach, Kay Schröter, Jan Wendler, Michael Gollin, Thomas Meineert, and Gerd Sander</i>	
AIRG Sibiu	409
<i> Ciprian Canea and Marius Staicu</i>	
The NOAI Team Description	413
<i> Anneli Dahlström, Fredrik Heintz, Martin Jacobsson, Johan Thapper, and Martin Öberg</i>	
Improved Agents of the magmaFreiburg2000 Team	417
<i> Klaus Dorer</i>	
Virtual Werder	421
<i> Christian Drücker, Sebastian Hübner, Esko Schmidt, Ubbo Visser, and Hans-Georg Weland</i>	
Kakitsubata Team Description	425
<i> Tetsuya Esaki, Taku Sakushima, Shinji Futamase, Nobuhiro Ito, Tomoichi Takahashi, Wei Chen, and Koichi Wada</i>	
PaSo-Team 2000	429
<i> Carlo Ferrari, Francesco Garelli, and Enrico Pagello</i>	
Sharif-Arvand Simulation Team	433
<i> Jafar Habibi, Ehsan Chiniforooshan, Majid Khabbazian, Mahdi Mirzazade, MohammadAli Safari, and HamidReza Younesi</i>	
SharifII Soccer Simulation Team	437
<i> Jafar Habibi, Ehsan Foroughi, Mehran Motamed, Pooya Karimian, Hamed Hatami, and Hossein Fardad</i>	
Essex Wizards 2000 Team Description	441
<i> H. Hu, K. Kostiadis, M. Hunter, and N. Kalyviotis</i>	
RoboCup-2000 Simulation League: Team KU-Yam2000	445
<i> Harukazu Igarashi, Yusuke Yamauchi, and Shuichi Iidoi</i>	
Harmony Team	449
<i> Hiroyuki Iizuka, Masatoshi Hiramoto, Hidenori Kawamura, Masahito Yamamoto, and Azuma Ohuchi</i>	
TakAI	453
<i> Tetsuhiko Koto</i>	

XIV Table of Contents

PSI Team.	457
<i>Alexander N. Kozhushkin</i>	
Gnez: Adapting Knowledge to the Environment with GA	461
<i>Takenori Kubo</i>	
Zeng00: The Realization of Mixed Strategy in Simulation Soccer Game ...	465
<i>Takuya Morishita, T. Kawarabayashi, J. Nishino, H. Shimura, and H. Ogura</i>	
RoboLog Koblenz 2000	469
<i>Jan Murray, Oliver Obst, and Frieder Stolzenburg</i>	
Open Zeng: An Open Style Distributed Semi-cooperative Team Development Project from Japan	473
<i>Junji Nishino, Takuya Morishita, and Takenori Kubo</i>	
Gemini in RoboCup-2000	477
<i>Masayuki Ohta</i>	
Team Description for Lucky Lübeck — Evidence-Based World State Estimation	481
<i>Daniel Polani and Thomas Martinetz</i>	
Karlsruhe Brainstormers 2000 Team Description	485
<i>Martin Riedmiller, Artur Merke, David Meier, Andreas Hoffmann, Alex Sinner, and Ortwin Thate</i>	
ATT-CMUnited-2000: Third Place Finisher in the RoboCup-2000 Simulator League	489
<i>Patrick Riley, Peter Stone, David McAllester, and Manuela Veloso</i>	
Headless Chickens IV	493
<i>Paul Scerri, Nancy Reed, Tobias Wiren, Mikael Lönneberg, and Pelle Nilsson</i>	
Mainz Rolling Brains 2000	497
<i>Birgit Schappel and Frank Schulz</i>	
Team Description of Spatial-Timer	501
<i>Kosuke Shinoda and Susumu Kunifugi</i>	
Polytech100	505
<i>L. Stankevitch and S. Akhapkin</i>	
Team YowAI-2000 Description	509
<i>Takashi Suzuki, Sinnosuke Asahara, Hideaki Kurita, and Ikuo Takeuchi</i>	
11MonkeysII	513
<i>Naotaka Tanaka and Mio Yamamoto</i>	

Small-Size Robot (F180) League

All Botz	515
<i>Jacky Baltes</i>	
4 Stooges	519
<i>Jacky Baltes</i>	
CIIPS Glory Small Soccer Robots with Local Image Processing	523
<i>Thomas Bräunl, Petter Reinholdtsen, and Stephen Humble</i>	
ViperRoos 2000.....	527
<i>Mark M. Chang, Brett Browning, and Gordon F. Wyeth</i>	
RoboCup 2000 (F180) Team Description: UPMC-CFA Team (France)	531
<i>Jerome Douret, Thierry Dorval, Ryad Benosman, Francis Bras, Gael Surtet, Thomas Petit, Nadege Quedec, Denis Philip, Gilles Cordurié, Mario Rebello, Didier Abraham, Nicolas Couder, and Marco Marcon</i>	
MuCows	535
<i>Andrew Howard</i>	
Role-Based Strategy in TPOTs RoboCup Team.....	539
<i>Nadir Ould Khessal</i>	
LuckyStar II — Team Description Paper	543
<i>Ng Beng Kiat, Quek Yee Ming, Tay Boon Hock, Yuen Suen Yee, and Simon Koh</i>	
FU-Fighters 2000	547
<i>Raúl Rojas, Sven Behnke, Lars Knipping, and Bernhard Frötschl</i>	
RoGi Team Description	551
<i>Josep Lluís de la Rosa, Bianca Innocenti, Miquel Montaner, Albert Figueras, Israel Muñoz, and Josep Antoni Ramon</i>	
UQ RoboRoos: Kicking on to 2000	555
<i>Gordon Wyeth, Ashley Tews, and Brett Browning</i>	

Middle-Size Robot (F2000) League

ART'00 - Azzurra Robot Team for the Year 2000	559
<i>Giovanni Adorni, Andrea Bonarini, Giorgio Clemente, Daniele Nardi, Enrico Pagello, and Maurizio Piaggio</i>	
RMIT United	563
<i>James Brusey, Mark Makies, Lin Padgham, Brad Woodvine, and Karl Fantone</i>	

Agilo RoboCuppers: RoboCup Team Description	567
<i>Sebastian Buck, Robert Hanek, Michael Klupsch, and Thorsten Schmitt</i>	
WinKIT	571
<i>Kosei Demura, Nobuhiro Tachi, Noriharu Kubo, and Kenji Miwa</i>	
CMU Hammerheads Team Discription	575
<i>Rosemary Emery, Tucker Balch, Rande Shern, Kevin Sikorski, and Ashley Stroupe</i>	
GMD-Robots	579
<i>Ansgar Bredenfeld, Thomas Cristaller, Horst Guenther, Jörg Hermes, Giovanni Indiveri, Herbert Jaeger, Hans-Ulrich Kobialka, Paul-Gerhard Plöger, Peter Schoell, and Andrea Siegberg</i>	
A Goal Keeper for Middle Size RoboCup	583
<i>M. Jamzad, A. Foroughnassiraei, T. Hadji Aaghai, V.S. Mirrokni, R. Ghorbani, A. Heydar Noori, M. Kazemi, H. Chitsaz, F. Mobasser, M. Ebraahimi Moghaddam, M. Gudarzi, and N. Ghaffarzadegan</i>	
The Dutch Team	587
<i>Pieter Jonker, Will van Geest, and Frans Groen</i>	
The RoboCup-NAIST	591
<i>T. Nakamura, H. Takeda, T. Terada, M. Oohara, T. Yamamoto, and M. Takeda</i>	
KIRC: Kyutech Intelligent Robot Club	595
<i>Takeshi Ohashi, Shuichi Enokida, Junich Minatodani, Kazuhiko Kawamoto, Youji Shigeoka, Takaichi Yoshida, and Toshiaki Ejima</i>	
CoPS-Team Description	599
<i>N. Oswald, M. Becht, T. Buchheim, P. Burger, G. Hetzel, G. Kindermann, R. Lafrenz, M. Schanz, M. Schulé, and P. Levi</i>	
Golem Team in Middle-Sized Robots League	603
<i>Paolo de Pascalis, Massimo Ferrarese, Mattia Lorenzetti, Alessandro Modolo, Matteo Peluso, Roberto Polesel, Robert Rosati, Nikita Scattolin, Alberto Speranzon, and Walter Zanette</i>	
Osaka University “Trackies 2000”	607
<i>Yasutake Takahashi, Eiji Uchibe, Takahashi Tamura, Masakazu Yanase, Shoichi Ikenoue, Shujiro Inui, and Minoru Asada</i>	
Sony Legged Robot League	
Essex Rovers Team Description	611
<i>Huosheng Hu, Donbing Gu, and Bo Li</i>	

French LRP Team's Description	615
<i>Vincent Hugel, Patrick Bonnin, and Pierre Blazevic</i>	
Team ARAIBO	619
<i>Yuichi Kobayashi</i>	
CMPack '00	623
<i>Scott Lenser, James Bruce, and Manuela Veloso</i>	
The McGill's RedDogs Legged League System	627
<i>Guillaume Marceau</i>	
BabyTigers: Osaka Legged Robot Team	631
<i>Noriaki Mitsunaga, Yukie Nagai, and Minoru Asada</i>	
S.P.Q.R.	635
<i>D. Nardi, C. Castelpietra, A. Guidotti, M. Salerno, and C. Sanitati</i>	
The University of Pennsylvania RoboCup Legged Soccer Team	639
<i>James P. Ostrowski, Kenneth A. McIsaac, Aveek K. Das, Sachin Chitta, and Julie Neiling</i>	
Team Sweden	643
<i>A. Saffiotti, M. Boman, P. Buschka, P. Davidsson, S. Johansson, and Z. Wasik</i>	
RoboMutts	647
<i>Robert Sim, Paul Russo, Andrew Grahm, Andrew Blair, Nick Barnes, and Alan Blair</i>	
Humboldt Heroes	651
<i>Matthias Werner, Helmut Myritz, Uwe Düffert, Martin Lötzsch, and Hans-Dieter Burkhard</i>	
Author Index	655

Overview of RoboCup-2000

Edited by Peter Stone¹ with

Minoru Asada² (humanoid), Tucker Balch³ (workshop),
Masahiro Fujita⁴ (legged), Gerhard Kraetzschmar⁵ (mid-size),
Henrik Lund⁶ (RoboCup Jr.), Paul Scerri⁷ (simulation),
Satoshi Tadokoro⁸ (rescue), and Gordon Wyeth⁹ (small-size)

¹ AT&T Labs – Research, 180 Park Ave., Florham Park, NJ, USA

² Adaptive Machine Systems, Osaka University, Osaka, Japan

³ The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA

⁴ Sony Corp., Tokyo, Japan

⁵ Neural Information Processing, University of Ulm, Ulm, Germany

⁶ Institute for Production Technology, University of Southern Denmark, Denmark

⁷ Computer Science, Linköping University, Linköping, Sweden

⁸ Computer & Systems Engineering, Kobe University, Kobe, Japan

⁹ Computer Science and Electrical Engineering, University of Queensland,
Queensland, Australia

Abstract. The Fourth Robotic Soccer World Championships was held from August 27th to September 3rd, 2000, at the Melbourne Exhibition Center in Melbourne, Australia. In total, 83 teams, consisting of about 500 people, participated in RoboCup-2000 and about 5,000 spectators watched the events. RoboCup-2000 showed dramatic improvement over past years in each of the existing robotic soccer leagues (legged, small-size, mid-size, and simulation), while introducing RoboCup Jr. competitions and RoboCup Rescue and Humanoid demonstration events. The RoboCup Workshop, held in conjunction with the championships, provided a forum for exchange of ideas and experiences among the different leagues. This article summarizes the advances seen at RoboCup-2000, including reports from the championship teams and overviews of all the RoboCup events.

1 Introduction

RoboCup is an international research initiative that encourages research in the fields of robotics and artificial intelligence, with a particular focus on developing cooperation between autonomous agents in dynamic multiagent environments. A long-term grand challenge posed by RoboCup is the creation of a team of humanoid robots that can beat the best human soccer team by the year 2050. By concentrating on a small number of related, well-defined problems, many research groups both cooperate and compete with each other in pursuing the grand challenge.

RoboCup-2000 was held from August 27th to September 3rd, 2000, at the Melbourne Exhibition Center in Melbourne, Australia. In total, 83 teams, consisting of about 500 people, participated in RoboCup-2000. Over 5,000 spectators

watched the events. RoboCup has been advancing steadily, both in terms of size and technological level since the first international event in 1997 which included 35 teams [16, 1, 4]. Specifically, RoboCup-2000 showed dramatic improvement in each of the existing robotic soccer leagues (legged, small-size, mid-size, and simulation), while introducing RoboCup Jr. competitions and RoboCup Rescue and Humanoid demonstration events.

In addition to the simulation-based and robotic events, the RoboCup-2000 workshop provided a forum for exchange of ideas and experiences among the different leagues. 20 oral presentations and 20 posters were presented, from which four papers were nominated for the RoboCup scientific and engineering challenge awards. These distinctions are given annually for the RoboCup-related research that shows the most potential to advance their respective fields.

This article summarizes the advances seen at RoboCup-2000. Sections 2–5 describe the 4 soccer-based competition leagues. Section 6 introduces RoboCup Rescue—a disaster rescue based research effort designed to transfer RoboCup-related research to humanitarian goals. RoboCup Jr., the RoboCup education effort aimed at school children is discussed in Section 7. Scheduled to debut as a full league in 2002, the RoboCup humanoid effort held a demonstration in Melbourne, which is described in Section 8. The article concludes with an overview of the RoboCup workshop in Section 9.

2 The Sony Legged Robot League

Since RoboCup-99, all participants in the Sony legged robot league have been using the quadruped robot platform [25] which is similar to the commercial entertainment robot AIBO ERS-110 (see Figure 1). The setup and the rules of the RoboCup-2000 legged competition were based on those of RoboCup-98 [6]. Each team has 3 robots, and the size of field is 1.8m x 2.8m. Objects such as the ball and goals are painted different colors. In addition, there are 6 poles with different colors at known locations for self-localization. As is the case in human soccer, there are penalties and regulations that govern the play. We introduced two changes from the previous year’s rules in order to keep the game flowing and to encourage development of “team play” strategies. First, we introduced an obstruction rule, by which a robot that does not see the ball but is blocking other robots is removed from the play. Second, we modified the penalty area and applied the “two defender rule;” if there are two or more defenders in the penalty area, all but one is removed. As a result, the ball became stuck in the corner much less frequently. Moreover, the champion team, UNSW, implemented teammate recognition in order to avoid obstructing a teammate that was controlling the ball.

12 teams from 9 countries were selected to participate in the RoboCup 2000 Sony Legged Robot League: Laboratoire de Robotique de Paris (France), University of New South Wales (Australia), Carnegie Mellon University (USA), Osaka University (Japan), Humboldt University (Germany), University of Tokyo (Japan), University of Pennsylvania (USA), McGill University (Canada), Swe-

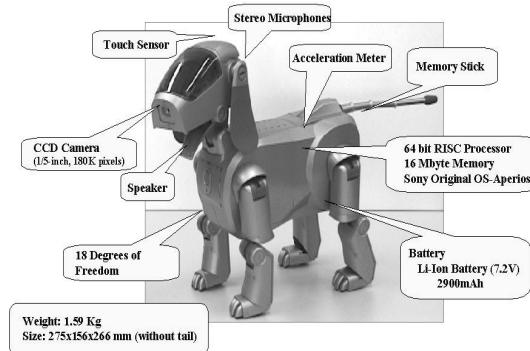


Fig. 1. The Legged Robot Platform.

den United team (Sweden), Melbourne United team (Australia), University of Rome (Italy), and University of Essex (UK). The first 9 teams above participated in the previous year's competition; the last 3 teams were new participants.

2.1 Championship Competition

For the competition, we divided the 12 teams into 4 groups of 3 teams each. After a round robin within in each group, the top 2 teams in each group proceeded to the final tournament. This year's champion is UNSW, followed by LRP in second place, and CMU in third place.

One significant improvement this year over past years was ball controlling technique. In RoboCup-99, the University of Tokyo team introduced the technique of propelling the ball with the robot's head, which can make the ball move a longer distance than can an ordinary kicking motion. This year almost all the teams implemented their own heading motion. Another impressive achievement for controlling the ball was introduced by UNSW. Their robots put the ball between their front legs, turned to change their heading while controlling the ball, and then kicked (pushed) the ball with both legs. This technique is very efficient for shooting the ball a long distance in a target direction.

2.2 RoboCup Challenge

In addition to the championship competition, every year we continue to hold the "RoboCup Challenge" as a technical routine competition. The challenge competition focuses on a particular technology more than the championship competition. This year we had 3 different technical routine challenges: (1) a striker challenge, (2) a collaboration challenge, and (3) an obstacle avoidance challenge.

The striker challenge was the simplest. The ball and one robot were placed in randomly selected positions (and orientation) on the field. The robot had to put the ball in the goal as quickly as possible. If it was unable to do so within 3 minutes, then the distance from the ball to the goal at the end of that period was measured. Note that the initial positions and orientation were selected after all the teams submitted their memory sticks with their developed software.

The collaboration challenge was defined in order to encourage the development of a passing behavior. There were two robots, one of which was put in the defensive half of the field (passer); the other was put in the offensive half (shooter). The passer and the shooter had to stay on their respective halves of the field, and the shooter had to kick the ball into the goal. The players were given 4 minutes to score a goal.

The obstacle avoidance challenge was also defined in order to encourage the development of team strategy as well as the ability to avoid a robot from the opposite team. One robot and the ball were placed on the field as in the striker challenge. In addition, two obstacles—a teammate robot with a red uniform and an opponent robot with a blue uniform—were placed at selected positions. The player had to score a goal without touching the obstacles. In both the collaboration and obstacle avoidance challenges, the time to score was recorded.

In order to complete the technical routine challenges, teams had to develop recognition algorithms for other robots, the half line, the ball, and the goals. Localization was also an important technology for the challenges.

In the striker challenge, 6 teams scored goals in an average time of 90 seconds. In the collaboration challenge, 6 goals were scored in an average of 100 sec. In the obstacle avoidance challenge, 4 teams scored in an average of 112 sec. All in all, about half of the participating teams were able to achieve the objectives of the 3 RoboCup Challenge tasks. UNSW won the challenge competition; Osaka University finished second; and CMU finished in third place.

RANK	TEAM	SCORE
1	UNSW	31
2	OSAKA	30
3	CMU	27
4	SWEDEN	27
5	TOKYO	24
6	Melbourne	16
7	HUMBOLDT	16
8	ESSEX	16
9	McGill	10
10	ROME	10
11	UPENN	10
12	LRP	10

Fig. 2. The result of the RoboCup Challenge.

3 F180: The Small-Size Robot League

Small-Size robot teams consist of up to 5 robots that can each fit into an area of 180 cm² (hence the alternative name Formula 180 or F180). The robots play on a green-carpeted table-tennis-sized field with sloping walls. The rules permit a camera to be perched above the field to be used with an off-field computer for a global vision system. This system is used to track the players, opponents and the ball. During a game the robots use wireless communication to receive tracking information from the off-field computer as well as commands or strategic information. No human intervention is allowed except for interpretation of the human referee's whistle.

The F180 games are exciting to watch as these robots can move *quickly*. The orange golf ball used as the soccer ball is propelled at speeds of over 3 m/s by ingenious kicking mechanisms. With the precise visual information from the global vision system the robots themselves can move at speeds over 1 m/s with smooth control. Nevertheless, robots moving at these speeds can and do have spectacular collisions. Intentional fouls can lead to robots being sent from the field under the shadow of a red card.

The need for speed and control has given the small-size league a reputation as the “engineering” league. Engineering disciplines including electro-mechanical design, applied control theory, power electronics, digital electronics and wireless communications have been the dominating factors in success in this league over recent years. Successful teams have typically demonstrated robot speed and powerful kicking rather than elegant ball control and sophisticated team strategies.

3.1 RoboCup 2000

Sixteen teams from nine different nations competed for the Small-Size Champion’s trophy. The early rounds of the contest demonstrated the depth of the league, with some quality teams being eliminated during the round robin section. In particular, the MuCows from Melbourne University, Australia achieved remarkable performance in their first year in the contest but were unlucky to lose in a high class group. As well as solid all-around performance, the team from Melbourne showed their engineering skill with a high-bandwidth, low power communications system that was seemingly immune to the problems experienced by most competitors.

Three Small-Size teams chose not to use the global vision system; instead these teams relied on on-board vision capture and processing to sense the environment. These teams demonstrated that it is possible to build vision hardware suitable for real time processing within the severe size constraints of the F180 league. The ViperRoos from the University of Queensland, Australia had the distinction of becoming the first local vision team to beat a global vision team—the score was 2:0. However, none of the local vision teams were able to reach the finals.

The eight finalists all had excellent technical merit. Team Crimson from Korea has a custom video processing board that extracts the position of the

players and the ball at the full NTSC video rate of 60 Hz. It does so without ever buffering the video in RAM, so that the position information is delayed by only 1/60th of a second. With such a small delay in vision processing combined with highly responsive robots, Team Crimson was capable of extremely fast and controlled motion. However, due to problems with communications (and some last minute code changes!) the team was knocked out in the quarter finals.

The French team from the Université Pierre et Marie Curie were the only team to score against the eventual champions, Big Red from Cornell University. The French curved path planning system allowed them to scoop the ball from in front of the opposition and make highly effective attacks on goal. They were unlucky to be knocked out by Cornell in their quarter final.

The first semi-final between FU-Fighters from the Freie Universität of Berlin, Germany and the RoboRoos from the University of Queensland, Australia showed a contrast of styles. The RoboRoos, competing for the third consecutive year, had relied on smooth control and an adaptive team strategy to reach the finals, whereas the FU-Fighters used fast, aggressive trajectories with an extremely powerful kicker. The FU-Fighters showed clear dominance winning the match 3:0.

The second semi-final between Cornell and Lucky Star from Ngee Ann Polytechnic in Singapore was the closest match of the Small-size tournament. The match was 0:0 at full time, playing through a period of sudden death extra time to come down to a penalty shoot out that was decided at 4:3. Lucky Star combined novel electro-mechanical design with excellent control to achieve their result. Their robots had an extremely effective kicking mechanism that was integrated in a narrow body design. The narrow body enabled the robots to slip between defenders to get to the ball, despite the crowding of the field. Their vision and control was sufficiently good that they would reliably kick the ball despite the small kicking face of the robot. Lucky Star won third place in the contest.

The team from Cornell went on to win the final against the FU-Fighters convincingly. Figure 3 is a shot from the final game. This is the second consecutive year that Cornell has won the small-size championship and the second year that FU-Fighters have come second. While it might seem natural to attribute their achievements to novel electromechanical design such as FU-Fighter's powerful kicker or Cornell's dribbling device, it is also apparent that these robots are superbly controlled. As these control issues, along with the other fundamental engineering issues, are addressed on an even scale across the competition, other factors such as effective team strategies will come more into play.

4 F2000: The Middle-Size Robot League

The RoboCup F2000 League, commonly known also as middle-size robot league, poses a unique combination of research problems, which has drawn the attention of well over 30 research groups world-wide.



Fig. 3. The small-size league final game.

4.1 Environment and Robots

The playing environment is designed such that the perceptual and locomotion problems to be solved are reasonably simple, but still challenging enough to ignite interesting research. The field size is currently $9m \times 5m$. The goals have colored walls in the back and on the sides (yellow/blue). The field is surrounded by white walls (50cm height) that carry a few extra markings (squared black markers of 10cm size plus black-and-white logos of sponsors in large letters). A special corner design is used and marked with two green lines. The goal lines, goal area, center line and center circle are all marked with white lines. The ball is dark orange. Illumination of the field is constrained to be within 500 and 1500 lux. Matches are played with teams of four robots, including the goalie.

The robots must have a black body and carry color tags for team identification (light blue/magenta). Quite elaborate constraints exist for robot size, weight, and shape. Roughly, a robot body may be up to about 50cm in diameter and be up to 80cm in height; must weigh less than 80kg; and must have no concavities large enough to take up more than one-third of the ball's diameter. The robots must carry all sensors and actuators on-board; no global sensing system is allowed. Wireless communication is permitted both between robots and between robots and outside computers.

4.2 Research Challenges

The most notable difference from the F180 league is that global vision is not permitted. In a global camera view, all the robots and the ball move, while the goals, the walls and the markings of the field remain fixed. If the moving objects can be tracked sufficiently fast in the video stream, all the positions and orientations are known and a global world model is available. The situation is completely different in F2000, where the cameras on top of the robots are moving through the environment. All the usual directional cameras, and most omnidirectional cameras, can perceive only a small part of the environment. This greatly complicates tasks like finding the ball, self-localizing on the field, locating teammates

and opponents, and creating and updating a world model. In addition, the vast majority of F2000 robots are completely autonomous, carrying all sensors and computational equipment onboard, which makes them much larger and heavier. Fast movements are much more difficult to control. These are two of the main reasons why F2000 robots play at much slower speeds than F180 robots.

The difficulties described above exert a strong force to new teams to think about robot design, and repeatedly new teams with new hardware designs have displayed stunning first-time appearances at RoboCup tournaments. This year we had another two examples: CMU Hammerheads from USA and GOLEM from Italy, each of which introduced a new mobile base into the middle-size league. The Hammerheads use a modified version of the commercially available Cye robot, a differential drive base with a trailer attached to it. The GOLEM robots feature a triangular omnidirectional drive design based on mechanum wheels, which provided for the best combination of maneuverability and speed the F2000 league has seen so far. The drive design of the GOLEM robots was complemented by the use of only a single sensor: an omnidirectional camera with a custom-made mirror design, which provided the robot with a complete view of the field from virtually every position. The clever combination of these two key design decisions allowed the GOLEM team to apply much simpler techniques for localization and world modeling as well as action selection, which significantly reduced development time.

4.3 RoboCup-2000 Tournament

Fifteen teams participated in the RoboCup-2000 middle-size league tournament. The rules for the middle-size robot league were only marginally changed from last year, which gave teams the opportunity to focus on software improvements rather than the design of new hardware. The play schedule was designed to give all teams ample opportunity to gain practical playing experience, with a total of 57 games. Each team was assigned to one of two groups, with 7 and 8 teams, respectively, for the qualification rounds. Each group played a single round robin schedule, such that each team played at least six or seven games. The four top teams in each group went to the playoff quarterfinals.

In this year's tournament, we had more exciting matches than ever, with quite a number of surprising performances. Most teams had previous tournament experience and showed significant progress over previous play levels. In addition, we had two remarkable newcomers this year, CMU Hammerheads from the United States and GOLEM from Italy, both of which made it to the quarterfinals, a remarkable success, especially for new teams. The other teams reaching the quarterfinals were last year's champion Sharif CE from Iran, RMIT United from Melbourne, Australia, the Osaka University Trackies from Japan, and the three German teams GMD Robots, Bonn, Agilo Robocuppers from Munich, and CS Freiburg. GOLEM, Sharif CE, Trackies, and CS Freiburg qualified for the semifinals. The semifinals and finals matches were the most exciting games in middle size league history, watched by a crowd of more than a thousand enthusiastic spectators. Both the third-place game and the final game took penalty

shootouts to determine the winners. Last year's champion Sharif finished 3rd after tying the Trackies 1:1 at full-time and winning the penalty kicks 3:2. The final game between Freiburg and GOLEM was tied 3:3 at full-time. During the penalty shootout, Freiburg first scored three of five penalty kicks. Then, it was GOLEM's turn and they scored the first penalty kick. Excitement was at its peak when they missed the next two. Freiburg defended the next one as well and became the RoboCup-2000 middle size league champion.

4.4 Lessons Learned and Future Developments

When the newcomer team Sharif CE from Iran won last year, many observers attributed their superior performance largely to their new hardware design, which gave them more speed and more maneuverability than most other teams. With the GOLEM team from Italy, we had yet another team with a new mobile platform making it to the finals. Many AI people were concerned that the focus in F2000 would shift mainly to new mechanical designs and hardware work. However, this year CS Freiburg won the championship because of their superior software capabilities; except for slightly redesigned kickers, the hardware design has remained almost the same since the team started out in 1998. Many teams have much faster, more maneuverable robots than Freiburg.

The outcome of the last two tournaments illustrate an old truth about mobile robots: *Better hardware may compensate for some software deficiencies, and better software may compensate for some hardware deficiencies.* Overall, there is still much work to do in *all* areas of cooperative, autonomous mobile robots. There is no precedence of any single field over another one; the only clearly identifiable precedence is that *robust and reliable* systems almost always win over less robust, less reliable teams. But robustness and reliability is something that one must achieve in all parts of a system.

After a year of keeping the rules virtually unchanged, it is now time to think about modifications that promote research particularly in two directions:

- *Making robots more robust and reliable.* Comparatively small changes in the environment often disturb the robots' performances significantly. Reducing the dependency on environmental color coding and to develop fast and robust algorithms for perceptual tasks like object detection, object localization, and object tracking is an essential goal for future research.
- *Enhancing playing skills.* Most robots push or kick the ball with a simple device; only few robots could demonstrate dribbling capabilities, such as taking the ball around an opponent in a controlled manner. Playing skills can be improved by more thorough application of learning techniques. In addition, we need to relax some of our constraints on robot's form and shape in order to promote the design of innovative ball manipulation devices.

Rule changes to foster research in these directions can be expected for future tournaments.

5 The Simulation League

The RoboCup 2000 competition was the most exciting and most interesting simulation competition so far. As in past years, the competition was run using the publicly available soccer server system [15]. 34 teams from 14 countries met in a round robin competition followed by a double elimination final series. While most of the teams had competed in previous competitions there were several notable new entries, including the eventual champions, FC Portugal who had an exciting 1:0 final with Karlsruhe Brainstormers. The high standard of the competition made for many exciting matches throughout the competition – nearly 25% of final-round games went into overtime, one eventually having to be decided by a coin toss after scoreless overtime lasted the length of two normal matches.

5.1 The RoboCup soccer server

The RoboCup soccer server provides a standard platform for research into multi-agent systems. The soccer server simulates the players and field for a 2D soccer match. 22 clients (11 for each team) connect to the server, each client controlling a single player. Every 100ms the Soccer Server accepts commands, via socket communication, from each client. The client sends low level commands (dash, turn or kick) to be executed (imperfectly) by the simulated player it is controlling. Clients can only communicate with each other using an unreliable, low bandwidth communication channel built into the soccer server. The soccer server simulates the (imperfect) sensing of the players, sending an abstracted (objects, e.g. players and ball, with direction, distance and relative velocity) interpretation to the clients every 150ms. The field of view of the clients is limited to only a part of the whole field. The Soccer Server enforces most of the basic rules of (human) soccer including off-sides, corner kicks and goal kicks and simulates some basic limitations on players such as maximum running speed, kicking power and stamina limitations.

An extra client on each team can connect as a “coach”, who can see the whole field and send strategic information to clients when the play is stopped, for example for a free-kick.

The SoccerMonitor connects to the soccer server as another client and provides a 2D visualization of the game for a human audience (see Figure 4). Other clients can connect in the same way to do things like 3D visualization, automated commentary and statistical analysis.

Observers of the competition noted that, occasionally, watching RoboCup soccer games was like watching real soccer. This observation was likely based on the realism of the team level strategies, especially in the way the players moved the ball between themselves and, when not having the ball strategically positioned themselves to the teams advantage. The realism may also have been due to the flexibility the teams showed against unknown opponents and in novel situations.



Fig. 4. A screen-shot of the Soccer Server monitor augmented with FC Portugal's debugging tools.

5.2 Research Themes

RoboCup simulation requires the building of complete intelligent agents, i.e. agents must be capable of everything from handling uncertain sensors, through low level, real-time control to team coordination, in order to compete successfully. This is exciting because it shows RoboCup is exposing the strengths and weaknesses of our research and forcing us to address the weaknesses – in turn making that research stronger.

Many of the research challenges addressed by teams in 2000 came out of problems observed by teams from previous competitions. Two research themes were especially prominent, the first theme being learning and the second being multiagent coordination. Other research areas included improving situational awareness given incomplete and uncertain sensing and high level team specification by human designers.

The first research theme, especially common amongst the successful teams, was learning. Teams adapted techniques like simulated annealing, genetic programming or neural nets to the problem of creating very optimized low level skills such as dribbling (e.g. [20]). Experience has shown that while advanced skills were an essential component of a successful team, building such skills by hand is difficult and time consuming. The skills developed with learning techniques were in some cases superior to the hand developed skills of previous years. Hence, RoboCup has provided a useful, objective example of a case where learning can produce a better outcome than labor intensive programming.

Not all learning research was focused on low-level skills – several teams addressed the problem of how to learn high level strategies. RoboCup provides an interesting domain to investigate such issues because although there is a clearly defined objective function, i.e. win the game, the huge state space, unpredictable opponent, uncertainty, etc. make the problem very challenging. Most approaches learning at a high level layered the learning in some way (a successful approach

in the 1999 competition), although the specifics of the learning algorithms varied greatly from neural networks to evolutionary algorithms.

The second major research theme was multiagent coordination. While in previous competitions, a highly skilled team might do reasonably well with “kiddie soccer” tactics, e.g. dribbling directly to goal, so many teams this year had high quality skills that more sophisticated team strategies were required to win games. Conversely, the high quality skills triggered more interest in team strategies because players had the ability to carry them out with some consistency. As well as the learning approach to developing high level strategies, a variety of human engineered approaches were used (e.g. [14]). A key to many of the approaches was the online coach. The coach was commonly used to analyze the opposition and determine appropriate changes to the team strategy [5]. Other teams developed tools or techniques aimed at empowering human designers to easily specify strategies. Yet, other teams relied on carefully engineered emergent team behavior (e.g. [19]) or dynamic team planning to achieve the desired team behavior.

5.3 RoboCup-2000

RoboCup simulation teams are increasingly complex pieces of software usually consisting of tens of thousands of lines of code with specialized components working together in real-time. Handling the complexity is forcing researchers to look critically at agent paradigms not only in terms of the resultant agent behavior but also at the ease with which very complex teams can be developed within that paradigm (and how that should be done).

However, the rapidly increasing complexity of RoboCup simulation agents should not deter new researchers from starting to work with RoboCup. An online team repository currently contains source code or binaries for 29 of the teams that competed in the 1999 World Cup plus many more from previous years. The repository allows new RoboCup participants to quickly get a team going. In fact a number of the top teams in 2000 were developed on top of the freely available code of the 1999 champions, CMUnited-99. The growing code base provides code for interaction with the soccer server, skills, strategies, debugging tools, etc. in a variety of programming languages and paradigms. A big effort has been made over the last few years to encourage teams to release source code or at least binaries of their teams.

The reigning champion team, CMUnited-99, was re-entered unchanged in the 2000 competition to assess the advances made during the year. In 2000 CMUnited-99 finished 4th. In 1999, CMUnited-99’s aggregate goals for and against tally was 110-0, while in 2000 the tally was the far more competitive 25-7 (including a 13-0 win). Also interesting was that 4 of 6 of CMUnited-99’s elimination-round games went into overtime (resulting in 3 wins and 1 loss). CMUnited-99’s record in 2000 shows two things: (i) although they finished fourth several teams were nearly as good and, perhaps, unlucky to lose to them and (ii) the competition was extremely tight. It also indicates just how good CMUnited-99 were in 1999.

As well as the main competition, there were extensive evaluation sessions designed to compare the ability of teams to handle increased sensor and effector uncertainty. The sensor test was a repeat of the test from last year and involved changing the average magnitude of the error in the simulated visual information players received. The effector test was a surprise to the teams and involved changing the average magnitude of the difference between a command sent by a player and what was actually executed. The evaluation session provides a unique opportunity to test a wide variety of agent implementations under identical conditions. Extensive evaluation log-files, providing a large amount of high quality date, are available for analysis.

Despite the advances made in 2000, the RoboCup simulator is far from a solved problem. While high level learning has progressed significantly, learned high level strategies were generally inferior to hand-coded ones – a challenge for 2001 is to have learned strategies outperform hand-coded ones. Using RoboCup simulation as a platform for research into high level multiagent issues is only just starting to emerge, via, for example, use of the online coach. Additionally, as the standard of play gets higher there is both increased interest and use for opponent modeling techniques that can counter complex, previously unseen team strategies. The rapidly increasing complexity of RoboCup software challenges us to continue improving our methods for handling complexity. The advances made and the research areas opened up in 2000 bode well for yet another interesting, exciting competition in 2001.

6 RoboCup Rescue

The RoboCup-Rescue Project was newly launched by the RoboCup Federation in 1999. Its objective is as follows.

1. Development and application of advanced technologies of intelligent robotics and artificial intelligence for emergency response and disaster mitigation for the safer social system.
2. New practical problems with social importance are introduced as a challenge of robotics and AI indicating a valuable direction of research.
3. Proposal of future infrastructure systems based on advanced robotics and AI.
4. Acceleration of rescue research and development by the RoboCup competition mechanism.

A simulation project is running at present, and a robotics and infrastructure project will soon start.

In Melbourne, a simulator prototype targeting earthquake disaster was open to the public to start international cooperative research. A real rescue robot competition was proposed to start a new league in 2001.

6.1 Simulation Project

Distributed simulation technology combines the following heterogeneous systems to make a virtual disaster field. (i) *Disaster simulators* model the collapse

of buildings, blockage of streets, spread of fire, traffic flow, and their mutual effects. (ii) *Autonomous agents* represent fire brigades, policemen, and rescue parties, all of which act autonomously in the virtual disaster. (iii) The *Simulation Kernel* manages state values and networking of/between the systems. (iv) The *Geographical Information System* gives spatial information to the whole system. (v) *Simulation Viewers* show 2D/3D image of simulation results in real time as shown in Fig. 5.

The RoboCup-Rescue simulation competition will start in 2001. The details are described in papers and a book [22, 9, 23, 7, 17, 21]. The simulator prototype can be downloaded from <http://robomec.cs.kobe-u.ac.jp/robocup-rescue/>.

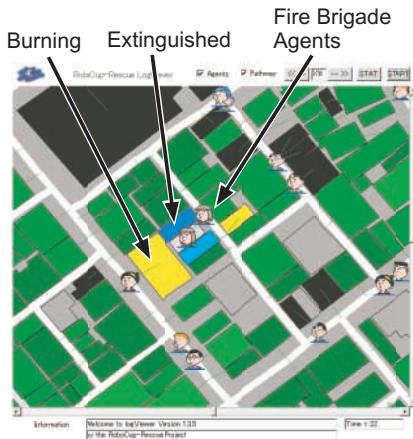


Fig. 5. 2D viewer image of RoboCup-Rescue prototype simulator.

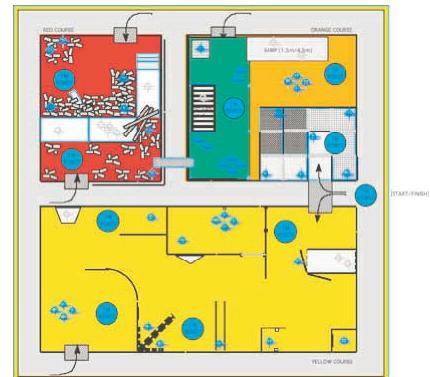


Fig. 6. AAAI USAR contest field.

6.2 AAAI/RoboCup Rescue Robot Competition

A rescue robot competition will start in 2001 in cooperation with AAAI. The target is search and rescue of confined people from collapsed buildings such as in earthquake disasters and explosion disasters. In Melbourne, Robin Murphy (USF) demonstrated 2 robots that are developed for real operations.

The large-scale arena of the AAAI Urban Search and Rescue (USAR) Contest (Fig. 6) will be used. It consists of three buildings simulating various situations. The easiest building has a flat floor with minimal debris, but the most difficult building includes a 3D maze structure consisting of stairs, debris, etc. with narrow spaces. The details are described on the AAAI USAR web page (<http://www.aic.nrl.navy.mil/~schultz/aaai2000/>).

More than other RoboCup competitions, the rules of the 2001 rescue competition will focus on direct technology transfer, specifically to real disaster problems on the basis of the 2000 AAAI USAR Contest. For example, practical

semi-autonomy with human assistance and information collection for realistic operation are potential competition components.

7 RoboCup Junior

RoboCup Jr. is the educational branch of RoboCup, and it puts emphasis on teaching young people about research and technology by giving them hands-on experience. RoboCup Jr. development was initiated in 1997, and the first public show was at RoboCup'98 in Paris with a demonstration of LEGO Mindstorms robots playing soccer in a big LEGO stadium with rolling commercials, LEGO spectators making the wave, stadium lights, etc. [11] and with children playing with other LEGO robot models. In 1999, during RoboCup'99 in Stockholm, children were allowed to program their own LEGO Mindstorms robots in the morning, and then play tournaments in the afternoon [12]. The fast development of complex robot behaviors was achieved with the Interactive LEGO Football set-up based on a user-guided approach to behavior-based robotics. This activity was refined for RoboCup-Euro-2000 in Amsterdam [8], where 10 Dutch and 2 German school groups participated in a one-day tournament.

The RoboCup Jr. 2000 activity in Melbourne, in which a total of 40 groups of children participated, differed from the previous activities in several aspects: (1) children were both building and programming their robots, (2) the development took place during 6-8 weeks prior to the competition, (3) in most cases, the work was done as part of a teaching project in schools, (4) there was a robot sumo competition and a robot dance performance, in addition to the soccer competition.

During previous events, children had no opportunity to build the robots. But educational approaches such as constructionism [18,10] suggest that the construction of an artifact is important in order to understand the artifact, so RoboCup Jr. 2000 allowed children to both build and program the robots. This endeavor was facilitated by the use of LEGO Mindstorms robots, partly because this tool allows for easy assembly of robots, and partly because most children are familiar with LEGO. The tasks were designed so that the simple sensors and actuators are sufficient, but a few children from the more advanced technical classes made their own sensors, and integrated them with the LEGO Mindstorms control unit.

There were three different events during RoboCup Jr. 2000, namely the Dance-Performance Event for students up to 12 years of age (Primary), the Converging Robot Race (Sumo) for students up to 14 years of age (Years 7 and 8), and RoboCup Jr 2000 Soccer for students of 14 to 18 years of age (Years 7 - 12). We put special emphasis on broadening RoboCup Jr. from being a purely competitive event to include the cooperative event of a robot dance/parade. In previous years, and during RoboCup Jr. 2000, we found the competitive robot soccer event to result in a gender bias towards boys. This bias is not surprising, since the robot soccer event promotes soccer, technology, vehicles, and competition, and we often find that boys are more enthusiastic about these subjects

than girls. We did not perform any rigorous scientific gender studies, but our experience from many events gave a clear picture of a gender bias. We therefore introduced the dance/parade, in order to address other issues, such as cooperation, context construction, and performance. Indeed, more than 50% of the participants who signed up for the robot dance/performance event were girls.

Each participating team had 3 minutes for the robot dance/performance. The teams designed the robots; designed the environment in which the robots danced; programmed the robots to perform; and made a music cassette with the appropriate music for the performance. Many of the teams also designed their own clothes to match the robots and the environment, and many teams designed clothes for the robots. There was no limitation to the hardware (any robot can be used), but during RoboCup Jr. 2000, all participating teams chose to use LEGO Mindstorms. The performing robots included a Madonna look-alike, a disco-vampire, a dragon on the beach, and four feather-dressed dancers. Ten teams participated in the Dance/Performance Event, and prizes were given for best dressed robot, best programming, best choreography, most entertaining (best smile value), best team T-shirt design, best oral presentation by participants to judges, and creativity of entry.

The RoboCup Jr. soccer game had 20 participating teams. Each team built one or two robots (in all cases from LEGO Mindstorms) to play on a field of approximately 150cm × 90cm. The floor of the field is a gradient from black to white, which allows the robots to detect position along one of the axes by measuring reflection from the floor with a simple light sensor. The ball used in the finals was an electronic ball produced by EK Japan (see [12]). The ball emits infrared that can be detected with very simple, off the shelf LEGO sensors. Bellarine Secondary College won the final by drawing 3-3 and winning on golden-goal, after being down 3-0 at half time.

The success of the RoboCup Jr. 2000 event was to a large degree due to the involvement of very enthusiastic local teachers and toy/hardware providers, who promoted and designed the event in collaboration with the researchers. The local teachers were able to incorporate the RoboCup Jr. project in their curricula. Involvement of local teachers seems crucial for the success of such events. In the future, RoboCup Jr. will make an effort to promote national and local competitions, apart from the big events at the yearly RoboCup. Figure 7 shows images from this year's event.

8 Humanoid Robot Demonstration

The RoboCup humanoid league will start in 2002 towards the final goal of RoboCup, which is to beat the human World Cup soccer champion team with a team of eleven humanoid robots by 2050. This league will be much more challenging than the existing ones because the dynamic stability of robots walking and running will need to be handled.

The main steps of such development will be: (i) building an autonomous biped able to walk alone on the field; (ii) locomotion of this biped, including



Fig. 7. Images of the RoboCup Jr. events

straight-line movement, curved movement, and in-place turns; (iii) identification of the ball, the teammates, and the opponents; (iv) kicking, passing, shooting, intercepting, and throwing the ball; (v) acquisition of cooperative behavior (co-ordination of basic behaviors such as passing and shooting); and (vi) acquisition of team strategy.

Although items (iii)–(vi) are already addressed in the existing leagues, the humanoid league has its own challenges related to handling the ball with feet and hands.

At RoboCup-2000, the humanoid demonstration was held with four characteristic humanoids. Figure 8 shows these four humanoids, pictured from left to right. Mark-V, on the left is from Prof. Tomiyama's group at Aoyama Gauin University. Mark-V showed its ability to walk and kick a ball into a goal. Second from the left is PINO from the Kitano Symbio Project, Japan. PINO demonstrated walking and waving his hand to say "Good Bye!" Second from the right is Adam from LRP, France. Adam walked 100 cm in a straight line autonomously and was also controlled by an off-board computer. On the right is Jack Daniel from Western Australia University. Jack demonstrated a walking motion while suspended in the air.

These humanoids are still under development. At RoboCup-2001 we expect to see more humanoids with improved walking and running and also some new capabilities.

9 RoboCup Workshop and Challenge Awards

There is no doubt that RoboCup is an exciting event: the matches are thrilling to watch and the robots and programs are fun to design and build. Even so, RoboCup is fundamentally a *scientific* event. It provides a motivating and an easy to understand domain for serious multiagent research. Accordingly, the RoboCup Workshop, which is held each year in conjunction with the Robot Soccer World Cup, solicits the best work from participating researchers for presentation.

The RoboCup-2000 Workshop was held in Melbourne, adjacent to the exhibition hall where the competitions were staged. This year 20 papers were selected



Fig. 8. Four humanoids demonstrated at RoboCup-2000

for full presentation and an additional 20 were selected for poster presentation from over 60 submissions. Paper topics ranged from automated intelligent sportscaster agents to motion planners and vision systems. The Workshop was attended by over 200 international participants.

The number of high-quality submissions to the RoboCup Workshop continues to grow steadily. To highlight the importance of the scientific aspects of RoboCup, and to recognize the very best papers, the workshop organizers nominated four papers as challenge award finalists. The challenge awards are distinctions that are given annually to the RoboCup-related research that shows the most potential to advance their respective fields. The finalists were:

- *A localization method for a soccer robot using a vision-based omni directional sensor* by Carlos Marques and Pedro Lima.
- *Behavior classification with self-organizing maps* by Michael Wünstel, Daniel Polani, Thomas Uthmann and Jürgen Perl.
- *Communication and coordination among heterogeneous mid-size players: ART99* by Claudio Castelpietra, Luca Iocchi, Daniele Nardi, Maurizio Piaggio, Alessandro Scalso and Antonio Sgorbissa.
- *Adaptive path planner for highly dynamic environments* by Jacky Baltes and Nicholas Hildreth.

These presentations were evaluated by a panel of judges who attended the presentations based on the papers themselves, as well as the oral and poster presentations at the workshop. This year two awards were given: The scientific challenge award was given to Wünstel, *et al.* for their work on applying self-organizing maps to the task of classifying spatial agent behavior patterns; the engineering challenge award was given to Marques and Lima for their contribution to sensing and localization. All four finalist papers appear in this volume.

We expect that Workshop will continue to grow. In future years we may move to parallel tracks so that more presentations will be possible.

10 Conclusion

RoboCup-2000 showed many advances, both in the existing competition leagues and in the introduction of several new events. The participation and attendance were greater than ever, with about 500 participants and more than 5,000 spectators.

RoboCup-2001 is going to be held in the United States for the first time. It will run from August 2nd through August 10th, 2001 in Seattle co-located with the International Joint Conference of Artificial Intelligence (IJCAI-2001). RoboCup-2001 will include a 2-day research forum with presentations of technical papers, and all competition leagues: soccer simulation; RoboCup rescue simulation (for the first time); small-size robot (F180); middle-size robot (F2000); four-legged robot; and RoboCup rescue robot in conjunction with the AAAI robot competition (for the first time). It will also include a RoboCup Jr. symposium including 1 on 1 robot soccer and robot dancing competitions, and other educational events for middle-school and high-school children. Finally, RoboCup-2001 will include an exhibition of humanoid robots.

For more information, please visit: <http://www.robocup.org>.

Appendix A: Sony Legged Robot League Results

This appendix includes all the results from the games in the Sony legged robot league. The tables show the preliminary round games, including numbers of wins, losses, and draws (W/L/D), and the rank of each team within each group. Figure 9 shows the results of the single-elimination championship tournament.

Group A

- A1: Osaka
- A2: Sweden
- A3: Tokyo

	A1	A2	A3	W/L/D	Rank
A1		0–3	2–1	1/1/0	3
A2	3–0		2–4	1/1/0	2
A3	4–2	1–2		1/1/0	1

Group B

- B1: LRP
- B2: UPenn
- B3: Rome

	B1	B2	B3	W/L/D	Rank
B1		2-1	4-0	2/0/0	1
B2	1-2		0-2	0/2/0	3
B3	0-4	2-0		1/1/0	2

Group C

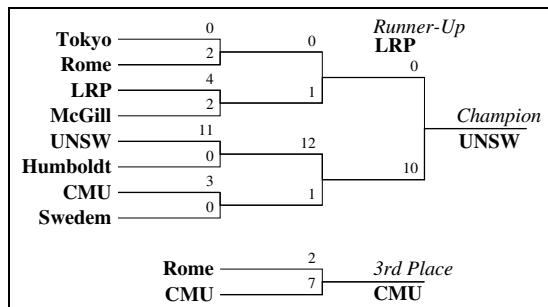
- C1: UNSW
- C2: McGill
- C3: Essex

	C1	C2	C3	W/L/D	Rank
C1		14-0	0-0	2/0/0	1
C2	0-14		0-0	1/1/0	2
C3	0-0	0-0		0/2/0	3

Group D

- D1: CMU
- D2: Humboldt
- D3: Melbourne

	D1	D2	D3	W/L/D	Rank
D1		4-0	8-0	2/0/0	1
D2	0-4		1-0	1/1/0	2
D3	0-8	0-1		0/0/1	3

**Fig. 9.** The Sony legged robot league championship tournament.

Appendix B: Small-Size Robot League Results

This appendix includes all the results from the games in the small-size robot league. The tables show the preliminary round games, including numbers of wins, losses, and draws (W/L/D), and the rank of each team within each group. Figure 10 shows the results of the single-elimination championship tournament.

Group A

- A1: Cornell Big Red
- A2: CFA UPMC
- A3: MU-Cows
- A4: 4 Stooges

	A1	A2	A3	A4	W/L/D	Rank
A1		1–1	7–0	40–0	2/0/1	1
A2	1–1		1–0	11–0	2/0/1	2
A3	0–7	0–1		14–0	1/2/0	3
A4	0–40	0–11	0–14		0/3/0	4

Group B

- B1: Fu Fighters
- B2: Rogi Team
- B3: TPOTS
- B4: CIIPS Glory

	B1	B2	B3	B4	W/L/D	Rank
B1		4–0	13–1	7–0	3/0/0	1
B2	0–4		2–0	10–0	2/1/0	2
B3	1–13	0–2		3–0	1/2/0	3
B4	0–7	0–10	0–3		0/0/3	4

Group C

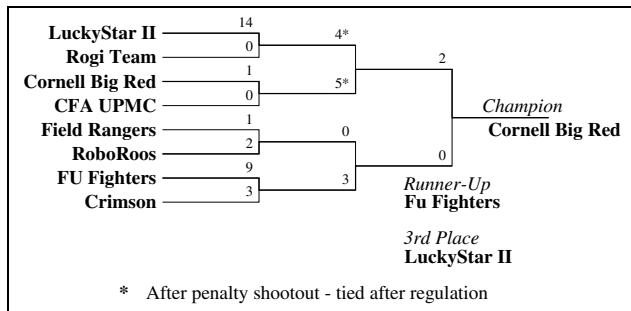
- C1: LuckyStar II
- C2: Crimson
- C3: TUD
- C4: ViperRoos

	C1	C2	C3	C4	W/L/D	Rank
C1		24–0	66–0	7–0	3/0/0	1
C2	0–24		14–0	6–0	2/1/0	2
C3	0–66	0–14		0–2	0/3/0	4
C4	0–7	0–6	2–0		1/2/0	3

Group D

- D1: RoboRoos
- D2: Field Rangers
- D3: All Botz
- D4: Yale Frobocup

	D1	D2	D3	D4	W/L/D	Rank
D1		0–1	14–1	22–0	2/1/0	2
D2	1–0		6–0	8–1	3/0/0	1
D3	1–14	0–6		1–0	1/2/0	3
D4	0–22	8–1	0–1		0/3/0	4

**Fig. 10.** The small-size robot league championship tournament.

Appendix C: Middle-Size Robot League Results

This appendix includes all the results from the games in the middle-size robot league. The tables show the preliminary round games, including numbers of wins, losses, and draws (W/L/D), and the rank of each team within each group. Figure 11 shows the results of the single-elimination championship tournament.

Group A

- A1: CS Freiburg
- A2: Vanquish
- A3: CoPS Stuttgart
- A4: RMITUnited
- A5: KIRC
- A6: NAIST 2000
- A7: Sharif CE
- A8: Golem Team

	A1	A2	A3	A4	A5	A6	A7	A8	W/L/D	Rank
A1		8-0	3-1	3-0	4-0	9-1	5-1	1-2	6/1/0	1
A2	0-8		0-1	0-8	2-2	0-1	0-4	0-8	0/6/1	8
A3	1-3	1-0		0-2	2-0	2-0	0-2	1-6	3/4/0	5
A4	0-3	8-0	2-0		8-1	7-0	1-3	3-5	4/3/0	4
A5	0-4	2-2	0-2	1-8		0-2	0-5	1-5	0/6/1	7
A6	1-9	1-0	0-2	0-7	2-0		0-9	0-6	2/5/0	6
A7	1-5	4-0	2-0	3-1	5-0	9-0		4-1	6/1/0	3
A8	2-1	8-0	6-1	5-3	5-1	6-0	1-4		6/1/0	2

Group B

- B1: Agilo Robocoppers
- B2: GMD-Robots
- B3: CMU Hammerheads
- B4: Vanquish
- B5: Win KIT
- B6: ART 2000
- B7: Trackies

	B1	B2	B3	B4	B5	B6	B7	W/L/D	Rank
B1		0-2	2-0	3-1	3-0	1-0	0-4	4/2/0	3
B2	2-0		1-1	2-0	3-1	1-0	0-7	4/1/1	2
B3	0-2	1-1		0-2	3-0	4-1	1-4	2/3/1	4
B4	1-3	0-2	2-0		4-1	0-0	0-5	2/3/1	5
B5	0-3	1-3	0-3	1-4		0-0	0-7	0/5/1	7
B6	0-1	0-1	1-4	0-0	0-0		1-8	0/4/2	6
B7	4-0	7-0	4-1	5-0	7-0	8-1		6/0/0	1

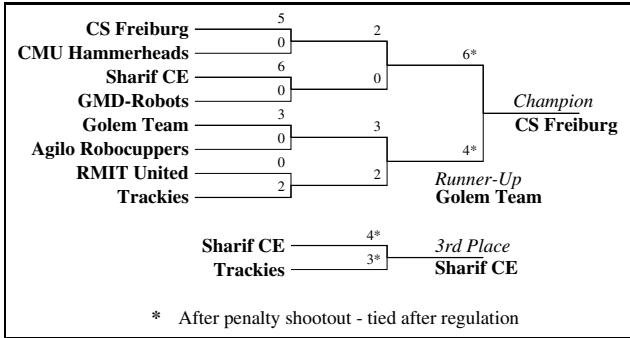


Fig. 11. The middle-size robot league championship tournament.

Appendix D: Simulation League Results

This appendix includes all the results from the games in the simulator league. The tables show the preliminary round games, including numbers of wins, losses, and draws (W/L/D), and the rank of each team within each group. Figure 12 shows the results of the double-elimination championship tournament.

Group A

- A1: ATT-CMUUnited-2000
- A2: Open Zeng
- A3: A-Team
- A4: Virtual Werder

	A1	A2	A3	A4	W/L/D	Rank
A1		1-0	13-0	6-0	3/0/0	1
A2	0-1		9-0	4-2	2/1/0	2
A3	0-13	0-9		0-16	0/3/0	4
A4	0-6	2-4	16-0		1/2/0	3

Group B

- B1: CMUUnited-99
- B2: Gemini
- B3: Sharif II
- B4: SBC

	B1	B2	B3	B4	W/L/D	Rank
B1		2-0	8-0	4-0	3/0/0	1
B2	0-2		4-0	1-0	2/1/0	2
B3	0-8	0-4		1-0	1/2/0	3
B4	0-4	0-1	0-1		0/3/0	4

Group C

- C1: Magma Freiburg
- C2: TakAI
- C3: Gnez
- C4: Paso Team

	C1	C2	C3	C4	W/L/D	Rank
C1		3–0	14–0	25–0	3/0/0	1
C2	0–3		4–0	21–0	2/1/0	2
C3	0–14	0–4		5–0	1/2/0	3
C4	0–25	0–21	0–5		0/3/0	4

Group D

- D1: Essex
- D2: HC-IV
- D3: Sharif Arvand
- D4: Spatial Timer
- D5: KU-Yam

	D1	D2	D3	D4	D5	W/L/D	Rank
D1		10–0	0–0	7–0	16–0	3/0/1	2
D2	0–10		0–18	0–11	1–1	0/3/1	5
D3	0–0	18–0		3–0	18–0	3/0/1	1
D4	0–7	11–0	0–3		3–0	2/2/0	3
D5	0–16	1–1	0–18	0–3		0/3/1	4

Group E

- E1: FC Portugal
- E2: Robolog
- E3: Zeng00
- E4: Oulu2000

	E1	E2	E3	E4	W/L/D	Rank
E1		20–0	18–0	33–0	3/0/0	1
E2	0–20		0–0	39–0	1/1/1	2
E3	0–18	0–0		21–0	1/1/1	3
E4	0–33	0–39	0–21		0/3/0	4

Group F

- F1: Karlsruhe
- F2: Mainz
- F3: Kakitsubata
- F4: Wright Eagle

	F1	F2	F3	F4	W/L/D	Rank
F1		10–0	18–0	1–0	3/0/0	1
F2	0–10		7–0	0–7	1/2/0	3
F3	0–18	0–7		0–8	0/3/0	4
F4	0–1	7–0	8–0		2/1/0	2

Group G

- G1: YowAI
- G2: Cyberoos 2000
- G3: GSP
- G4: Lucky Lubeck
- G5: Polytech 100

	G1	G2	G3	G4	G5	W/L/D	Rank
G1		4–0	28–0	18–0	6–0	4/0/0	1
G2	0–4		27–0	13–0	3–0	3/1/0	2
G3	0–28	0–27		0–12	0–16	0/4/0	5
G4	0–18	0–13	12–0		1–0	2/2/0	3
G5	0–6	0–3	16–0	0–1		1/3/0	4

Group H

- H1: 11 Monkeys
- H2: AT Humboldt 2000
- H3: Harmony
- H4: PSI

	H1	H2	H3	H4	W/L/D	Rank
H1		4–0	13–0	10–0	3/0/0	1
H2	0–4		9–0	5–0	2/1/0	2
H3	0–13	0–9		0–3	0/3/0	4
H4	0–10	0–5	3–0		1/2/0	3

Acknowledgements

RoboCup-2000 was sponsored by Sony, SGI, FujiXerox, Australon, and RMIT University. The sections of this article reflect the work of the indicated authors along with Tomoichi Takahashi and the RoboCup-Rescue team (rescue) and Dominique Duhamel (humanoid).

References

1. M. Asada, M. M. Veloso, M. Tambe, I. Noda, H. Kitano, and G. K. Kraetzschmar. Overview of RoboCup-98. *AI Magazine*, 21(1), 2000.
2. J. Baltes and N. Hildreth. Adaptive path planner for highly dynamic environments. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
3. C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalso, and A. Sgorbissa. Communication and coordination among heterogeneous mid-size players: ART99. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
4. S. Coradeschi, L. Karlsson, P. Stone, T. Balch, G. Kraetzschmar, and M. Asada. Overview of RoboCup-99. *AI Magazine*, 21(3), 2000.

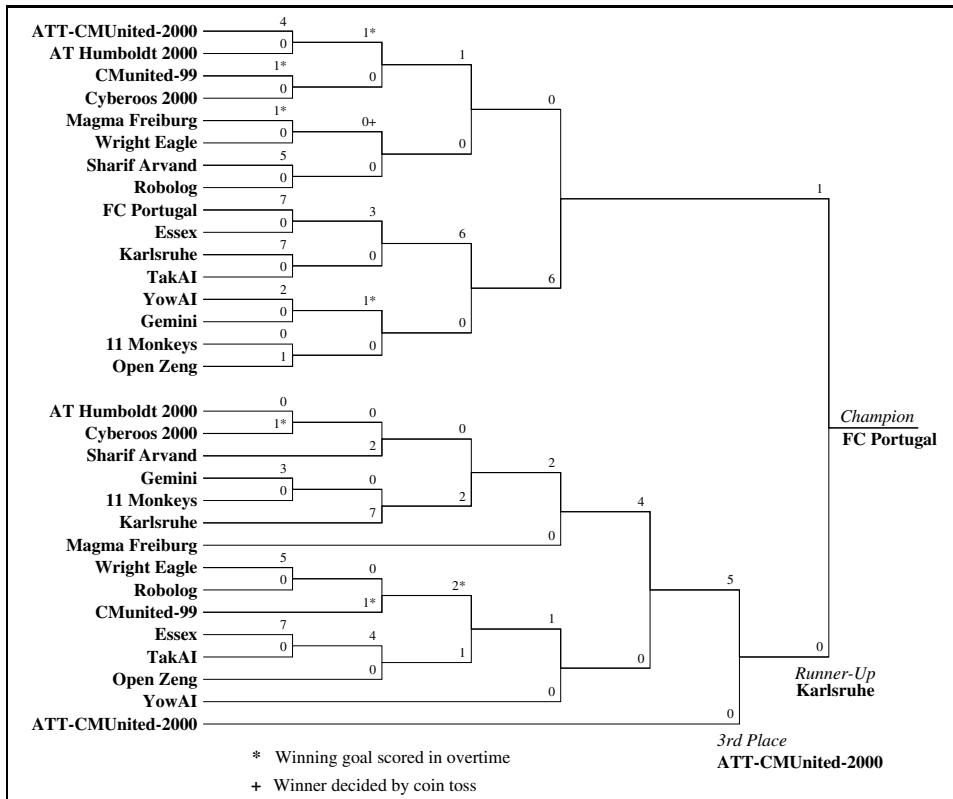


Fig. 12. The simulation league championship tournament.

5. T. Esaki, T. Sakushima, S. Futamase, N. Ito, T. Takahashi, W. Chen, and K. Wada. Kakitsubata team description. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Berlin, 2001. Springer Verlag. To appear.
6. M. Fujita, M. Veloso, W. Uther, M. Asada, H. Kitano, V. Hugel, P. Bonnin, J. C. Bouramoue, and P. Blazevic. Vision, strategy, and localization using the sony legged robots at robocup-98. *AI magazine*, 21(1):45–56, 2000.
7. T. Kaneda, F. Matsuno, H. Takahashi, T. Matsui, M. Atsumi, M. Hatayama, K. Tayama, R. Chiba, and K. Takeuchi. Simulator complex for RoboCup-Rescue simulation project – as test-bed for multi-agent organizational behavior in emergency case of large-scale disaster. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
8. B. Kröse, R. Bogaard, and N. Hietbrink. Programming robots is fun: RoboCup jr. 2000. In *Proceedings of Belgium-Netherlands AI Conference 2000*, 2000.
9. Y. Kuwata and A. Shinjoh. Design of RoboCup-Rescue viewers — toward a real world emergency system. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
10. H. H. Lund. Robot soccer in education. *Advanced Robotics Journal*, 13(8):737–752, 1999.
11. H. H. Lund, J. A. Arendt, J. Fredslund, and L. Pagliarini. Ola: What goes up, must fall down. *Journal of Artificial Life and Robotics*, 4(1), 1999.
12. H. H. Lund and L. Pagliarini. RoboCup jr. with lego mindstorms. In *Proceedings of Int. Conf. On Robotics and Automation (ICRA2000)*, NJ, 2000. IEEE Press.
13. C. Marques and P. Lima. A localization method for a soccer robot using a vision-based omni-directional sensor. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
14. J. Murray, O. Obst, and F. Stolzenburg. Towards a logical approach for soccer agents engineering. In G. K. Peter Stone, Tucker Balch, editor, *RoboCup-2000: Robot Soccer World Cup IV*, LNAI. Springer, Berlin, Heidelberg, New York, 2001. To appear.
15. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
16. I. Noda, S. Suzuki, H. Matsubara, M. Asada, and H. Kitano. RoboCup-97: The first robot world cup soccer games and conferences. *AI Magazine*, 19(3):49–59, Fall 1998.
17. M. Ohta. RoboCup-Rescue simulation: in case of fire fighting planning. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
18. S. Papert. Constructionism: A new opportunity for elementary science education. A proposal to the National Science Foundation, 1986.
19. M. Prokopenko, M. Butler, and T. Howard. On emergence of scalable tactical and strategic behaviour. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Berlin, 2001. Springer Verlag. To appear.
20. M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thaté, and R. Ehrmann. Karlsruhe brainstormers - a reinforcement learning approach to robotic soccer. In P. Stone, T. Balch, and G. Kraetszschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Berlin, 2001. Springer Verlag. To appear.

21. S. Tadokoro and H. Kitano, editors. *The RoboCup-Rescue: A challenge for emergency search & rescue at large-scale disasters*. Kyoritsu Publ., 2000. (in Japanese).
22. S. Tadokoro, H. Kitano, T. Takahashi, I. Noda, H. Matsubara, A. Shinjoh, T. Koto, I. Takeuchi, H. Takahashi, F. Matsuno, M. Hatayama, J. Nobe, and S. Shimada. The robocup-rescue project: A robotic approach to the disaster mitigation problem. In *Proc. IEEE International Conference on Robotics and Automation*, 2000.
23. T. Takahashi, I. Takeuchi, T. Koto, S. Tadokoro, and I. Noda. RoboCup-Rescue disaster simulator architecture. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
24. M. Wünstel, D. Polani, T. Uthmann, and J. Perl. Behavior classification with self-organizing maps. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001. To appear.
25. T. Yamamoto and M. Fujita. A quadruped robot platform with basic software for robocup-99 legged robot league. In *Proceedings of the International Conference on Robotics and Intelligent Systems 2000 (IROS-00)*, 2000.

FC Portugal Team Description: RoboCup 2000 Simulation League Champion

Luís Paulo Reis¹ and Nuno Lau²

lpreis@fe.up.pt, lau@det.ua.pt

¹LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal

²DET – Electronics and Telecommunications Department, University of Aveiro, Portugal

<http://www.ieeta.pt/robocup>

Abstract. FC Portugal is the result of a cooperation project between the Universities of Aveiro and Porto in Portugal. The project started in February 2000 and only three months later, in Amsterdam, FC Portugal became the first European Champion of RoboCup scoring a total of 86 goals without conceding a single goal. Three months later, in Melbourne, FC Portugal became RoboCup Simulation League World Champion scoring 94 goals, again without conceding any goal. This paper briefly describes some of the most relevant research developments and innovations that lead to FC Portugal team success.

1 Introduction

Soccer is a very complex game, both very animated to play and exciting to watch. The soccer server simulator [2], although being a 2D simplified soccer simulation, manages to keep most of this complexity, animation and excitement. Because of this, we argue that, in the simulation league on RoboCup, to be successful and win, a team must be able to play like a real soccer team. Thus, FC Portugal project was conceived as an effort to create intelligent players, capable of thinking like real soccer players and behave like a real soccer team. Expertise from real soccer experts (players, fans and coaches) was gathered and adapted to the specificities of this 2D simulation before creating a team strategy and agent architecture capable of supporting real soccer reasoning.

CMUnited99 publicly available low-level source code [7] was used as a starting point for FC Portugal. This saved a huge amount of time in the beginning of the project and enabled our initial research to be mainly focused on multi-agent cooperation and communication issues while concentrating on some low-level details later on. FC Portugal introduces several research innovations in RoboCup:

- Flexible team strategy composed by tactics, formations (used inside tactics) and player types to be used in different game situations;
- Extensive use of the concept of player type defined at three different behaviors levels (strategic, ball possession and ball recovery);
- Distinction between strategic and active situations;
- Situation based strategic positioning mechanism;
- Dynamic positioning and role (player type) exchange mechanism based on utility functions;
- Intelligent communication based on teammate modeling and on a communicated world state;
- Intelligent perception through a strategic looking mechanism based on utility functions;
- Integration of soccer knowledge in the positioning, ball possession and ball recovery modules;
- Very strong kick based on online optimization;
- An intelligent goal keeping strategy for 2D soccer;

- Marking techniques based on teammate modeling;
- An agent architecture, multi-level world state and high level decision module capable of supporting this approach;
- Visual debugging to show what the agents see, hear, feel, think and do;
- Offline client to repeat and fully debug the execution of an agent;
- World state error analyzer.

Unfortunately due to space limitations it is impossible to fully explain all these issues in this paper and so, some of them are only explained at an overview level. Throughout this article we assume the reader has good knowledge about the RoboCup initiative [3] and soccer server [2] and basic knowledge of artificial intelligence and real soccer.

The structure of the article is as follows. Section 2 describes FC Portugal team strategy, the concepts of tactic, formation and player type, the situation based positioning mechanism and the dynamic positioning and role exchange mechanism. Section 3 is concerned with the agent architecture and the high level decision module, while the next section describes the intelligent perception and communication methods used by FC Portugal. Section 5 describes the integration of soccer knowledge in the agent's individual decision modules and the next section describes some of the low-level skills implemented in our agents. Several development tools implemented in this project are described in section 7 and we conclude the paper with some results and conclusions.

2 Team Strategy

Tactics, formations, positionings and player types are common concepts in soccer and a good RoboCup team must use them in order to be able to play simulated robosoccer. CMUnited brought the concepts of formation and positioning to RoboSoccer [6,9] and used dynamic switching of formations depending on the result and time of the game. We extended this concept and introduce tactics, situations and player types. FC Portugal team strategy definition is based on a set of player types (that define player strategic, ball possession and ball recovery behaviors) and a set of tactics that include several formations (433, 442, Open433, 532, etc.). Formations are used for various game situations (defense, attack, transition from defense to attack, etc), assigning each player a base strategic position and a player type. Figure 1 shows the structure of FC Portugal team strategy.

2.1 SBSP - Situation Based Strategic Positioning

One of the main strengths of FC Portugal is its positioning mechanism based on the clear distinction between strategic and active situations. SBSP mechanism is used for strategic situations (in which the agent believes that it is not going to enter in active behavior soon). For active situations, the agent position on the field is defined by specific ball possession, ball recovery or stopped game decision mechanisms.

To calculate its base strategic position, the agent analyses which is the tactic and formation in use and its positioning (and corresponding player type). This position is then adjusted according to the ball position and velocity, situation (attack, defense, scoring opportunity, etc.) and player type strategic information. Player strategic characteristics include ball attraction, admissible regions in the field, specific positional characteristics for some regions in the field, tendency to stay behind the ball, alignment in the offside line, and attraction by specific points in the field in some situations. Due to lack of space, we present only a simplified SBSP algorithm (that does not consider ball velocity, attraction rectangles

and attraction points and simplifies the use of situations applying them only for choosing the formation):

```
/* Variables: P - Player Positioning, BSP - Base Strategic Position,
PT - Player Type, BallP - Ball Position, SP - Strategic Position */

Vector SBSPosition(Positioning P) {
    Vector BSP = getPosition(Tactic, Formation, P);
    PType PT = getPlayerType(Tactic, Formation, P);
    SP = adjustSBSPPosition(BSP, PT, BallP);
    return SP;
}

Vector adjustSBSPPosition(Vector BSP, PType PT, Vector BallP) {
    SP = BSP + (BallP.X*BallAttraction.X(PT), BallP.Y*BallAttraction.Y(PT));
    If alignsOnDefenseLine(PT) then SP = adjustToDefenseLine(SP, BallP)
    If behindBall(PT) then SP = adjustToBehindBall(SP, BallP)
    If offsidePosition(SP) then SP = adjustToOnside(SP)
    If not inside admissibleRectangle(PT) then
        SP = adjustToAdmissibleRectangle(SP, admissibleRectangle(PT))
    return SP;
}
```

This positioning system enables the team to move like a real soccer team, keeping the ball well covered while players remain distributed along the field.

2.2 Player Types

Soccer teams are heterogeneous in nature, including players with different abilities, performing different roles. Although simulated agents are homogeneous (using soccer server 6.06), using heterogeneous behaviors gives the team greater flexibility. For example, midfielders must be more aggressive players, while central forwards must be good shooters and central defenders must be positional players. FC Portugal uses extensively the concept of player type. Player types are built by defining a set of different characteristics for players that include: strategic, ball possession and ball recovery characteristics.

Strategic characteristics are the SBSP parameters for each player type. For example, a given player type may like to run a lot in the field, taking part in defensive movements but also running to the opponent's area when the game situation is changed to attack. Other

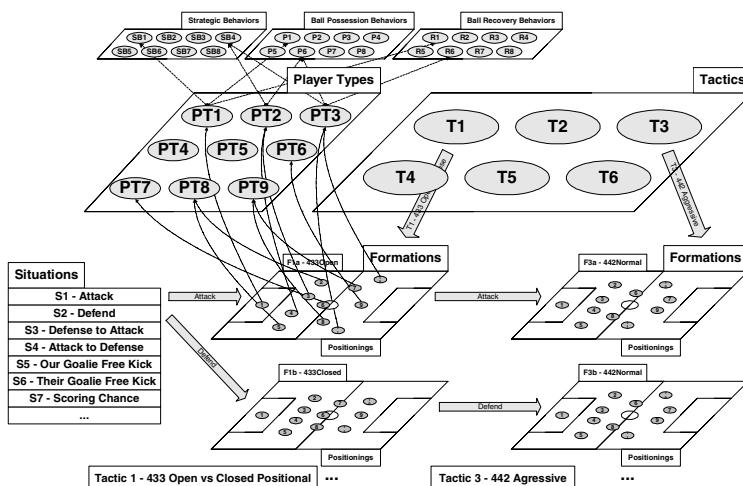


Fig. 1: Team Strategy composed of Tactics, Formations and Player Types

player types may remain strategically more static keeping their energy for active situations. Another example is concerned with the tendency of a player to integrate the defense line (in defensive situations) or to position itself in a possible scoring position (in offensive moves).

Ball possession characteristics concern the player type behavior, in active situations, when it has the ball controlled. These characteristics include the relative tendency for shooting, dribbling, passing, etc. Different weights are used for the evaluation of each of the action types. For example, a safe passer may give a very high weight to the confidence that the receiver is prepared to receive the pass and to the opponent's interception possibilities and low weights for positional gain and shooting possibility values of that pass.

Ball recovery characteristics are used to define the behavior of player types, in active situations, without the ball. They define the tendency of that player type to intercept, mark, mark pass lines, etc. For example, a player with the positional defender player type (used by FC Portugal central defenders) does not mark pass lines or tries risky interceptions. This type of player keeps its strategic position, entering in active ball recovery actions safely.

2.3 DPRE - Dynamic Positioning and Role Exchange

DPRE is based on previous work from Peter Stone et al [6,9] that suggested the use of flexible agent roles with protocols for switching among them. In FC Portugal, players may exchange not only their positionings (place in the formations) but also their player types in the current formation. Positioning exchanges are performed only if the utility of that exchange is positive for the team. Utilities are calculated using the distances from the player's positions to their strategic positions, the importance of each positioning and the adequacy of each player to each positioning in the formation on that situation.

Although spatial coordination is enough to guarantee that positioning exchanges are successful, coordination is reinforced by communication. Triple positioning exchanges and positioning covering (a player covers an unguarded important position in the field) are also possible but were not used on the team in RoboCup 2000. Contrarily to CMUnited that found out that role exchange was not useful on their team and did not use it in RoboCup 99, our experimental results showed that DPRE increases significantly team performance keeping stamina levels higher and the number of uncovered useful positions significantly reduced. FC Portugal with DPRE wins about 85% of the games against the team without DPRE. Also, this mechanism enables the team to play using only 6 or 7 players on the field while winning to every RoboCup 99 available team.

3 Agent Architecture and High Level Decision Module

FC Portugal, team strategy includes the knowledge about tactics, formations, player types, situations, intelligent communication and perception strategies and opponent modeling strategies. These structures are instantiated initially by a human coach according to the opponent team characteristics.

The main control loop of our agents uses perception interpretation and action prediction to update the world state [9] and then use a high-level decision module to decide the next action to do. Information model in FC Portugal is a multi-level structure with data at four levels of abstraction:

- **Global Situation Information** – High-level information like result, time, game statistics (shoots, successful passes, etc.) and opponent behavior, used to decide the team tactic at a given moment;

- **Situation Information** – Information relevant to the selection of the appropriate formation and for the SBSP, SLM and ADVCOM mechanisms;
- **Action Selection Information** – Set of high-level parameters used to identify active situations and to select appropriate ball possession or ball recovery behaviors;
- **World State** – Low-level information, including positions and velocities of players and ball.

The high-level decision module (Fig. 2) decides not only the player action but also the tactic, formation, positioning and player type of the player. This high-level decision module initializes some internal structures and enters the agent main control loop. This loop starts by running the STRATEGY module to decide the current tactic and formation. Then, it analyses DPRE possibilities, performs intelligent communication and decides where to look. If the game is stopped, stopped game positioning is performed. If an active situation is not identified SBSP is performed, otherwise if the agent has the control of the ball, an appropriate ball possession action is selected and if not, an appropriate ball recovery action is selected. The loop closes with modules not directly connected with decision-making: action execution and multi-level world state update.

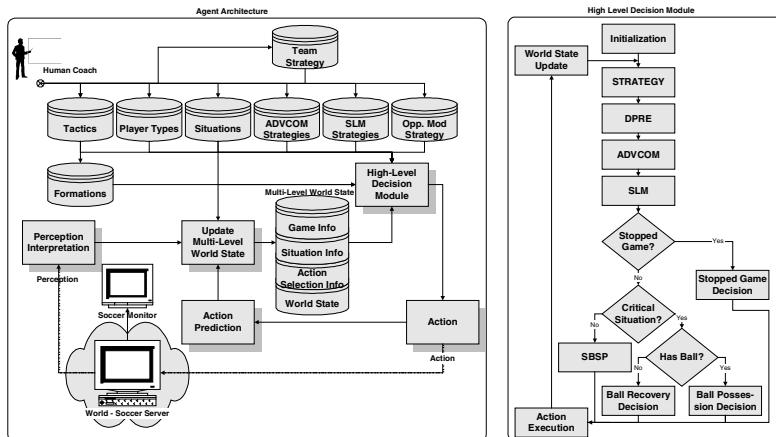


Fig. 2. Agent Architecture and Control Flow of the High Level Decision Module.

4 Intelligent Perception and Communication

FC Portugal low-level world state model is based on the world state model from CMUnited99 source code [7]. This model keeps track of object positions and velocities in field coordinates assigning a confidence value to each of these values. Position confidences in CMUnited99 source code are assigned their greatest value (1.0) when the object is seen and then decay with time (in cycles). FC Portugal position confidences are also linked with the precision a player could get when he saw the object. When a player sees an object, he assigns it a confidence that decays with that object distance to the player. Soccerserver visual info precision degrades with this distance so this is reflected in the position confidences. A similar reasoning is performed for velocity confidences. This kind of confidence assignment is especially useful when players share their world state knowledge through communication.

4.1 ADVCOM - Intelligent Communication Mechanism

Communication in single channel, low bandwidth, and unreliable domains should only be used when it is useful from a team global point of view. The main research challenges are concerned with deciding what and when to communicate. FC Portugal agents use communication in order to:

- Maintain agent's world states updated and precise by sharing individual world states at appropriate moments;
- Increase team coordination by communicating useful events (for example a pass or positioning swap).

When communication is performed, FC Portugal players communicate their complete world state and eventually some higher-level events. The main innovation of our communication strategy is related with the use of teammate modeling techniques and event information (like ball velocity changes), to evaluate the communication utility and decide whether or not to communicate. Agents communicate when they believe that the utility of their communication is higher than those of their teammates. When a given player witnesses a useful event, he reasons if he should communicate that event or if other teammates are in better conditions to do it. If no useful event (e.g. ball drop, ball velocity change, pass, etc.) is detected, players' reasoning is concerned with the amount of useful information that their world state may contain to the teammates. To enable this kind of reasoning, agents maintain a "communicated world state". Communicated world state is updated using only the aural information, without using any other perception or action prediction information. Players use this communicated world state as an indication of their teammates knowledge.

Communication utility is calculated by adding the confidence increments of the world state object positions relatively to the corresponding confidences in the communicated world state. The same type of reasoning is applied to the ball confidence values. After a player communicates, he will not communicate for a while (because his world state is similar to the communicated world state).

4.2 SLM - Strategic Looking Mechanism

In a complex domain such as soccer, sensors must be coordinated and used in an intelligent way to enable an agent to maintain an accurate world state. However, the interest of a given part of the world state is different according to the situation. If an agent has the ball controlled and is near the opponent's goal, then the position and velocity of the opponent's goalie is a very important part of the world state, while the positions of the players near the middle of the field have reduced interest.

FC Portugal SLM intelligent perception mechanism decides the agent looking direction according to the player type, formation, player position and with the confidence it has on the positions and velocities of all the objects in the field. For each of the possible looking directions, a utility measure is calculated that estimates the usefulness for the agent to look on that direction. This utility is based on the world state confidence increment expected by looking on that direction. When evaluating the confidence increment, players use different weights for different objects depending on the situation, ball position and self position. The best looking direction is then selected and an appropriate turn neck command is issued.

5 Integrating Soccer Knowledge

Most of the player's individual decisions are based on soccer knowledge adapted to the 2D simulation. This knowledge includes the way to decide what to do with the ball (pass, shoot, forward, dribble or hold), without the ball in active situations (intercept, mark opponent, mark pass line, approach the ball, cover the goal, get free, etc.) and in stopped game situations. Also, the definition of tactics, formations and players' strategic behavior strongly considers real soccer playing and coaching expertise.

5.1 Ball Possession

If a player has the ball its decision is based on the evaluation of the following options:

- **Shoot:** Shoot to a given spot in the opponent's goal. Hard shoots and fast shoots are considered for different scoring opportunities;
- **Pass:** Pass the ball to a teammate to a given position. Passes are assumed to be fast (although several velocities are considered) and made to a position near the receiver;
- **Forward:** A forward pass sends the ball to a point in front of a given player so that the player may run and get the ball ahead. While normal pass receptions are planned so that the ball is still moving quickly, forward passes are planned so that at the reception point the ball is almost stopped. This ball behavior allows a forward to be done to a point that is far from the receiving player if that player will be the first to get there.
- **Dribble:** Advance in the field, to a given position, keeping the ball always controlled;
- **Hold:** Keep the ball controlled, avoiding opponents without moving.

The best shoot, pass, forward, dribble and hold are compared to compute the final decision. To evaluate each of the possible options, several evaluation measures, with different weights, are used. For example, to evaluate a pass, 12 different evaluation measures, are used (including the positional gain, destination spot congestion, possibility of opponent interception, etc.). These measures are calculated using the action selection information in a qualitative scale {VGood, ..., VPoor} and are combined using different weights. A maximum of 72 pass options (at 5 degree interval) are evaluated. In most circumstances, backward passes are not considered.

5.2 Ball Recovery

Players should select this behavior when they detect that ball recovery is possible, or when they find themselves in a critical defending situation where, although ball recovery is not probable immediately, they find it advantageous for the team (for example marking a pass line or covering the goal). The ball recovery behaviors implemented in FC Portugal are:

- **Ball Interception** – Run to the fastest possible ball interception point;
- **Ball Passive Interception** – Intercept the ball, not in the fastest point but in a more advantageous point, although taking more time;
- **Mark Pass Line** – Mark a pass line from the opponent that is (or will be) in control of the ball, to another opponent. Marking is performed if a player detects a useful and uncovered passing line that may be well marked (better than any other teammate) near that player strategic position;
- **Approach Ball Position** – Player approaches ball position in order to reduce opponent options;
- **Mark Opponent** – Mark the player that has the ball keeping him from advancing in the field;
- **Cover Goal** - Player tries to get a good defensive position by keeping his position between ball position and its own goal.

The process of selecting between these actions is performed based on several characteristics of the current situation and of the high level world state using an extensive set of rules. The goalie defensive strategy is based on five actions: mark position, passive interception, active interception, goal interception and catch. FC Portugal Goalie strategic position is usually quite near the limits of the penalty area

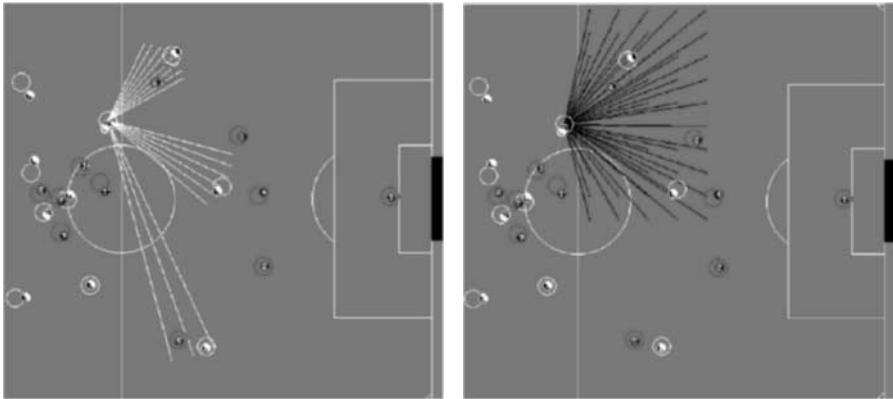


Fig 3: An example of Passes and Forwards considered for evaluation.

6 Low-Level Skills – Optimization Kick

Since FC Portugal research is not focused on low-level skills and there is very good source code for low-level skills available on the Internet, most of our skills are adapted from CMUnited99 source code [7]. However, and since this is a crucial skill, we implemented a new strong kick based on optimization techniques.

The probability of success of players' actions depends heavily on the ball handling ability of players. Players should be able to kick the ball with the fastest possible speed. Soccerserver, allows the possibility of a player to kick the ball in successive cycles. If this possibility is efficiently exploited, players can provide the ball with a greater speed than a single kick would permit. However, the problem of finding these accelerative kicking sequences is not trivial. Several approaches have been proposed either analytically [7] or by using learning techniques [1,5]. We modeled the problem as an optimization problem and applied some simple (but effective) search algorithms in order to find these kick sequences. Optimization is executed online, during the game, for each particular situation each time the player decides to kick the ball powerfully.

Online search is divided in two stages. The first stage uses a random search of kick sequences to select the "best" kick sequence while the second stage tries to improve the kick sequence found in the previous stage.

Our random search uses the following algorithm. Random kicks are performed until the ball gets out of the kickable area. When this happens, the last kick is replaced by an analytically determined kick to adjust the final angle. Each kick sequence is analyzed by an evaluation function that considers the final speed of the ball, the kick sequence length and the number of cycles in which opponents may also kick the ball. The kick sequence with the best value is selected. Simple heuristics are applied in order to conduct the search to good solutions.

After random search, the best kick sequence is improved using a hill-climbing algorithm. This sequence is disturbed by random noise and if the resulting kick sequence is better it is considered as the new best kick sequence. When it improves the result, the last disturbing noise is reused in our hill climbing because that direction in the search space should be tested again. Hill climbing improved results taking an insignificant time (below 0.5 ms for Pentium 200MHz). Without using the limits of the kickable area (to account actuators and sensors noise) this approach allowed us a final speed median of 2.47 taking 3 or 4 cycles (over 2000 tests) for a typical case of ball stopped in front of a player.

Online search of the best kick sequence based on optimization techniques is a novel approach to the problem of kicking the ball powerfully in simulated RoboSoccer. This approach enabled FC Portugal players to kick the ball in any direction (without turning) taking few simulation cycles and quite powerfully.

7 Development Tools

7.1 Visual Debugger

CMUnited99 introduced the concept of layered disclosure [8] and extended the functionality of the logplayer application (distributed with the soccerserver) to support it. Their extended logplayer included the possibility of synchronous visualization of the game (using soccermonitor) and of one of the players reasoning (at several levels of abstraction) saved in action logfiles. We have integrated the two applications (logplayer with layered disclosure and soccermonitor) in a powerful team debugging tool which includes several new features.

Visual information is much more easily handled than text information. Soccermonitor [2] includes the possibility of drawing points, lines and circles over the field, but this functionality is not reported as being used by other teams. The possibility of drawing over the field provided by soccermonitor was exploited, modified and extended (possibility of drawing text over the field). On the other hand, we extended the log action files syntax so



Fig. 4 : Integrated debug application window

that they could include draw commands. As with action log messages, draw commands can be inserted at different reasoning levels of abstraction. Normal action debug messages still exist but they are shown in scrollable debug text windows.

The debugger determines the real world state based on the visualization information saved in server record log files. The real positions and velocities of all objects are calculated and probable last player commands are deduced from a set of simple rules that reason on world state changes. This internal view of the real world state is shown in a text window. Some features (ball position/velocity, selected player position/velocity and ball distance) are shown directly over the field.

We used the possibility of drawing text over the field quite extensively to show player action mode, synchronization, basic action, lost cycles, evaluation metrics, position confidences and to compare players beliefs on its (and ball) position/velocities with real values. Drawing circles and lines over the field was used to show player beliefs on object positions, player view area, best pass, shoot or forward, communication events, etc (Fig 4).

We have used this tool to tune strategic positions, test new behaviors, tune the importance and precision of each of the evaluation metrics used in the high-level decision module, test world update, communication and looking strategies empirically, analyze previous games of other teams, etc.

7.2 Offline Client

In some situations the action log files created by our players were not enough to understand what was really happening in a certain situation. A much finer debugging degree can be achieved by employing our offline client tool. The principle of the offline client is that we can repeat the execution of a player over exactly the same setting of a previous game without the intervention of the server and without real-time constraints. Then we can use a normal debugger (like gdb) to examine the contents of all variables, set breakpoints, etc at the execution situations we want to analyze. A special log file, that records every interaction with the server and the occurrence of timer signals, must be generated to use the offline client. If an agent has probabilistic behavior some more information might be needed (the only probabilistic behavior of our agents is the optimum kick). The offline execution of an agent is achieved through a stub routine that reads previously saved server messages from the log file instead of reading network messages. Player's behavior is maintained, as it is not affected by the substitution of this stub routine. The execution of a normal debugger over this offline client allows a very fine tracking of what happened.

8 Results and Conclusions

Although being a very recent team, FC Portugal won the European RoboCup 2000, held in Amsterdam (May 29- June 2), and the World RoboCup 2000, held in Melbourne (August 28 - September 3). In these two competitions, FC Portugal scored a total of 180 goals, without conceding a single goal. Table 1, summarizes FC Portugal's results in both competitions.

In Amsterdam FC Portugal used almost unchanged CMU basic skills. However, the team beat easily other teams with much better low-level skills (e.g. Essex Wizards [4] with an excellent dribbling, Brainstromers [5] with an amazingly powerful kick). The reasons for this (apparently strange) success were:

- **Team strategy**, based on very flexible tactics with well conceived formations for different game situations and flexible player types;

- **Situation based strategic positioning** that (enhanced with DPRE) enabled the team to move in the field as a real soccer team;
- **Individual decision modules** (ball possession and ball recovery) that enabled the agents to think like real soccer players. Knowing in advance that their low level skills were inferior they were able to play with that limitation;

FC Portugal could not score many goals, even against some medium quality teams. This was due to the poor low level skills and imprecise world state of the agents that prevented the team to play a faster type of game. With this limitation, FC Portugal used the SBSP mechanism combined with the individual decision modules to explore the free space on the field to attack and to cover that same free space while defending. The goals were almost always scored going through the (usually uncovered) side wings of the opponents and crossing to a point near the opponent's goalie where a FC Portugal player could easily score with a light kick. Against teams that had good positional systems or that defended with a huge amount of players inside their penalty box, this playing type revealed to be less effective.

Table 1: Scores of FC Portugal in EuRoboCup2000 and RoboCup2000.

Euro RoboCup – Amsterdam	Score	RoboCup 2000 - Melbourne	Score
Essex Wizards (England)	3 – 0	Oulu2000 (Finland)	33 - 0
Lucky Luebeck (Germany)	13 – 0	Zeng2000 (Japan)	18 - 0
Cyberos (Australia)	4 – 0	Robolog (Germany)	20 - 0
Pizza Tower (Italy)	22 – 0	Essex Wizards (England)	7 - 0
Polytech (Russia)	19 – 0	Karlsruhe Brain. (Germany)	3 - 0
PSI (Russia)	6 – 0	YowAI (Japan)	6 - 0
Wroclaw (Poland)	13 – 0	ATTCMUnited2000 (U.S.A)	6 - 0
Essex Wizards (England)	5 – 0	Karlsruhe Brain. (Germany)	1 – 0
Karlsruhe Brain. (Germany)	2 – 0	Total Score	94 - 0
Total Score	86 – 0		

In Melbourne, several teams used positioning mechanisms similar to SBSP, (mostly the ones that competed in Amsterdam). So, the positional advantage, FC Portugal had in Amsterdam, was only visible in games against weaker teams. In the games against very good teams, FC Portugal superiority was related, not only with the team strategy, SBSP, DPRE and decision modules, but also with:

- **Intelligent perception and communication**, that enabled the players to maintain a very precise world state but at the same time keeping alert to useful events (like ball velocity changes);
- **Optimization kick**, that enabled the players to kick the ball in a very fast and powerful way;
- **Marking techniques**, responsible for not letting the other teams play their normal game because almost all useful pass lines were always marked by a FC Portugal player;
- **Debugging tools**, which enabled to tune all FC Portugal configurable strategies and parameters in a very easy and robust way.

The development of a strong kick and the availability of a much better world state, enabled FC Portugal to change its type of playing. Instead of the medium velocity passes, progression by the wings and crosses to players near the opponent's goal, it was now possible to do very fast passes in the middle of the field and to shoot with success from outside the penalty box. FC Portugal playing game became much more flexible. With teams with lots of players in the center of the field, the team plays identically to the Amsterdam team. But against other types of teams, FC Portugal adjust its playing mode. Several teams used very defensive strategies with lots of players inside the penalty box, against FC

Portugal. However, that did not prevent FC Portugal to adapt its flexible strategy to that crowded type of defense and to easily score 6 or more goals and have lots of other scoring opportunities.

We believe that the Seattle simulation competition is going to finally be very much like a real soccer tournament. In Melbourne, the games between FC Portugal and Karlsruhe Brainstorms seemed just like real soccer games. Since, several teams made available their low level skills or even their complete source code we believe that at the low-level, the best teams are going to be very tight. At the team and individual level, intelligent perception and communication, SBSP and DPRE (or similar concepts) and marking techniques are surely going to be used by the best teams. Although the quality of the individual players (their low-level skills and mainly their individual decision making) will continue to be crucial, the winner of RoboCup 2001 will be most likely decided on the bench. The quality of offline and online coaches that define the team strategy and analyze the opponents, changing that strategy accordingly (different tactics, formations and player types), being one of the major distinctive factors! Just like in real soccer...

Acknowledgments

The authors would like to thank to Luis Seabra Lopes for having this (wonderful) idea of participating in RoboCup 2000 and for his excellent contribution on all administrative, promotional and funding work of FC Portugal as well as fruitful discussions on several topics. Our thanks goes also to Peter Stone, Patrick Riley and Manuela Veloso for making available the CMUnited99 low-level source code that saved us a huge amount of time at the beginning of the project. We would also like to thank the financial support from FCT - Portuguese Foundation for Science and Technology, Compaq Portugal, PT Innovation, LIACC, University of Aveiro and APPIA – Portuguese Association for Artificial Intelligence.

References

1. Mihal Badjonski, Kay Schroeter, Jan Wendler, Hans-Dieter Burkhard. Learning of Kick in Artificial Soccer. *Proc. of Fourth Int. Workshop on RoboCup*. Melbourne, August 2000
2. Emiel Corten et al. *Soccerserver Manual, Version 5 rev 00 beta.*, At URL <http://www.dsv.su.se/~johank/Robocup/manual>, July 1999
3. Hiroaki Kitano. RoboCup: *The Robot World Cup Initiative, Proceedings of the 1st International Conference on Autonomous Agent (Agents-97)*, Marina del Ray, The ACM Press, 1997.
4. Kostas Kostiadis and H. Hu, A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition, In Veloso M., Pagello E., and Kitano H., editors, *RoboCup-99: Robot Soccer World Cup III*. pp. 366-377 Springer Verlag, Berlin, 2000
5. Martin Riedmiller et al. Karlsruhe Brainstormers 2000 - A Reinforcement Learning approach to robotic soccer. *Proc. of Fourth Int. Workshop on RoboCup*. Melbourne, August 2000
6. Peter Stone and Manuela Veloso. Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for RealTime Strategic Teamwork. *Artificial Intelligence*, 110 (2), pp. 241-273, June 1999.
7. Peter Stone, Patrick Riley and Manuela Veloso. CMUnited-99 source code, 1999. Accessible from <http://www.cs.cmu.edu/~pstoney/RoboCup/CMUnited99-sim.html>.
8. Peter Stone, Patrick Riley and Manuela Veloso. Layered Disclosure: Why is the agent doing what it's doing?, *Agents 2000, Fourth Int. Conf. on Autonomous Agents*, Barcelona, June 2000
9. Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD Thesis, Computer Science Dep., Carnegie Mellon University, December 1998

The Cornell Robocup Team

Raffaello D'Andrea¹, Tamás Kalmár-Nagy², Pritam Ganguly² and Michael Babisch³

¹ Sibley School of Mechanical and Aerospace Engineering
Ithaca, NY 14853, USA
rd28@cornell.edu

² Department of Theoretical and Applied Mechanics
Ithaca, NY 14853, USA
{nagy,pritam}@tam.cornell.edu

³ Department of Computer Science
Ithaca, NY 14853, USA
mjb33@cornell.edu

Abstract. This paper describes the Cornell Robocup Team, which won the RoboCup-2000 F180 championship in Melbourne, Australia. The success of the team was due to electro-mechanical innovations (omni-directional drive and a dribbling mechanism) and the control strategies that rendered them effective. As opposed to last year's "role-based" strategy, a "play-based" strategy was implemented, which allowed us to make full use of the robot capabilities for cooperative control.

1 Introduction

The RoboCup competition is an excellent vehicle for research in the control of complex dynamical systems. From an educational perspective, it is also a great means for exposing students to the systems engineering approach for designing, building, managing, and maintaining complex systems.

In an effort to shift the current emphasis of the competition away from simple strategies to more complicated team-based strategies, the main emphasis of this year's team was to play a controlled game. In other words, in a game without ball control, effective strategies essentially consist of overloading the defensive area during a defensive play (the so called "catenaccio" in human soccer, a strategy that is very effective, if not extremely dull and frustrating for the spectators), and shooting the ball towards open space or the goal area in the opponent's half during offensive plays. This was, in fact, the simple role based strategy adopted by our championship team in 1999, which was shown to be extremely effective. In order to bring coordination and cooperation to the RoboCup competition, the Cornell team developed two electro-mechanical innovations (omni-directional drive and dribbling, described in Section 2) *and* the associated control strategies that rendered them effective (described in Sections 3 and 4).

The key system concept employed by the Cornell RoboCup team is that of *hierarchical decomposition*, the main idea of which is depicted in Figure 1.

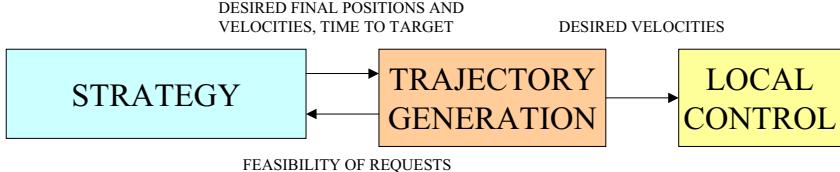


Fig. 1. Hierarchical decomposition

At the lowest level is the local control, which physically occurs on the robots; local feedback loops regulate the wheel velocities about the desired wheel velocities commanded by a centralized computational unit (a workstation). The desired wheel velocities are generated by the Trajectory Generation block; these velocities are generated in such a way to ensure that the robots are physically capable of executing the maneuvers (as limited by power, friction, and stability constraints). The Trajectory Generation block uses desired final position, velocity, and time to target information, generated by the Strategy block, to calculate the desired robot trajectories; feasibility information is sent back to the Strategy block to ensure that only executable commands are sent to the actual robots. The reader is referred to [3] for details on how this general approach can be used to control complex systems.

This decomposition structure renders the real-time vehicle coordination problem tractable. In particular, the high level strategy (described in Section 4) does not need to consider the details of vehicle motion, except for feasibility information; if a desired play is not deemed feasible (for example, leading a pass to another robot is not deemed feasible because the receiving robot cannot execute the interception), alternate strategies are implemented.

The RoboCup competition is relatively mature, with many teams competing at a high level; we have thus decided to emphasize in this paper aspects of our system that we feel are innovative, and forgo a detailed explanation of the sub-systems that are well understood and documented in previous RoboCup publications; interested readers are referred to [5], [1], and [10]. The rest of the paper is organized as follows: in Section 2, a brief description of omni-directional drive and dribbling is presented, followed by an in-depth discussion of the Trajectory Generation scheme in Section 3. A description of team strategy is presented in Section 4.

2 Omni-Directional Drive and Dribbling

A very effective way of position control was implemented by the Cornell Robocup Team in the 2000 competition. The omni-directional drive consists of three pairs of wheels (see Figure 2). Each pair of wheels has an active degree of freedom in the direction of the rotation of the motor and a passive one perpendicular

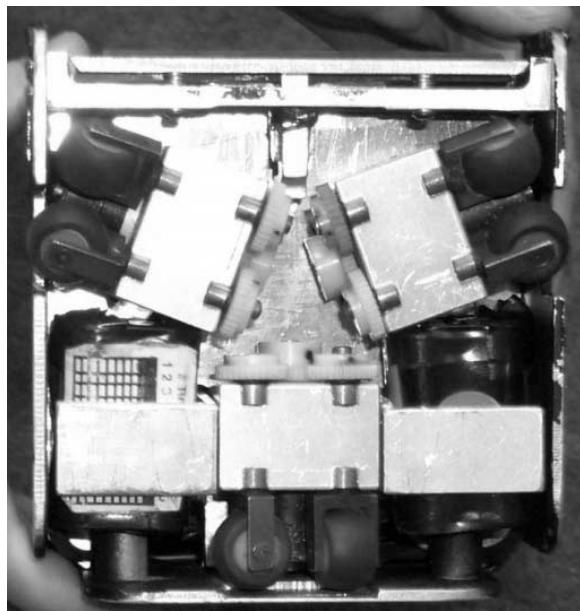


Fig. 2. Bottom view, omni-directional drive

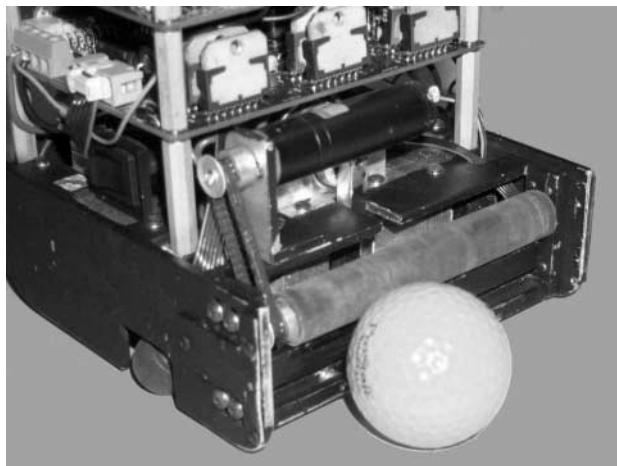


Fig. 3. Dribbling mechanism

to it (see [4], for example). Since the drive directions are pair-wise linearly independent, the translation and the rotation of the robot can independently be controlled by a judicious choice of drive velocities.

The dribbling mechanism is a rotating bar with a latex cover placed just above the kicking mechanism (see Figure 3). Upon contact with the ball, the rotation of the bar imparts a backward spin on the ball; the bar is strategically placed such that the net component of the force on the ball is always towards the robot, and this is achieved without violating the 20 % convexity rule.

The omni-directional drive, coupled with the dribbling mechanism, greatly increases the *potential* capabilities of the robots (we stress the word potential, since it is not obvious that a real-time control strategy can be developed to fully utilize these features). The main capability which is rendered possible by this combination is the effective receiving of passes, which must be central to any sophisticated team-based strategy.

3 Trajectory Generation and Control

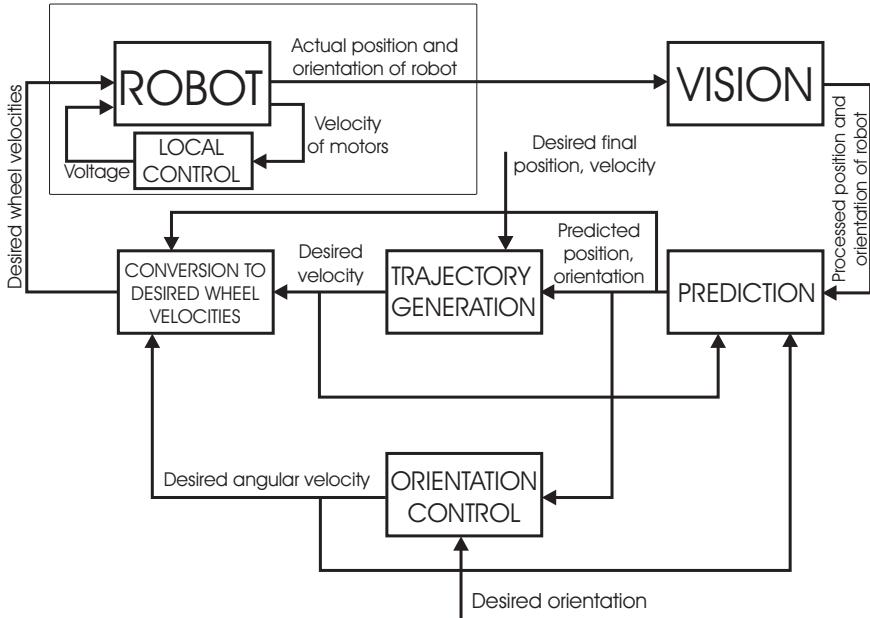


Fig. 4. Block diagram representation of trajectory generation

The overall trajectory generation and control scheme for one robot is depicted in Figure 4. The Vision block feeds the calculated position and orientation of the

robot to the Prediction block, which estimates the position and orientation of the robot *in the robot temporal frame* based on the vision data and the history of the commanded velocities. The Trajectory Generation block solves a relaxation of an optimal control problem to calculate the future robot velocity profile required to reach the prescribed final position and final velocity in either the shortest possible time, or in a prescribed amount of time using the least amount of control effort; a similar step occurs for the robot orientation. The dynamics of the motors and the robots are taken into account to ensure that the generated velocity profiles are feasible. This data is then converted to wheel velocities, which is sent to the robot via wireless communication. A local control loop on the robot regulates the actual wheel velocities about the desired wheel velocities. Note that the overall system feedback is achieved by solving an optimal control problem for every robot at every frame (approximately 30 times per second).

3.1 Feasible Trajectory Generation

As described in Section 1 and depicted in Figure 1, desired wheel velocities are calculated by the Trajectory Generation block based on desired final position and velocity information passed down by the Strategy block. Two types of trajectories are generated: minimum time and minimum control effort. Minimum time trajectories are called upon when a robot must move to a given location, with a given final velocity, as quickly as possible; minimum control effort trajectories are called upon when a robot must arrive at a given location, with a given final velocity, at a given time. A minimum control effort trajectory is clearly not feasible if the minimum time counterpart takes longer to execute.

The full equations of motion of an omni-directional vehicle are complex enough to render the real-time computation of optimal trajectories impossible with today's computational resources. In addition, it is desirable for feasible trajectories to be generated very quickly to allow high-level strategies to explore as many scenarios as possible. Our approach is based on relaxation: can a simplified, computationally tractable optimal control problem be solved whose solution yields feasible, albeit sub-optimal, trajectories? The omni-directionality of our robots permits a positive answer to the above question. The basic idea is to decouple the three degrees of freedom (two translational and one rotational), to solve an optimal control problem for each degree of freedom independently.

Each degree of freedom can be captured by the following differential equation:

$$\ddot{z} = \alpha u - \beta \dot{z}, \quad (1)$$

where z is the displacement or rotation, u is the effective voltage, and α and β are physical parameters which can be calculated from the physical characteristics of the robots and motors. Note that equation (1) captures, to a very high level of accuracy, the dynamics of a single motor driving a load [6]. The problem becomes a constrained boundary value problem when the initial position, initial velocity, final position, and final velocity are prescribed:

$$z(0) = z_0, \quad z(t_F) = z_F, \quad \dot{z}(0) = v_0, \quad \dot{z}(t_F) = v_F \quad (2)$$

subject to the physical constraint $|u| \leq u_{max}$. Since α and β are fixed, we can introduce new length and time scales and assume, without loss of generality, that $\alpha = \beta = u_{max} = 1$. In this non-dimensional form, the maximum achievable velocity has magnitude 1, achieved asymptotically when $\ddot{z} = 0$. We will thus assume that $|v_0| < 1$ and that $|v_F| < 1$.

When performing a minimum control effort maneuver, the decoupling of the degrees of freedom does not pose a problem, since a time of execution is specified. When performing a minimum time maneuver, however, the degrees of freedom will invariably take different amounts of time to execute. This is easily solved by executing a minimum time trajectory for the *slowest* degree of freedom, and executing minimum control effort trajectories for the remaining degrees of freedom.

The minimum time and minimum control effort solutions are provided below. The amount of time required to calculate a trajectory for one robot was on the order of 1000 FLOPS (floating point operations), which is an extremely small number. In particular, as of the writing of this paper a typical desktop computer can execute upwards of 100 MFLOPS per second; at a sampling rate of 30 times per second, the computational resources required to calculate one robot trajectory amounts to less than 0.03 percent of the time allotted to execute one complete system cycle.

Minimum Time: The minimum time problem consists of finding a velocity profile $u(t)$ such that the boundary conditions in (2) are satisfied for as small a t_F as possible. It can be shown (see [7], for example), that the boundary value problem always has a solution, and that the voltage profile which minimizes time t_F consists of a piecewise constant $u(t)$ composed of two segments of magnitude 1. This type of control strategy is commonly referred to as “bang-bang” control. In particular, the following must be solved for U , t_1 , and t_2 :

$$\ddot{z} + \dot{z} = U, \quad 0 < t \leq t_1 \quad (3)$$

$$\ddot{z} + \dot{z} = -U, \quad t_1 < t \leq t_1 + t_2 = t_F, \quad (4)$$

where U is either 1 or -1. Subject to the constraints in (2), we can solve the above to yield

$$t_1 = t_2 - \frac{C}{U}, \quad \exp(t_2 - \frac{C}{U}) = \frac{v_0 - U}{(v_F + U) \exp(t_2) - 2U}, \quad (5)$$

where $C := z_0 + v_0 - z_F - v_F$. The second expression is equivalent to a quadratic equation in $\exp(t_2)$, with solution:

$$\exp(t_2) = \frac{1 \pm \sqrt{1 - \exp(C/U)(1 + v_F/U)(1 - v_0/U)}}{1 + v_F/U}. \quad (6)$$

Four different cases must be considered (two for U , and two for equation (6)). A solution with $t_1 > 0$ and $t_2 > 0$ always exists; the one which yields a minimum value of $t_1 + t_2$ is chosen. Note that once U and t_1 are determined, a closed form solution for $\dot{z}(t)$ can be constructed for $0 \leq t \leq t_1 + t_2$. The computational effort required to solve for t_1 , t_2 , and U is less than 200 FLOPS.

Minimum Control Effort: The minimum control effort problem is based upon the minimum time solution. In particular, let $t_F^* = t_1^* + t_2^*$ denote the time required to execute the minimum time trajectory. It can be shown that for all $t_F > t_F^*$, there exists $t_1 > 0$, $t_2 > 0$, and $0 < \gamma \leq 1$ such that equations (5) are satisfied with $t_F = t_1 + t_2$ and $|U| = \gamma$. In other words, a bang-bang trajectory with lower control effort can always be constructed such that the desired state is reached in the prescribed amount of time. Unlike the minimum control effort problem, a closed form solution cannot be constructed; a simple iterative scheme can be used, however, to solve for t_1 and t_2 which executes in less than 1000 FLOPS.

A second scheme for generating fixed time trajectories can be employed if 1000 FLOPS is too computationally expensive. The equations of motion can be expressed in first order form as follows $\dot{x} = Ax + Bu$, where $x = (z, \dot{z})$. It can readily be shown (see [2], for example), that for a given t_F , the control strategy $u(t)$ which minimizes the 2-norm of u is given by

$$u(t) = B^* \exp(A(t_F - t)) W^{-1} (x_F - \exp(At_F)x_0), \quad (7)$$

where

$$W = \int_0^{t_F} \exp(A\tau) BB^* \exp(A^*\tau) d\tau. \quad (8)$$

We can then solve for x as follows:

$$x(t) = \exp(At)x(0) + \int_0^t \exp(A(t-\tau)) Bu(\tau) d\tau. \quad (9)$$

The above expression can be solved analytically for $x(t)$; the second component of $x(t)$ yields the desired velocity profile $\dot{z}(t)$. The computational effort required to solve for $\dot{z}(t)$ is less than 100 FLOPS. Note however, that the minimum 2-norm trajectory need not be feasible; in particular, there may exist a t such that $|u(t)| > 1$, which violates the actuator constraints. This was not found to be a problem in practice, however, and was the algorithm used in the competition.

3.2 Latency Determination

An integral component of the trajectory control is the Prediction block; this block constructs the optimal estimate of the robot position and orientation in the robot temporal frame. In order to make this estimate, the overall latency of the system must be known precisely. A simple and robust method was devised to accurately estimate the overall system latency, described below.

Consider the angular control of a robot:

$$\dot{\theta}(t) = v_\theta(t), \quad (10)$$

where $v_\theta(t)$ is the commanded angular rate. When the commanded rate is set to a multiple of the observed angular position, the following expression is obtained:

$$\dot{\theta}(t) = -k\theta(t - \tau), \quad (11)$$

where τ is the overall system latency. Consider solutions of the form $\theta(t) = ce^{\lambda t}$, where λ is a complex number. This yields the following characteristic equation:

$$\lambda + ke^{-\lambda \tau} = 0 \quad (12)$$

The locations of the (infinitely many) roots λ determine system stability (negative real parts correspond to stable behavior). The motion is oscillatory (with a decaying or growing envelope) and the change from stable to unstable behavior will take place as a root crosses the imaginary axis, i.e., $\lambda = i\omega$ (see [9], for example). This yields

$$\omega = k, \quad \tau = \frac{\pi}{2k}. \quad (13)$$

This leads to the following method for estimating the system latency: find the value of k for which oscillations neither grow nor decay. The observed frequency of oscillation should be k radians per second, and the system latency $\frac{\pi}{2k}$. For our system, it was indeed observed that the frequency of oscillation was equal to k , validating the approach. The system latency was calculated to be 0.12 seconds.

3.3 Obstacle Avoidance

The Cornell team was able to implement effective obstacle avoidance mainly due to the hierarchical decomposition of overall robot control into trajectory generation and the higher level strategy algorithms. The algorithm we used was developed for circular obstacles (the robots are approximated by their circumscribing circle), and is similar to the one presented in [8]. Even though the obstacles are treated as static, the algorithm works well since it is run every frame. Extension of this algorithm to include instantaneous velocities of obstacles (both friendly and opponent) is straightforward.

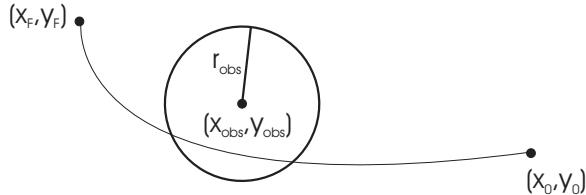


Fig. 5. Trajectory intersecting an obstacle

If the trajectory from the starting point (x_0, y_0) to the final point (x_F, y_F) intersects a single circular obstacle, the path has to be modified (Figure 5). In this case, the algorithm generates two new paths, with via-points located on lines perpendicular to the tangents to the circle from the starting point (Figure 6). The best path is then chosen (for example, the faster one). A straightforward extension of the above algorithm is used for multiple obstacles.

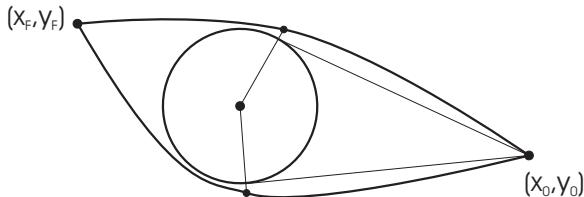


Fig. 6. Avoiding a single obstacle.

4 Strategy

The effectiveness of Cornell's trajectory generation system made it much easier to develop the high level strategy. The design only needs to choose a destination for each robot, and trust that the low level control can move the robot towards that destination. This year, Cornell abandoned its "role-based" strategy in favor of a "play-based" strategy. The new system is a finite state automaton based on a playbook, as depicted in Figure 7. At any point during a game, the robots are involved in a specific play (such as Offense, Defense, or PassPlay) and have specific goals to carry out. The main advantage of this design is that it was extremely easy to develop and debug, and very easy to adapt during the competition when we discovered problems with our strategy. The main disadvantage is that it is dependent on human intuition for its strategy - the system had no way to adapt to unexpected situations. In the future we hope to develop a team with many different strategy options, and the ability to choose the best strategy based on the current opponent.

5 Conclusions

The proposed increase in the field size will greatly reward teams that implement effective team play. We strongly feel that the dribbling mechanism will greatly improve the quality of the game, and allow teams to effectively use sophisticated team-based strategies. The main benefit of the omni-directional drive mechanism is a simplification of the resulting control problem, which *greatly* reduces the computation required for generating nearly optimal trajectories, and thus free up computational resources for higher level control decisions; we do not feel, however, that it will be a necessary feature of future competitive teams. It is clear, however, that successful future teams must seriously address dynamics and control issues, such as estimation, coping with system latency, robustness, and optimal control; only by doing so can the full benefits of team play and cooperation be achieved.

Play Transitions - Midfield

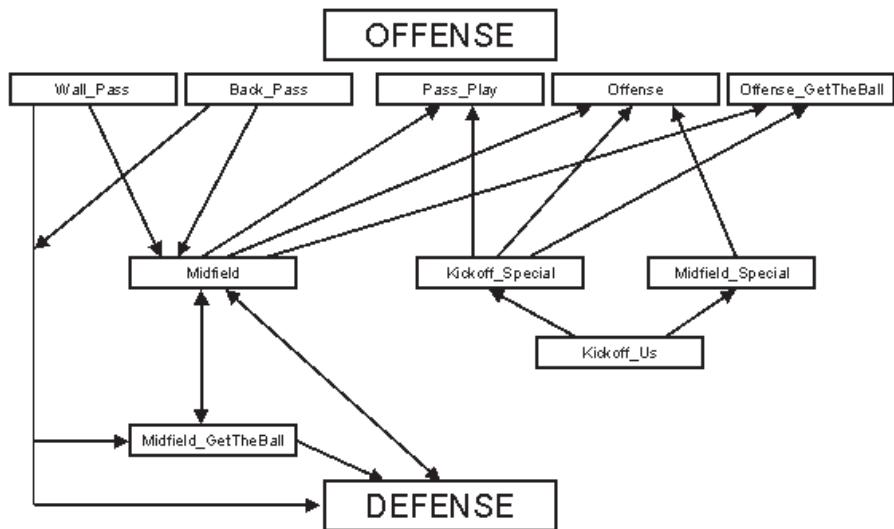


Fig. 7. Example of Playbook.

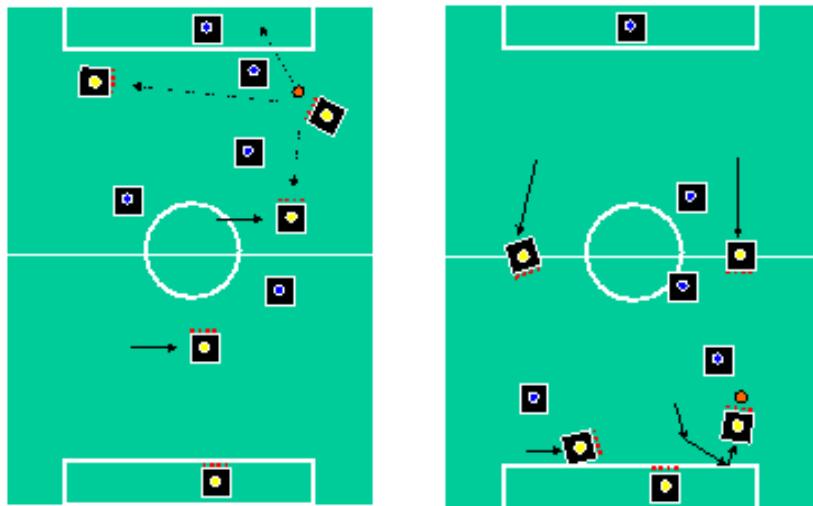


Fig. 8. Example of Plays. Left diagram, offensive play; right diagram, defensive play.

6 Acknowledgements

The lead author would like to thank all of the students involved in the RoboCup project at Cornell University, past and present, for their dedication to the project, and for their numerous accomplishments. We would also like to thank our many sponsors for their continuing support, in particular Jim Morgan at Applied Materials, John Swanson at ANSYS, Lockheed Martin, and United Technologies.

References

CS Freiburg: Doing the Right Thing in a Group^{*}

Thilo Weigel¹, Willi Auerbach³, Markus Dietl¹, Burkhard Dümler¹, Jens-Steffen Gutmann², Kornel Marko¹, Klaus Müller¹, Bernhard Nebel¹, Boris Szerbakowski³ and Maximilian Thiel¹

¹Institut für Informatik
Universität Freiburg

79110 Freiburg, Germany

last-name@informatik.uni-freiburg.de

²Corporate Technology
Siemens AG

81730 Munich, Germany

gutmann@ieee.org

³Optik Elektronik
SICK AG

79183 Waldkirch, Germany

first-name.last-name@sick.de

Abstract. The success of CS Freiburg at RoboCup 2000 can be attributed to an effective cooperation between players based on sophisticated soccer skills and a robust and accurate self-localization method. In this paper, we present our multi-agent coordination approach for both, action and perception, and our rich set of basic skills which allow to respond to a large range of situations in an appropriate way. Furthermore our action selection method based on an extension to behavior networks is described. Results including statistics from CS Freiburg final games at RoboCup 2000 are presented.

1 Introduction

After winning RoboCup in 1998 and coming in third in 1999, CS Freiburg won the competition in the F2000 league at RoboCup 2000 again. One of the reasons for this success is most probably the accurate and reliable self-localization method based on laser range finders [10]. However, while this was basically enough to win the competition in 1998, it was necessary to work on a number of different problem areas in order to stay competitive.

Our main research aim this year was to develop and test new techniques in the areas of *action selection* and *multi-robot cooperation*. In order to do so we also had to redesign the *action repertoire* and to rethink what kind of problems could be solved by a group of robots in which way. In addition to these issues we further enhanced our *perception technology*. However, most of the work in this area was already done last year [13] and will be described only briefly.

Figure 1 depicts the software architecture of our players. The *perception* technology as described in Section 3 is the basis of our team. In the area of *cooperative sensor interpretation* we were able to come up with interesting and significant results. As described in Section 3.3, the group estimation of the ball position is done using a combination of *Markov localization* and *Kalman filtering*. For RoboCup 2000 we completely redesigned our *strategy* component. We employ a variation of the dynamic role assignment as used in the *ART* team [5] and a variation of the *SPAR* [14] technique for determining optimal positions on the field for each player. One of the objectives in the development this year was to enhance the *basic skills* of the soccer agents. In particular,

* This work has been partially supported by *Deutsche Forschungsgemeinschaft* (DFG), by *Medien- und Filmgesellschaft Baden-Württemberg mbH* (MFG), and by *SICK AG*, who donated a set of laser range finders and supported us in constructing a new kicker.

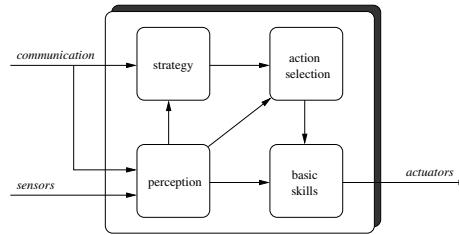


Fig. 1. Player architecture

we added a *dribbling* skill and a *ball-shooting* skill. For both of these new skills, it was necessary to modify the hardware and to account for these modifications by incorporating new software. In order to choose the right action in each situation, some *action selection mechanism* is necessary. We designed a new action selection mechanism based on Dorer's [6] extended behavior networks, which are used in Dorer's simulation team [7], the runner-up in the 1999 competition.

All in all, it turned out that the combination of techniques we employed allowed us to stay competitive, to demonstrate that our robots can play an attractive (robotic) soccer game, and to win the competition a second time after 1998.

2 Hardware

The basic hardware setup of CS Freiburg has remained mainly unchanged since the team first participated at RoboCup in 1998 [9]. Figure 2(a) shows a picture of one of our Pioneer 1 robots enhanced by a custom-made kicking device, the Cognachrome vision system for ball recognition, a SICK laser range finder for self-localization and object recognition, a Toshiba Libretto Notebook for local information processing and the WaveLan radio ethernet for communication.

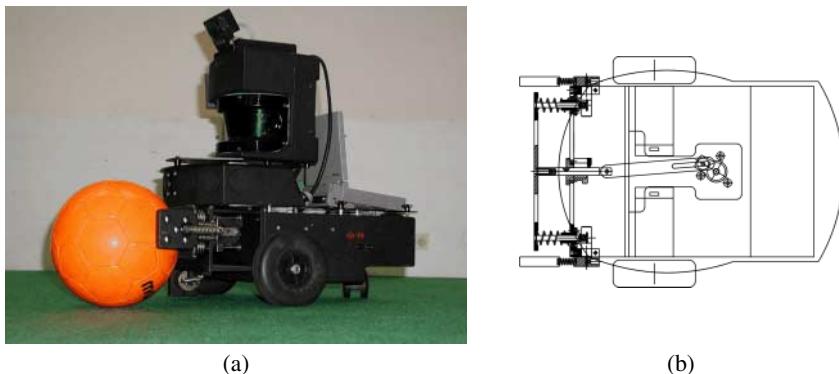


Fig. 2. A CS Freiburg player (a) and a sketch of the new kicking device (b).

In 1999 we already put effort in improving our perception technology [13]. We replaced the old PLS200 laser range finders with the new LMS200 models which provide depth information for a 180° field of view with an angular resolution of 0.5° and an

accuracy of 1 cm. With a serial connection of 500 KBaud we now get up to 25 scans per second. We also modified and extended the Cognachrome software to reduce the amount of vision data by discarding all informations not needed for our ball recognition module. Doing so we were able to raise the frame rate to 60 frames per second. The overall performance of our perception module was further improved by exploiting the feature of RTLinux to assign millisecond accurate time stamps to all sensor readings.

The major hardware improvement of this year is a new powerful kicker and ball steering mechanism. Figure 2(b) shows a sketch of the completely rebuild device. The kicker is based on a wind-screen wiper motor which compresses four springs attached to the ball steering plate when turning. The springs can be unlocked by a solenoid with the result of a very strong kick. The ball steering flippers can be turned to an upright position and back by two separately working DC motors. The movable flippers helped to avoid situations where robots would get stuck or accidentally hit the ball while turning towards it. In the future we intend to exploit the new feature for an elaborate ball interception behavior.

3 Perception

Back in 1998, the CS Freiburg team has been designed under the hypotheses that it is of great advantage if every robot would know its exact position and orientation on the soccer field. The general perception approach can roughly be divided into laser-based perception, vision-based perception and multi agent sensor integration.

3.1 Laser-Based Perception

As our research group already got a lot of experience with different self-localization methods using laser range finder data [8] it has been an obvious step to adapt a variant of them for our soccer robots. The method developed for the CS Freiburg team extracts line segments from a laser scan and matches them to an *a priori* line model of the soccer field. Only lines of a certain minimum extend are taken into account for discarding other players that are present in the field of the range finder. The scan-matched position is fused in a Kalman filter with the estimate from odometry.

Several experiments have been performed using this localization method and a comparison with other scan-matching methods shows that the line-based method is faster and more robust than point-based scan-matching methods while still retaining the precision of the other methods [10].

After matching a scan to the *a priori* model of the soccer field, players are extracted from the scan by removing all points belonging to the field walls and clustering the remaining ones. For each cluster the center of gravity is computed and considered as the center of a player. Inherent to this approach is the systematical error due to the different shapes of the robots [17]. At least for some players this error can be reduced, e.g. in our system we assume that the opponent goalkeeper is usually oriented parallel to the goal line, thus adding a constant offset to the center of gravity can reduce the position error for the opponent goalie¹.

¹ Of course this offset depends on the shape of the opponent's goalie and has to be adjusted before the game.

3.2 Vision-Based Perception

For recognizing and tracking the ball we make use of the Cognachrome vision system which delivers pixel-area information (so called *blobs*) of previously trained colors. We extended the supplied software to overcome problems with noisy readings, to enhance color training, and to customize the computed blob information to our needs.

A filter does plausibility tests to discard blobs whose shape, size or position make it very unlikely to correspond to the ball. From the remaining blobs the one closest to the previously selected blob is chosen and – by fitting a circle to it – various properties such as center, radius, or size in pixels are determined. Fitting a circle to the blob seemed to improve the overall position estimation of the ball as localization was still accurate when the ball was partially occluded by other robots.

From the computed blob center the global ball position is determined by using an off-line learned mapping table that contains distance and heading values for each pixel. This lookup table is autonomously learned by the robot before the game by positioning itself at various position on the field and taking measurements of the ball which stays at a fixed position. Interpolation is then used to compute the distance and heading pairs for pixels where no explicit information is available [15].

Despite of the applied filters we still observed infrequent wrong ball measurements due to reflections on shiny surfaces or badly trained colors, e.g. we had problems with the white field markings because when training the ball color, shiny reflections on the ball appeared to have a similar color. In order to detect such wrong ball measurements a global sensor integration module compares the observations of all players. This is described in the next section.

3.3 Multi Robot Sensor Integration

The perceptions of all players are sent to the *multi robot sensor integration module* for building and maintaining a global view of the world. Because the global model integrates more data than a single player is able to perceive, it should be more accurate and comprehensive than any local world model of a single player. After merging the player's perceptions into the global world model it is sent back to all players. Using the sensor informations of the whole team enables a player to take into account hidden objects when selecting an action. This proved to be especially advantageous for the global ball position, since the ball is seen almost all the time by at least one player. Furthermore knowing whether an observed object is an opponent or a teammate is, of course, very helpful for an effective cooperative team play.

In the first version of our *multi robot sensor integration module* we were using a simple averaging method for computing global positions of all objects on the field [9]. However, we encountered problems with very inaccurate or even completely wrong measurements, e.g. wrong ball observations as mentioned above, which corrupted a coherent global position. Therefore a more sophisticated method, namely Kalman filtering, for fusing observations and rejecting outliers is now employed. When tracking the ball, however, rejecting outliers by using a validation gate [3] has a big disadvantage: Once the system is tracking the ball based on a wrong ball observation, the filter needs several cycles until it accepts other observations again.

Therefore, we followed the proposal of Gutmann *et al.*[8] and combined Markov localization [4] which employs a multi-modal probability distribution for fusing ball observations with a uni-modal Kalman filter for precise ball localization. This way, the Kalman filter always tracks the ball using observations validated by one or more robots and only few cycles are needed in case the system was tracking a ball based on wrong observations.

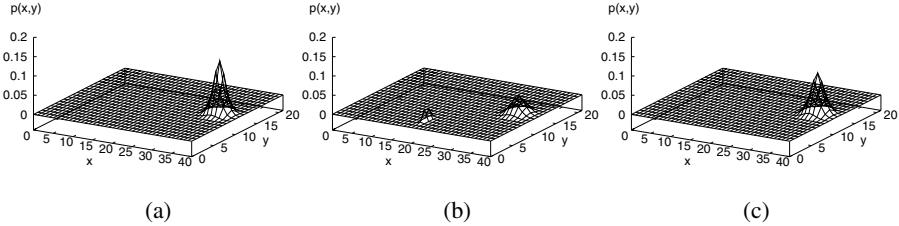


Fig. 3. Probability grids after (a) correct measurements, (b) one additional wrong measurement and (c) another correct one.

Figure 3 illustrates our Markov localization variant on the basis of real measurements. At first the ball estimate is very reliable. An erroneous measurement makes this estimate less likely, but still more probable than the new estimate. Another correct measurement strengthens the assumptions that the previous one was wrong.

4 Strategy

Coordination among our players and their positioning on the field followed a rather inflexible scheme at the last tournaments. In order to achieve a more flexible and more effective coordinated team play we redesigned our strategy component completely this year. As described in the following sections we introduced dynamic *roles* for the players and dynamic target positions for each role.

4.1 Role Assignment

The CS Freiburg players organize themselves in roles, namely *active*, *support* and *strategic*. While the active player always tries to get and play the ball, the supporting player attempts to assist by positioning itself appropriately. The strategic player always occupies a good defensive position.

Each player constantly calculates its utility to pursue a certain role and communicates the result to its teammates. Based on its own and the received utilities a player decides which role it wants to take. This approach is similar to the one taken by the ART team [5],² however, the CS Freiburg players additionally communicate to their teammates which role they are currently pursuing and which role they desire to take. A role can only be taken from another player if the own utility for this role is the best of all players and the robot currently pursuing the role also wants to change its role. Following this strategy makes it less likely that two or more players are pursuing the same role at the same time than assigning roles based on utility values only.

Figure 4 shows a screenshot of the global view during a game. While the active player dribbles the ball around an opponent the supporting player moves to its target position and the strategic player observes the ball. Roles for the field players are assigned in the order *active*, *strategic*, *support*. Since the goalkeeper has a special hardware setup, its role is fixed by assigning appropriate utility values to all roles.

² Note, however, that our approach was developed independently.

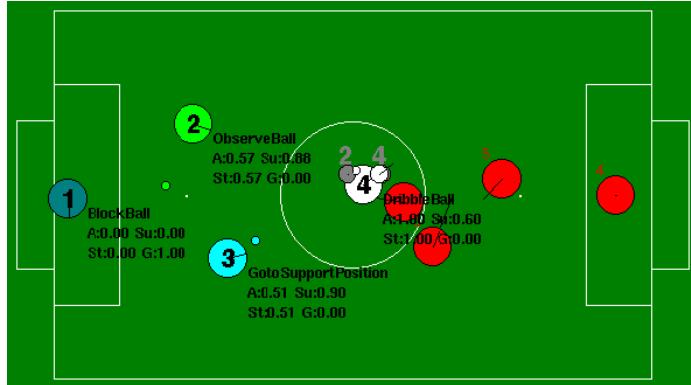


Fig. 4. Visualization of the results of the global sensor integration together with a players utilities for taking a certain role, its current role and its current action. The small white circle denotes the position of the ball as determined by the global sensor integration, whereas the small grey ones correspond to the ball as perceived by the individual players.

4.2 Positioning

The target positions of the players are determined similar to the SPAR method of the *CMU* team in the small size league [14]. From the current situation observed by the robots a potential field is constructed which includes repulsive forces arising from opponent players and attracting ones from desirable positions, e.g. positions from where the ball is visible. Positions are then selected based on the robot's current role, e.g. the position of the active player is set close to the ball, the supporting player is placed to the side and behind the active one, and the strategic player takes a defensive position which is about half way between the own goal and the ball but behind all opponent players.

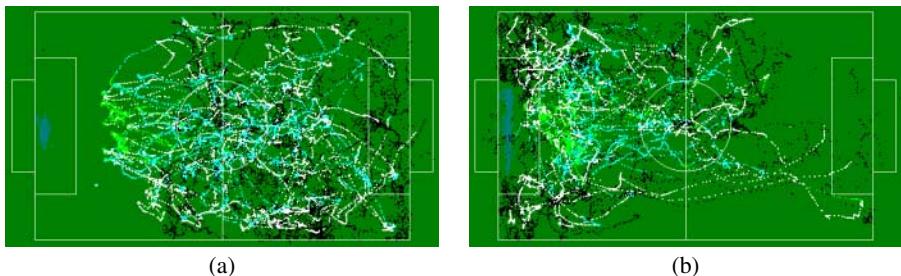


Fig. 5. Trace of the positions of the ball (black) and the CS Freiburgs' players during (a) the quarter final and (b) the final.

Figure 5 shows a trace of the positions of our players and the ball during the quarter final against *CMU* and the final against *Golem*. While in the quarter final our team spent most of the time in the opponents half our players were forced into our own half most of the time during the final game. The high correlation between the position of the ball and our players however demonstrates the effective positioning of our team.

5 Tactics: Basic Skills and Action Selection

In order to play an effective and successful game of robotic soccer a powerful set of basic skills is needed. This section describes the basic skills implemented for the goalkeeper and the field players in CS Freiburg, as well as the method of action selection developed for the field players.

5.1 Goalkeeper

As in most other robot soccer teams in the middle size league the hardware configuration of our goalkeeper differs from the one of the field players which is mainly because of the different tasks the goalkeeper and field players are designed for. Our goalie has a special hardware setup where the *head* of the robot, containing the sonar array, laser range finder and vision camera, is mounted 90° to one side allowing the robot to move quickly parallel to the goal line (see Figure 6). This kind of setup is quite popular in the middle size league and used by other teams, too, e.g. the *Agilo* team [2].

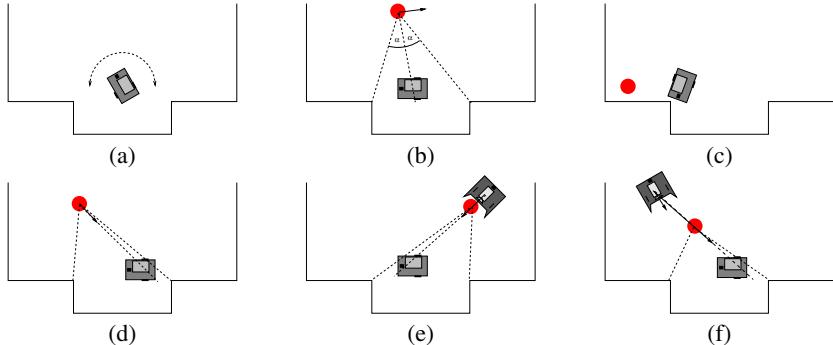


Fig. 6. Goalkeeper's tactics for saving CS Freiburg's goal: (a) ball searching, (b) minimizing area of attack, (c) turning to corner, (d) intercepting ball, (e) *intelligent* interception using opponent heading, and (f) *intelligent* interception using opponent to ball heading.

Our goalkeeper uses six skills sketched in Figure 6 for keeping the ball from rolling over our goal line. If the robot doesn't know where the ball is, it rotates left and right searching for it (*SearchBall*). If the ball does not directly roll towards our goal, the area of attack is minimized by moving to a position from where both sides to the robot allow approximately the same amount of angular space for the ball passing between goalie and goal posts (*BlockBall*). On the left and right side of the goal, the robot turns towards the corners giving less chances for an opponent to score a direct goal. The last three skills concern cases of direct goal danger where the ball or an opponent is moving directly towards our goal (*InterceptBall*). Here the robot moves to an interception point based on the heading of the ball (Figure 6(d)), the heading of an opponent owning the ball (Figure 6(e)), or the heading of an opponent to the ball (Figure 6(f)). The last two and more sophisticated tactics are based on the assumption that the attacking robot will kick the ball in a straight way which is true for most robot teams participating in the middle size league but is not true for teams like *Golem* or *Alpha++*. Therefore these two tactics can be turned on or off before a game starts.

5.2 Basic Skills for Field Players

To get hold of the ball a player moves to a position behind the ball following a collision-free trajectory generated by a path planning system which constantly (re)plans paths based on the player's perception of the world (*GoToBall*). The system is based on potential fields and uses A^* search for finding its way out of local minima. If close to the ball a player approaches the ball in a reactive manner to get it precisely between the fingers while still avoiding obstacles (*GetBall*). Once in ball possession, a player turns and moves the ball carefully until facing in a direction which allows for an attack (*TurnBall*). If the player is right in front of the opponents goal it kicks the ball in a direction where no obstacles block the direct way to the goal (*ShootGoal*). Otherwise it first heads towards a clear area in the goal and turns sharply just before kicking in case the opponent goalkeeper moved in its way (*MoveShootFeint*). However if obstacles are in the way to the goal, the player tries to dribble around them (*DribbleBall*) unless there is not enough room. In this case the ball is kicked to a position close to the opponents goal by also considering rebound shots using the walls (*ShootToPos*). In the event of being too close to an opponent or to the field border the ball is propelled away by turning quickly in an appropriate direction (*TurnAwayBall*). If a player gets stuck close to an obstacle it tries to free itself by first moving away slowly and (if this doesn't help) then trying random moves (*FreeFromStall*). However a player doesn't give way if the ball is stuck between himself and an opponent to avoid being pushed with the ball towards his own goal (*WaitAndBlock*).

Against fast playing teams our robots were often outperformed in the race for the ball when approaching it carefully. We therefore developed two variants of a skill for situations in which speed is crucial. Both let the robot rush to the ball and hit it forwards while still avoiding obstacles. In offensive play *BumpShootOffense* is employed to hit the ball into the opponents goal when very close to it. In defensive play the use of *BumpShootDefense* can be switched on or off according to the strength of the opponent.

Players fulfilling strategic tasks position themselves following collision-free paths to dynamically determined positions (*GoToPos*). From these positions the players either search the ball if not visible (*SearchBall*) by rotating constantly or observe it by turning until facing it (*ObserveBall*). In offensive play a strategic player may also take a position from where it should be able to score a goal directly (*WaitForPass*). Once in such a position he signals to his teammates that he is waiting to get the ball passed. The decision is then up to the ball owning player whether to pass the ball (*PassBall*) or try to score a goal by itself. Especially against fast playing teams we rarely got the chance to try out ball passing between players. Equipped with the strong kicking device shooting at the opponents goal always appeared more promising than passing the ball to another teammate.

To comply with the "10-seconds rule" a player keeps track of the time he is spending in a penalty area. Whenever he spent more than the allowed time he leaves the area following a collision-free path generated by the same path planning system as employed in the *GoToBall* skill (*LeavePenaltyArea*).

At RoboCup 2000, our team seemed to be one of the few teams capable of effectively dribbling with the ball and the only one which exploited deliberately the possibility of rebound shots using the walls. Therefore these skills will be described in more detail.

Figure 7(a) shows a screenshot of a player's local view while dribbling. In every cycle, potential continuations of the current play are considered. Such continuations

are lines to points closer to the opponents goal within a certain angle range around the robot's heading. All the possible lines are evaluated and the direction of the best line sample is taken as the new desired heading of the robot. A line is evaluated by assigning it a value which is the higher the further it is away from objects, the less turning is necessary for the player and the closer its heading is to the opponents goal center. Determining the robots heading this way and adjusting the wheel velocities appropriately in every cycle lets the robot smoothly and safely dribble around obstacles without loosing the ball. The CS Freiburg team scored some beautiful goals in this year's tournament after a player had dribbled the ball over the field around opponents along an S-like trajectory. Of course, all this only works because the ball steering mechanism allows for a tight ball control.

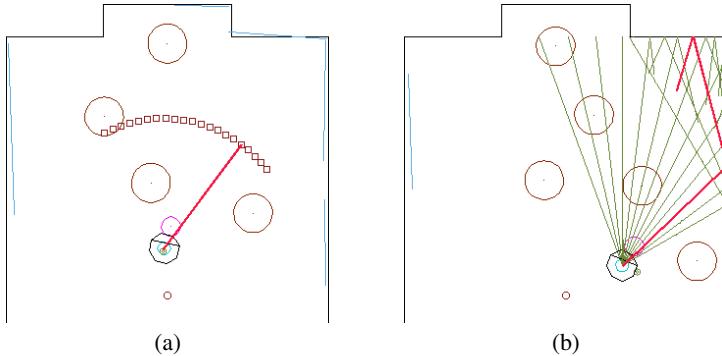


Fig. 7. A CS Freiburg player's view of the world while (a) dribbling and (b) ball-shooting. Circles denote other robots and the small circle in front of the player corresponds to the ball. Lines almost parallel to the field borders are perceived by the laser range finder. The other lines leading away from the player are evaluated by the skills.

Figure 7(b) shows a screenshot of a player during ball-shooting. For this skill the lines are reflected at the walls and are evaluated to find the best direction where to kick the ball. A lines value is the higher the further away from obstacles it is, the closer its endpoint is to the opponents goal and the less turning is required for the player to face in the same direction. Taking into account that the ball doesn't rebounce at the field borders in a perfect billiard-like manner we calibrated manually the correlation between the angles of reflection. Using the passing skill our players were able to play the ball effectively to favorable positions and even to score goals directly.

Figure 8 shows statistics of how long a skill was active for a certain role. Since we constantly improved and modified the action selection mechanism during the preliminary games the statistics are based on the three final games only. We found it quite surprising that the players were searching for the ball up to 10 % of the playing time. However analysis of the log-files showed that the high numbers can be either attributed to communication problems or to one of few situations where no player was seeing the ball. In the final games the ball was seen by the team in 95% of the playing time.

The statistics show that the goalkeeper was most of the time minimizing the area of attack but not under direct threat since the InterceptBall skill was only active in 11 % of the time. The supporting and strategic player spent most of their time observing the ball. This was intended since moving more often or more accurately to their target positions

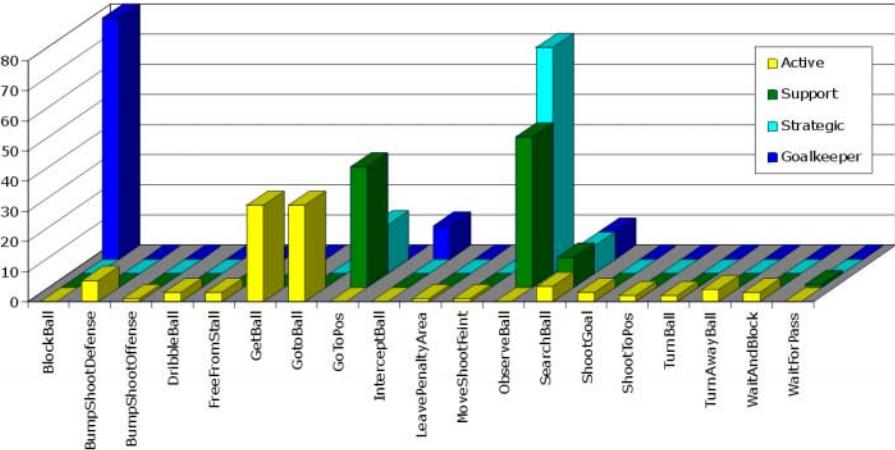


Fig. 8. Time in percent a skill was active in the final games broken down for the different roles.

results in a very nervous behavior as we could observe in one of our preliminary games. It turned out that for our rather clumsy robots staying at one spot was more effective than moving and turning constantly. Because the strategic player usually occupied a good defending position we allowed it to move even less.

At first view it seems surprising that the active player was most of the time (64 %) occupied with getting hold of the ball. However the fact that after kicking the ball the active player usually starts to follow the ball again explains the high numbers for GoTo and GetBall. Nevertheless it made use of all the skills available to it. This demonstrates that the active player in fact distinguished between a large number of different game situations.

5.3 Action Selection for Field Players

Our action selection module is based on extended behavior networks proposed by Dorer [6]. They are a revised and extended version of the behavior networks introduced by Maes [11] and can be viewed as a particular form of decision-theoretic planning. The main structural elements in extended behavior networks are *competence modules* which consist of a certain behavior to be executed, preconditions and positive or negative effects. *Goals* can explicitly be specified and can have a situation-dependent relevance, reflecting the agent's current motivations. The state of the environment, as it is perceived by the agent, is described via a number of continuous propositions $p_i \in [0..1]$. Competence modules are connected with goals if they are able to influence goal conditions and also with each other, if a competence module has an effect that is a precondition of another. Along the resulting network connections an *activation spreading mechanism* is defined, with goals being the source of activation. An action is selected by considering each competence module's executability and received activation.

Figure 9 shows a part of the extended behavior network that was used in this year's competition [12]. The ellipses represent the competence modules with their preconditions below and their effects on top of them. Currently our players have two goals: Shoot a *soccergoal* or *cooperate* with teammates. The relevance condition *role_active*

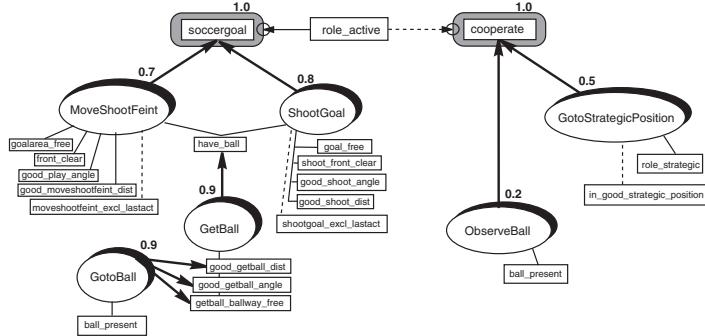


Fig. 9. A part of CS Freiburg's extended behavior network.

(player has role *active*) ensures that only one of these goals is relevant at a time depending on the player's current role. The strength of the effect connections (indicated by the numbers next to arrows) are set manually so far and reflect priorities among the competence modules.

6 Conclusion

The games played by CS Freiburg at RoboCup 2000 looked much more dynamic than the games in 1999 and 1998. So, the cooperation mechanism and the action selection (with the new action repertoire) seem to have improved our play considerably.

However, also the other teams have improved their play. In general, it seems to be the case that the performance of all F2000 teams have increased over the years and that there are more very good teams now – provided we measure the performance by goals/minute. As can be seen in Fig. 10, the average goal rate has almost doubled from 1997–2000,³ and the goal rate of the two best teams became better over the years.

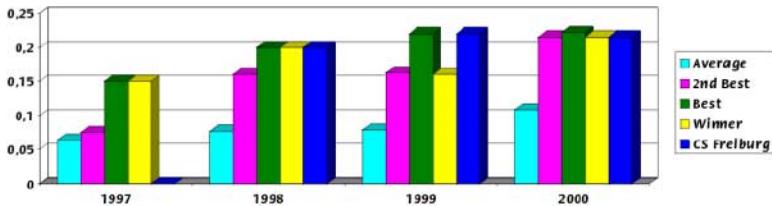


Fig. 10. Goal rates at RoboCup tournaments in goals per minute.

Furthermore, at RoboCup 2000 there was a large number of teams with a performance very close to the performance of CS Freiburg. In fact, while in previous years, CS Freiburg was the only team with an average goal rate of more than 0.2 goals/minute, this year there were 5 teams with a similar goal rate. Furthermore, it was obvious to everybody that the mechanical design of *CE Sharif* and *Golem* was definitely superior to

³ Note, however, that the conclusion that the goal rate in 2000 was higher than the goal rate in 1997 is statistically significant on the 90% level only.

ours. That we nevertheless were able to stay ahead can at least partially attributed to the careful design of our actions, the action selection mechanism as well as the cooperation mechanisms. It should also be noted, however, that some game results were very close. In particular, the final and the third/fourth place games were decided by penalty kicks.

In the future, we intend to enhance our robot base to make it faster, to use learning techniques for tuning skill parameters, to refine the method of action selection, and to enhance team play.

References

1. M. Asada and H. Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. Lecture Notes in Artificial Intelligence. Springer-Verlag, 1999.
2. Thorsten Bandlow, Robert Hanek, Michael Klupsch, and Thorsten Schmitt. Agilo RoboCuppers: RoboCup Team Description. In Veloso et al. [16], pages 691–694.
3. Robert G. Brown and Patrick Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley & Sons, 1997.
4. Wolfram Burgard, Dieter Fox, Daniel Hennig, and Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *AAAI'96*, 1996.
5. Claudio Castelpietra, Luca Iocchi, Maurizio Piaggio, Alessandro Scalzo, and Antonio Sgorbissa. Communication and coordination among heterogeneous mid-size players: ART99. In *Proceedings of the 4th International Workshop on RoboCup*, pages 149–158, Melbourne, Australia, August 2000.
6. Klaus Dorer. Behavior networks for continuous domains using situation-dependent motivations. In *IJCAI'99*, pages 1233–1238, 1999.
7. Klaus Dorer. The magmaFreiburg Soccer Team. In Veloso et al. [16], pages 600–603.
8. Jens-Steffen Gutmann, Wolfram Burgard, Dieter Fox, and Kurt Konolige. An experimental comparison of localization methods. In *IROS'98*, Victoria, October 1998.
9. Jens-Steffen Gutmann, Wolfgang Hatzack, Immanuel Herrmann, Bernhard Nebel, Frank Ritter, Augustinus Topor, Thilo Weigel, and Bruno Welsch. The CS Freiburg robotic soccer team: Reliable self-localization, multirobot sensor integration, and basic soccer skills. In Asada and Kitano [1], pages 93–108.
10. Jens-Steffen Gutmann, Thilo Weigel, and Bernhard Nebel. Fast, accurate, and robust self-localization in polygonal environments. In *IROS'99*, October 1999.
11. Pattie Maes. Situated agents can have goals. In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT Press, Cambridge, MA, 1990.
12. Klaus Müller. Roboterfußball: Multiagentensystem CS Freiburg (in German). Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 2000.
13. Bernhard Nebel, Jens-Steffen Gutmann, and Wolfgang Hatzack. The CS Freiburg '99 team. In Veloso et al. [16], pages 703–706.
14. P. Stone, M. Veloso, and P. Riley. CMUnited-98: Robocup-98 simulator world champion team. In Asada and Kitano [1].
15. Maximilian Thiel. Zuverlässige Ballerkennung und Positionsschätzung (in German). Diplomarbeit, Albert-Ludwigs-Universität Universität Freiburg, Institut für Informatik, 1999.
16. M. Veloso, E. Pagello, and H. Kitano, editors. *RoboCup-99: Robot Soccer World Cup III*. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000.
17. Thilo Weigel. Roboter-Fußball: Selbstlokalisierung, Weltmodellierung, Pfadplanung und verhaltensbasierte Kontrolle (in German). Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

The UNSW RoboCup 2000 Sony Legged League Team

Bernhard Hengst, Darren Ibbotson, Son Bao Pham, John Dalgliesh, Mike Lawther,
Phil Preston, Claude Sammut

School of Computer Science and Engineering
University of New South Wales, UNSW Sydney 2052 AUSTRALIA
{bernhardh,s2207509,sonp,johnd,mikel,philp,claude}@cse.unsw.edu.au

Abstract. We describe our technical approach in competing at the RoboCup 2000 Sony legged robot league. The UNSW team won both the challenge competition and all their soccer matches, emerging the outright winners for this league against eleven other international teams. The main advantage that the UNSW team had was speed. The robots not only moved quickly, due to a novel locomotion method, but they also were able to localise and decide on an appropriate action quickly and reliably. This report describes the individual software sub-systems and software architecture employed by the team.

1 Introduction

Each team in the Sony legged robot league consists of three robots that play on a pitch about the size of a ping pong table. All teams use the same Sony quadruped robots. The 2000 competition included entries from twelve international laboratories. Since all teams use the same hardware, the difference between them lies in the methods they devise to program the robots. The UNSW team won the championship as a result of their innovative methods for vision, localisation and locomotion. A particular feature of these methods is that they are fast, allowing the robots to react quickly in an environment that is adversarial and very dynamic.

The architecture of the UNSW United software system consists of three modules that provide *vision*, *localisation* and *action* routines. A *strategy* module coordinates these capabilities. Currently two strategy modules implement the roles of forward and goalkeeper. Each role can invoke a set of behaviours to achieve its goal. In the following sections, we describe the infrastructure modules that perform the vision processing, object recognition, localisation, and actions. We then describe the basic behaviours and strategies.

2 Vision

Since all the objects on the field are colour coded, the aim of the first stage of the vision system is to classify each pixel into the eight colours on the field. The colour classes of interests are orange for the ball, blue and yellow for the goals and beacons,

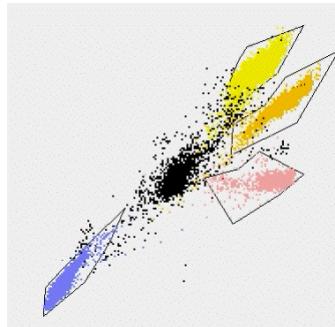


Fig. 1 A polygon growing program finds regions of pixels with the same colour.

pink and green for the beacons, light green for the field carpet, dark red and blue for the robot uniforms.

Currently, we only use the medium resolution images (88×60 pixels) available from the camera. The information in each pixel is in YUV format, where each of Y, U and V is in the range 0 to 255. The U and V components determine the colour, while the Y component represents the brightness. The Sony robots have an onboard hardware colour look up table. However, for reasons that will be explained later, we have chosen to perform the colour detection entirely in software.

Our vision system consists of two modules: an offline training module and onboard colour look up module. The offline software generates the colour tables and stores them in a file. At boot time, the onboard software reads the colour table from the file and then performs a simple table lookup to classify each pixel in the input image. We next explain how the colour table is generated.

Because colour detection can be seriously affected by lighting conditions, we need a vision system that can be easily recalibrated. The first step is to take about 25 snapshots of different objects at different locations on the field. Then for each image, every pixel is manually classified by “colouring in” the image by hand, using a purpose designed painting program. The labelled pixels form the training data for a learning algorithm.

In the 1999 team’s software, all pixels were projected onto one plane by simply ignoring the Y value. For each colour, a polygon that best fits the training data for that colour was automatically constructed. An unseen pixel could then be classified by looking at its UV values to determine which polygons it lies in. As the polygons can overlap, one pixel could be classified as more than one colour.

Figure 1 shows a screen grab at the end of a run of the polygon growing algorithm. It also illustrates why we chose to use polygonal regions rather than the rectangles used by the hardware colour lookup system. We believe that polygonal regions give greater colour classification accuracy.

For the 2000 competition, we kept the polygon growing algorithm but now also use the Y values. Initially, Y values were divided into eight equally sized intervals. All pixels with Y values in the same interval belong to the same plane. For each plane, we run the algorithm described above to find polygons for all colours.

Once the polygons have been found, they must be loaded onboard the robots to allow them to perform the colour lookup. Because we cannot use the Sony hardware, the colour information must be stored in such a way as to allow fast operation in software. We chose a set of two-dimensional arrays, where each $\langle U, V \rangle$ pair specifies one element in an array. The value of the element is determined by the polygons in which the $\langle U, V \rangle$ values lie. To determine the colour of an unseen pixel, the Y value is first examined to find the relevant plane, then $\langle U, V \rangle$ index into the array and the value of that element gives the colour.

Discretisation of the Y values into eight equal intervals leads to better colour discrimination, but the robots were still unable recognise the red and blue colours of other robots. To the onboard camera, those colours appear very dark and were being mapped to the same plane as black and other dark colours.

Being able to classify these colours is vital for robot recognition and consequently, team play, so a further refinement was attempted. A manual discretisation of Y values was attempted, settling on 14 planes of *unequal* size. More planes are assigned to lower Y values, reflecting the fact that dark colours are hard to separate. With 14 planes, the robots can recognize the colour of the robot uniforms with reasonable accuracy, but further work is required to obtain greater reliability.

The 1999 version of the polygon growing algorithm allowed polygons to overlap. Pixels could be classified as more than one colour. This caused two problems. One is the obvious ambiguity in classification; the other is inefficiency in storage. By ignoring pixels that occur in overlapping polygons, we removed the overlap. Object Recognition

Once colour classification is completed, the object recognition module takes over to identify the objects in the image. Four-connected colour blobs are formed first. Based on these blobs, we then identify the objects, along with and their distance, heading and elevation relative to the camera and the neck of the robot.

2.1 Blob formation

The robot's software has a decision making cycle in which an image is grabbed, and object recognition and localisation must be performed before an appropriate action is chosen and then executed. Thus, every time the robot receives an image, it must be processed and action commands sent to the motors before the next image can be grabbed. The faster we can make this cycle, the quicker the robot can react to changes in the world. Blob formation is the most time-consuming operation in the decision making cycle. Therefore a fast, iterative algorithm [3] was developed that allows us to achieve a frame rate of about 26 frames/second most of the time.,

2.2 Object Identification

Objects are identified in the order: beacons, goals, ball and finally the robots. Since the colour uniquely determines the identity of an object, once we have found the bounding box around each colour blob, we have enough information to identify the object and compute various parameters. Because we know the actual size of the object

and the bounding box determines the apparent size, we can calculate the distance from the snout of the robot (where the camera is mounted) to the object. We then calculate heading and elevation relative to the nose of the robot and the blob's centroid.

Up to this point, distances, headings, etc, are relative to the robot's snout. However to create a world model, which will be needed for strategy and planning, measurements must be relative to a fixed point. The neck of the robot is chosen for this purpose. Distance, elevations and headings relative to the camera are converted into neck relative information by a 3D transformation using the tilt, pan, and roll of the head [2].

Every beacon is a combination of a pink blob directly above or below a green, blue or yellow blob. The side of the field the robot is facing is determined by whether the pink blob is above or below the other blob. The beacons are detected by examining each pink blob and looking for the closest blob of blue, yellow or green to form one beacon. Occasionally, this simple strategy fails. For example, when the robot can just see the lower pink part of a beacon and the blue goal, it may combine these two blobs and call it a beacon. A simple check to overcome this problem is to ensure that the bounding boxes of the two blobs are of similar size and the two centroids are not too far apart. The relative sizes of the bounding boxes and their distance determine the confidence in identifying a particular beacon.

After the beacons have been found, the remaining blue and yellow blobs are candidates for the goals. The biggest blob is chosen as a goal of the corresponding colour. Since the width of the goal is roughly twice as long as the height of the goal, the relative size between height and width of the bounding box determines confidence in the identification of that goal. There are also some sanity checks such as the robot should not be able to see both goals at the same time and the goal cannot be to the left of left-side beacons nor to the right of right-side beacons. Sometimes, the robot would try to kick the ball into a corner because it could only see the lower blue part of the beacon in the corner and would identify that as the goal. To avoid this misidentification, we require the goal to be above the green of the field.

The ball is found by looking at each orange blob in decreasing order of bounding box size. To avoid misclassifications due to orange objects in the background, the elevation of the ball relative to the robot's neck must be less than 20°. The elevation must also be lower than that of all detected beacons and goals in the same image.

When the camera is moving, pixels are blurred, resulting in the combination of colours. Since red and yellow combine to form orange, red robots in front of a yellow goal can be misclassified as the ball. A few heuristics were used to minimise the problems caused by this effect. If there are more red pixels than orange pixels in the orange bounding box then it is not the ball. When the ball is found to be near the yellow goal, it must be above the green field. That is, if the orange blob must be above some green pixels to be classified as the ball. These heuristics allowed our robots to avoid most of the problems encountered by other teams. However, more work is required to completely overcome this problem.

2.3 Robot Recognition

The robot recognition algorithm used at RoboCup 2000 uses a combination of visual and infrared sensors to identify the presence of, at most, one robot in the visual field and to approximate the distance from the camera to the object. For the purposes of obstacle avoidance, important frames generally don't contain multiple robots.

The on-board infrared sensor provides accurate distance information for any obstacle aligned directly in front of the head of the robot at a distance between 10-80cm. Below 10cm, the IR will read somewhere between 10-20cm. The main noise factor for the IR sensor is the ground. A work-around for this is that the IR reading is passed on as full range (1501mm) when the IR sensor is pointing downward more than 15°.

The initial design of the robot recognition algorithm was based upon a sample of 25 images of robots taken from the robot's camera, as well as manually measured distances to each of the robots in the samples. A further set of 50 images was taken when new uniforms were specified by Sony. The colour detection and blob formation algorithms were run over the sample images and blob information obtained. Blobs with fewer than ten pixels were discarded as noise and the following two values calculated: total pixels in each blob and the average number of pixels per blob. From this information, a curve was manually fitted to the sample data and a distance approximation was derived based purely on the feedback from the camera.

While the vision system is used to detect the presence of a robot and estimate its distance at long range, the infrared sensor is used at short range. Once the distance has been approximated, several sanity checks are employed. These filter out spurious long-range robot detections (> 60 cm) and robots that are probably only partially on camera, that is, the number of patches of the uniform is unlikely in comparison to distance.

Although robot recognition was not very reliable, using the infrared sensors at short ranges allowed the algorithm to identify situations where there is a risk of collision. The primary weakness of robot recognition is its reliance on accurate colour classification. The algorithm does not adapt well to background noise that often causes it to misclassify a robot or produce a grossly inaccurate distance approximation. This main weakness is almost exclusive to blue robot detection, with the red uniforms being far easier to classify accurately.

3 Localisation

The Object Recognition module passes to the Localisation module the set of objects in the current camera image, along with their distances, headings and elevations. Localisation tries to determine where the robot and other objects are on the field. It does so by combining its current world model with the new information received from the Object Recognition module. Since all beacons and goals are static, we only need to store the positions of the robot and the ball. We do not attempt to model the other robots.

The world model maintains three parameters for the robot: its x and y coordinates and its heading. The left-hand corner of the team's own goal is the origin, with the x-axis going through the goal mouth. The robots first attempt to localise using only the objects detected in the current image. Being stationary, beacons and goals serve as the landmarks to calculate a robot's position. Because of the camera's narrow field of view, it is almost impossible to see three landmarks at once, so any algorithm that requires more than two landmarks is not relevant. If two landmarks are visible, the robot's position is estimated using the triangulation algorithm used by the 1999 team [2]. This technique requires the coordinates, distance and heading of two objects relative to the robot.

More information can be gathered by combining information from several images. Thus, the localisation algorithm can be improved by noting that if the robot can see two different landmarks in two consecutive images while the robot is stationary, then triangulation can be applied. Typically, this situation occurs when the robot stops to look around to find the ball. To implement this, we use an array to store landmark information. If there is more than one landmark in the array at any time, triangulation is used. This array is cleared every time the robot moves.

The world model receives feedback from the locomotion module, *PWalk*, to adjust the robot's position. The feedback is in the form the distances, in centimetres, that the robot is estimated to have moved in the x and y directions and the number of degrees through which the robot is estimated to have turned. This feedback is received about every 1/26 second when the robot is moving. Odometry information is clearly not very accurate and small errors in each step accumulate to eventually give very large inaccuracies. Also, if the robot is blocked by an obstacle, *PWalk* is not aware of this and sends incorrect information to the world model.

Since the robot usually sees only one landmark in an image, we devised a method for updating the robot's position from a single landmark. This is explained in Figure 2, with details given in [3]. The main feature of the algorithm is that with a fast frame rate of about 26 frames/second, it converges on an accurate estimate of the robot's position quite quickly and accurately. Within a reasonably short period of time, the robot usually catches sight of several landmarks, thus approximating triangulation. One landmark update overcomes many of the problems caused by odometry error. Even when the robot's movement is blocked and the odometry information is incorrect, if the robot can see one landmark it will readjust its position. Because we use a low trot gait, the robot can see goals and beacons most of the time, if they are not obscured by other robots.

One problem remains due to the perception of landmarks. A goal is large and often the robot is only able to see a part of it or much of it may be obstructed by the goalie. Consequently, distance measurements may be inaccurate. Therefore, when the robot is in the middle of the field or near the edge, the goals are ignored, if the beacons are visible. Near a goal, the beacons are often difficult to see, so the heading of the goal is used to update the robot's heading. However, the robot's (x, y) position is not updated because the measurement of distance to the goal is too unreliable. Thus, the goal is never used in triangulation.

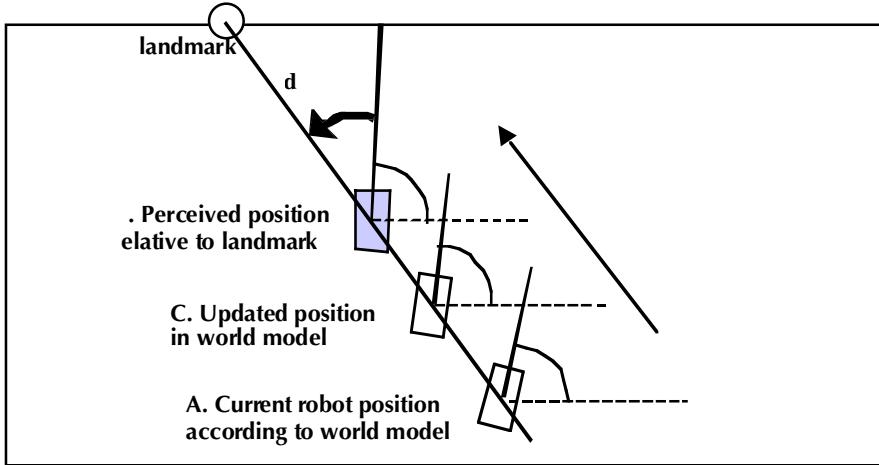


Fig. 2. To estimate the position of the robot, we draw a line between the landmark and the estimated current position (A) of the robot in the world model. The robot's perceived position (B), relative to the landmark, is the point on that line d cm away from the landmark. The localisation algorithm works by “nudging” the estimated position in the world model (C) towards the perceived position relative to the landmark.

4 Action/Execution

The purpose of the action module is to move the head and legs in response to commands from the behaviour module. Head and leg commands are given concurrently and are executed concurrently. The three primary design objectives of the action module were to:

1. Drive the robot as if controlled by a joystick with three degrees of freedom: forward, backward, sideways left or right and to turn on the spot clockwise or counterclockwise.
2. Move the robot over the ground at a constant speed, thus reducing the strain on the robot motors by not accelerating and decelerating the body.
3. Keep the camera as steady as possible. Using other walks, we observed that images from the robot's camera showed wildly erratic movements due to the robots head and leg motions.

The solution adopted was to move the paws of the robot's feet around a rectangular locus (Figure 3). The bottom edge of the rectangle describes that part of the path during which the paws make contact with the ground. The sides and top of the locus describe the path used to lift the paw back ready for it to take the next step. In the trot gait, used for the competition, diagonally opposite legs touch the ground alternately. If the paws that touch the ground move at a constant velocity, the robot should move at that same constant velocity. This requires that the time taken to move the paw

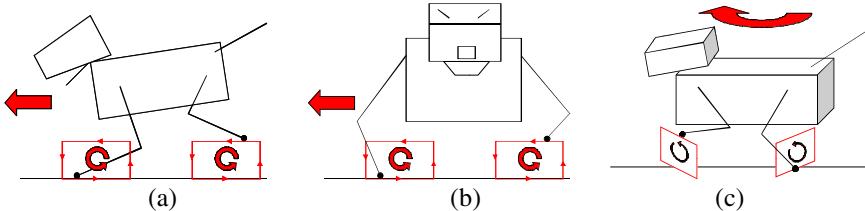


Fig. 3. Forward, sideways and turning achieved by adjusting angles of rectangular locus of leg movement

along the bottom edge of the rectangle is equivalent to the total time taken to move the paw along the other three edges.

Design objectives 2 and 3 were achieved in this way. The speed over the ground is constant as long as the size of the rectangular locus does not change and its traversal is at a constant frequency. The robot's camera is steadied because the bottom edge of the rectangular locus is a straight line lying in the ground plane. When the robot loses balance in the trot walk there is camera movement until it is arrested by falling on one of the legs that is off the ground. This movement can be minimised by lifting the legs as little as possible during the walk. Unfortunately in practice, it was necessary to specify a significant leg lift height to ensure that the spring-loaded claws would clear the carpet. This introduced some unwanted camera movement.

We now address design objective 1, that is, how to control which way the robot moves. The plane containing the rectangular locus for the paw is always perpendicular to the ground. By changing the angle of this plane relative to the sides of the robot, we determine whether the robot moves forward, backward or sideways. For example, if the locus plane is parallel to the sides, the robot will move forward or backward (Figure 3(a)). If we angle the locus plane perpendicular to the sides of the robot it will either move left or right (Figure 3(b)). Figure 3(c) shows how the robot can be made to turn by angling the locus planes tangentially at each shoulder.

Components of each of the three movements can be combined, so that the robot moves forward, sideways and turns simultaneously. The width of the rectangular locus and the speed at which it is traversed by the paw determine the speed at which the robot moves.

Twelve parameters influence the leg movements for a particular walk style. We considered automating the search for the best parameter settings. However, we were concerned about the wear on the robot motors and leg joints, given the long periods of training required. Hornby, *et al* [1] report training times of 25 hours per evolutionary run using their genetic algorithm (GA). The approach we adopted was to manually adjust the parameters after a number of runs of observing and measuring the performance. Unlike gradient ascent or a GA, we were able to adjust the parameters using our judgement and knowledge about the robot's dynamics.

A refinement that increased ground speed considerably was the introduction of a canter. The canter sinusoidally raises and lowers the robot's body by 10mm synchronised with the trot cycle. The parameters were manually tuned so that the robot was able to reach speeds of 1200cm/min. This compares to 900cm/min achieved in [1] using a genetic algorithm, which was reported to have improved on the

previously fastest hand developed gait of 660cm/sec. The camera is not as steady in this type of walk because of the additional canter movement.

5 Behaviours

The team consists of two Forwards and a Goalkeeper. Each role has its own set of strategies, described below.

5.1 The Forward

The pseudo code below describes the Forward's high-level strategy.

```

if see team mate at a distance < 15 cm
    backup
else if no ball in world model
    findBall;
else if canKickBall
    kickBall;
else if canChargeBall
    chargeBall;
else getBehindBall;

```

There are five main skills namely *backup*, *findBall*, *kickBall*, *chargeBall* and *getBehindBall*, which we now explain.

backup

When a robot sees one of its teammates nearby, it backs away, reducing the chances of our robots interfering with each other. The backup behaviour tends to keep one robot on the “wing” of its teammate, which effectively makes one robot wait for the ball. If the robot with the ball loses it, the “wing” can quickly pick it up.

findBall

When the robot does not know where the ball is, it moves the head in a rectangular path searching for the ball. The head is moved in a direction so that if it hits the ball in the lower scan, the ball will roll in the direction of the target goal. If the ball is still not found, the robot turns 45°. The direction of the turn is also chosen so that if the robot accidentally hits the ball, it will hit the ball towards target goal. The robot continues alternately turning and scanning the head until it finds the ball or when it has made six turning moves. When it has turned 45° six times without seeing the ball, it is likely that the ball is obstructed or outside its field of vision. The robot then goes to a defensive position and spins on the spot until it sees the ball.

kickBall

The kick is intended to be accurate, easy to line up and powerful. We tried many variants such as using a single leg or dropping the head on ball, but found that bringing both fore-limbs down in parallel on the ball best met the objectives.

For the kick to be effective, the ball has to be positioned between the front legs of the robot and touching the chest. The kick is implemented as two sets of absolute leg positions executed sequentially. The motor joint positions were found by conducting many trials and adjusting them until the best performance was achieved.

It was found that when the robot is near the edge of the field, kicking is not very effective. When setting up to kick the ball, the robot approaches at approximately 80% of maximum speed and maintains a heading to the ball between $\pm 15^\circ$. The robot only tracks the ball's movement with the head pan while the head tilt is held constant so that the head just clears the ball. Upon losing sight of the ball, the robot transitions into a very low stance with the head placed directly above the ball. The mouth is then used to sense the presence of the ball, as it cannot be seen with the camera. If the mouth does not sense the ball or the ball is identified by the vision system, the kick is aborted. Once in possession of the ball, the robot can take a limited number of rotational steps, set at two complete steps for RoboCup 2000, to align the goal before triggering the kicking motion.

chargeBall

When the robot has the ball near the target goal, then it is worth taking time to line up on the goal. However, if the robot is far from the target, it is more effective to simply knock the ball into the opponent's half. This wastes little time and does not allow opponents the chance to take the ball away. Thus, the robot only tries to line up the goal and the ball in region in front of the target goal before it runs at the ball. There are two skills that the robot can use to charge with the ball namely *dribbling* and *head butting*.

Dribbling is invoked when the robot is facing the opponents' half, the ball is close and in front of the robot. The robot then starts walking forward with the head just above the ball, using the mouth to sense if it has the ball. If, after few steps, the robot does not sense the ball, it stops and takes a few steps backwards to try to bring the ball into view. If it sees the ball, the robot continues to walk with the ball at its "chest". Otherwise, this mode is exited.

If the ball is not in position to dribble, the robot will head butt or bunt the ball. Although head butting is not accurate, we only need to knock the ball to the other half. The bunting strategy is very simple in that the robot walks directly at the ball with full range head tracking enabled. Directional control of the ball is obtained by inducing a component of sideways walking in proportion to the heading of the target, relative to the robot.

With these strategies, the robots keep the ball moving constantly giving less chance for opponents to control the ball.

GetBehindBall

The ball circling technique used in both the Goalkeeper and Forward, defines parameters for the walk that drive the robot from any position on the field to a position directly behind the ball. This circling technique involves no aggressive transitions in the robot's movement, always keeps the ball in sight, and keeps the robot's body pointing toward the ball.

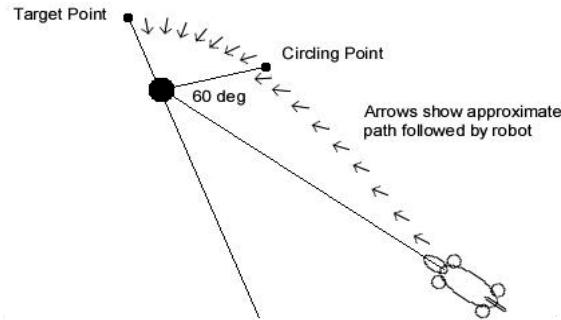


Fig. 4. Circling Skill

Ball circling is specified by two points (Figure 4). The target point is the intended destination and the circling point deflects the path around the ball. To perform the skill, the robot is simply driven towards the closer of the circle and target point, while the body is oriented towards the ball.

If the robot is trying to circle around the ball to line up the goal and it sees an opponent nearby, it will become more aggressive. The robot will run at the ball immediately as long as it is not facing its own goal.

5.2 The Goalkeeper

The goalkeeper has three behaviours used in defending the goal.

Finding the Ball

Finding the ball begins with a 360° rotation in the direction that would knock a ball stuck behind the robot away from the goal. Therefore, the robot will rotate clockwise on the left side of the field, otherwise anti-clockwise. During the rotation, the robot raises and lowers the head quickly to perform a combined short and long-range search. If the ball is not found during the rotation, the head of the robot begins following a rectangular search path scanning for the ball. At the same time, the robot orients itself facing directly away from the goal it is defending and walks backwards, to the goal. Once close to the goal, the goalie turns to face the opposing goal.

Tracking the Ball and Acquiring a Defensive Position

Once the ball has been found, the robot enters its tracking and defending mode. In this mode, the robot places itself on the direct line between the ball and the defended goal, at a position 45cm from the defended goal. As the ball position changes, the robot tracks along a semicircle around the defended goal, keeping the body oriented towards the ball. While tracking the ball, the robot oscillates the head side to side as much as it can, without losing the ball, to try to maximise the chances of seeing landmarks and help maintain localisation. However, watching the ball always takes precedence over localisation.

Clearing the ball

Clearing the ball is activated when the ball comes within 80cm of the goal or the ball enters the penalty area. It ends when the ball is kicked, lost or moves more than 120cm from the goal. Upon deciding to clear the ball, the robot determines whether it can directly attack the ball or should reposition itself behind the ball. Once the robot is clear to attack the ball, it aligns itself with the ball to kick it. On approach to the ball, if the ball gets out of alignment, the robot aborts its kick and simply bunts the ball with the front of the head.

6 Conclusions and Future Development

In reviewing why the UNSW team was successful, we can identify technical advances in locomotion, vision and localisation and the repertoire of behaviours that were developed. Some practical considerations also contributed to the team's win.

Following our experience in 1999, we decided that we would play regular games between teams of two robots. As we devised new strategies, these were played off against each other. We also insisted that whenever testing a new behaviour, we should have as many robots on the field as possible. These "management decisions" ensured that we tested the robots, as much as possible, under competition conditions and thus were able to discover and overcome many problems.

One consequence of this approach was that as a new special case was encountered, we introduced a new fix. It is evident from the description of our software that there are many *ad hoc* solutions to various problems. Thus, it might be argued that we are not learning much of general interest to robotics because we have not pursued solutions that are more general.

We believe there is much of value to be learned from our effort. It is clear that in a highly dynamic environment, speed of perception, decision making and action are essential. Our experience has been that implementations of very general approaches to these problems tend to be slow, whereas, problem-specific solutions are simple and fast. However, developing these problem-specific solutions is very labour intensive. Thus, one of the areas of future research for us will be in finding methods for automating the construction of domain-specific behaviour. The generality of our approach will, hopefully, be in the learning, and not in a particular skill.

References

1. Hornby, G. S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O. (2000) *Evolving Robust Gaits with Aibo*. IEEE International Conference on Robotics and Automation. pp. 3040-3045.
2. Dalgiesh, J., Lawther, M. (1999). *Playing Soccer with Quadruped Robots*. Computer Engineering Thesis, University of New South Wales.
3. Hengst, B., Ibbotson, D., Pham., Sammut, C. (2000). *UNSW RoboCup2000 Team Report*. <http://www.cse.unsw.edu.au/~robocup>.

Adaptive Path Planner for Highly Dynamic Environments

Jacky Baltes and Nicholas Hildreth

Centre for Image Technology and Robotics
University of Auckland, Auckland
New Zealand
j.baltes@auckland.ac.nz
<http://www.citr.auckland.ac.nz/~jacky>

Abstract. This paper describes adaptive path planning, a novel approach to path planning for car-like mobile robots. Instead of creating a new plan from scratch, whenever changes in the environment invalidate the current plan, the adaptive path planner attempts to adapt the old plan to the new situation. The paper proposes an efficient representation for path that is easily amendable to adaptation. Associated with the path planner is a set of repair strategies. These repair strategies are local methods to fix a plan to compensate for object movement in the domain. The repair strategies are specific and have a high probability of being able to fix a plan. An empirical evaluation shows that adaptive path planning is suitable to highly dynamic domains, such as RoboCup. Adaptive path planning reduces the cumulative planning time by a factor of 2.7 compared to Bicchi's planner. At the same time, the quality of the plans generated by the adaptive path planner were similar to those generated by Bicchi's planner.

1 Introduction

Navigation is an important task for any mobile robot. Before a robot can affect the world in any sensible way, it must be at the right place at the right time. The navigation task can be broken down into a number of important sub tasks: localization, path planning, and plan execution.

Path planning is the problem of creating a collision free path through a set of obstacles from an initial to a goal position. A simple example is planning to go from the entrance of the CITR lab to the desk in room 305 on the third floor. Path planners can be categorized based on whether global or local information about the environment is available and whether objects are static or dynamic.

The F180 league in RoboCup, a game of soccer between autonomous robots, is a good example of a global path planning domain with dynamic obstacles. The RoboCup domain has a number of properties, which makes it interesting for real world planning. It features a high density of highly dynamic obstacles. This means that an agent has to re-plan often, since plans may be invalidated through objects moving. In fact, RoboCup features an active opponent that tries

to prevent a robot from executing its plan (e.g., a defender tries to prevent a striker from moving into a position in front of the goal) successfully. A robot designed to perform real world applications, such as search and rescue in dangerous environments will have to cope with similar environments.

A path planner in such a domain needs to be able to react quickly to changes in the environment. Furthermore, the path planner needs to be able to return at least an approximate plan (a best guess) immediately, that is it must be an *anytime* path planner. On the other hand, as more planning time is available, a better plan should be returned. For example, if a truck is barreling down on a robot, the planner must return an escape plan (run left) immediately. With more planning time, a complex plan for the robot to reach its destination can be developed.

This paper discusses a novel approach, adaptive path planning, for global path planning in dynamic domains. Although the approach is applicable to holonomic robots, the description focuses on car-like robots, since path planning is a more interesting problem for these non-holonomic robots.

The motivation for the adaptive path planner is to reuse previous planning work and to adapt a plan to a changing world rather than to re-plan from scratch. The adaptive planner has been tested on sample problems derived from the RoboCup competitions and during the actual competition at RoboCup-99. An empirical evaluation shows that the cumulative runtime of the adaptive path planner is 2.7 times faster than that of a path planner based on Bicchi's work, but optimized for the RoboCup domain.

Section 2 gives a brief introduction to related work. Section 3 describes the design and implementation of the adaptive path planner. The section focuses on the novel plan representation as well as the set of repair strategies. The results of the empirical evaluation are shown in section 4. Section 5 concludes and discusses how the described ideas can be applied to holonomic robots.

2 Literature Review

There have been many different approaches to path planning, both for holonomic and car-like robots. The major approaches for holonomic path planners are skeletonization methods (e.g., visibility graphs, Voroni diagrams, or road maps), cell decomposition (e.g., approximate or exact), or local approaches (e.g., potential fields, landmarks) [6, 4].

There are a number of problems with the path planning methods listed above. Firstly, these algorithms have a high computational cost. Secondly, these algorithms are not anytime path planners, since they require a large pre processing step. For example, quad tree decomposition, a popular approximate cell decomposition, has to break the domain up into cells before any planning can occur. Therefore, visibility graphs are the most popular global path planning methods.

A number of non-holonomic path planning methods are two stage approaches; a holonomic path planner is used to create a holonomic path, which is then converted into a non-holonomic path. Two stage approaches are not suitable to

an ytime planning, since there is no gurantee that the holonomic plan can be converted into a non-holonomic one.

Other approaches use variations of standard holonomic path planners to create a non-holonomic path directly. An example of this approach is Bicchi's path planner, which is an extension of visibility graph algorithms [1]. The idea behind Bicchi's path planner is to add circles of minimum turn radius around all vertices of obstacles. The planner then finds a non-holonomic path by searching the connections between circles from the start to the goal. An example of Bicchi's path planner is shown in Fig. 1.

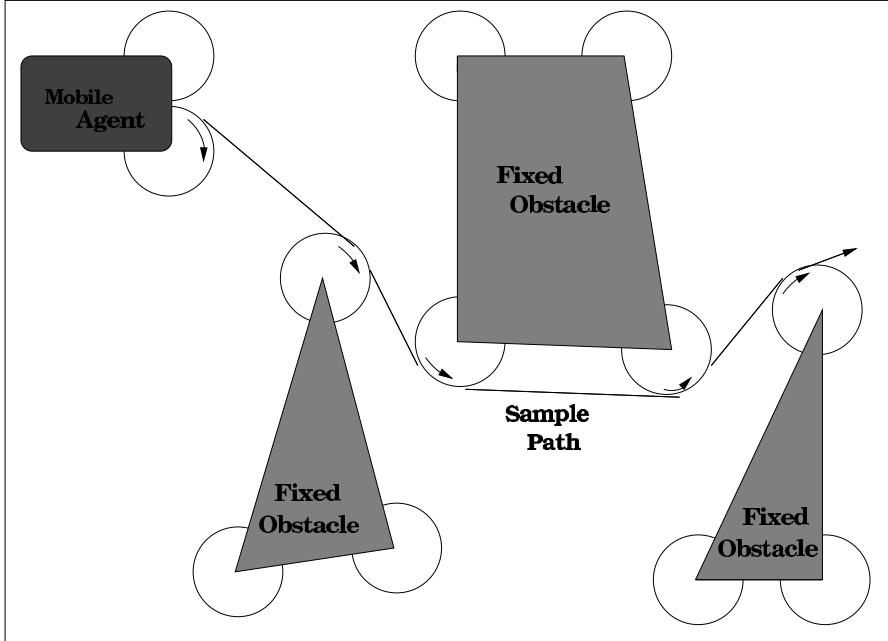


Fig. 1. Bicchi's Path Planner. A path consists of straightlines between circles located at the vertices of the obstacles.

There are a number of heuristics that can be exploited in the RoboCup domain to speed up the search step in Bicchi's path planner. For example, since all obstacles are modeled as circles, only a single circle around this obstacle is needed. Also, the search heuristic can construct the turning circles dynamically instead of creating all the connecting lines in a preprocessing step.

3 Adaptive Path Planning

Observation of a RoboCup quickly leads to the realization that most object movements from one frame to the next are rather small. Although it is true that a tiny movement can invalidate a plan or indeed make this particular planning

problem unsolvable (e.g., two opponent robots are covering the ball), in most cases, the necessary changes to the plan are minimal. So, the main motivation behind adaptive path planning is that:

- Path planning is an expensive operation, so the result of this work should be reused if possible.
- The result or output of a path planner is a partial or complete path
- Assuming that changes in the domain are small between individual planning episodes, the best plan for the current situation will be structurally similar to that for the previous situation.

This motivation is similar to Hammond's case-based planning ([2]) with some important differences. Firstly, case-based planning assumes that a plan database exists with previous plans and that the most similar plan to the current situation can be found in the database. Case retrieval from the database uses commonly a similarity metric. Secondly, the database of previous plans needs to be maintained; new plans need to be added so they can be reused in the future or old plans that are not useful must be removed. So, a lot of work on case-based planning focuses on the design of suitable similarity metrics (i.e., how to find a similar case to the current one) and on database maintenance policies (should a case be added to the database or deleted).

In adaptive path planning, we assume the existence of an albeit slow static path planner that can be used to create an initial plan. Furthermore, the previous plan is the most similar one to the current situation and that, therefore, there is no need to maintain a plan database.

3.1 Path Representation

At the heart of any path planner is the plan representation. A plan consists of a sequence of path segments. Bicchi's path planner and most other planners use a representation where each path segment is either a straight line or a maximal turn to the right or left. These representations are based on a result by Reed and Shepp, that proves that any shortest path for a vehicle consists of exactly three types of segments: straight lines, full left turn, or full right turn ([5]).

However, the shortest path may not always be the optimal one, since it may lead too close to the obstacle, may include many reversals, or may result in the robot driving backwards for long stretches.

Furthermore, this representation makes it difficult to adapt a path, since the adaptations will need to be specific for a given segment type.

To simplify the adaptation, the adaptive path planner uses a uniform representation for all path segments. Each segment is an arc that contains the following information:

- start point I
- initial bearing α
- traversal direction D specifies whether the robot should travel backwards or forwards along the segment.

- length of the arc segment L
- radius of the segment R . Straight lines are represented as arcs with a large radius. There is also a minimum radius since car-like robots are limited in the radius of their turns.
- time limit to traverse the segment T . This information is used by the strategic component. If we can not reach a point in the desired time (e.g., be in place to receive a pass), it is better to recognize this fact and pursue a different course of action (e.g., play defensively), instead of reaching the goal 30 seconds after the pass was made.
- A possibly empty attachment A . An attachment is used to attach an object to a path segment, so that if the object moves, all attached path segments will move as well.

This representation, shown in Fig. 2, proved very useful, because plans in this representation can be easily adapted to compensate for movements of objects or goal locations in the domain.

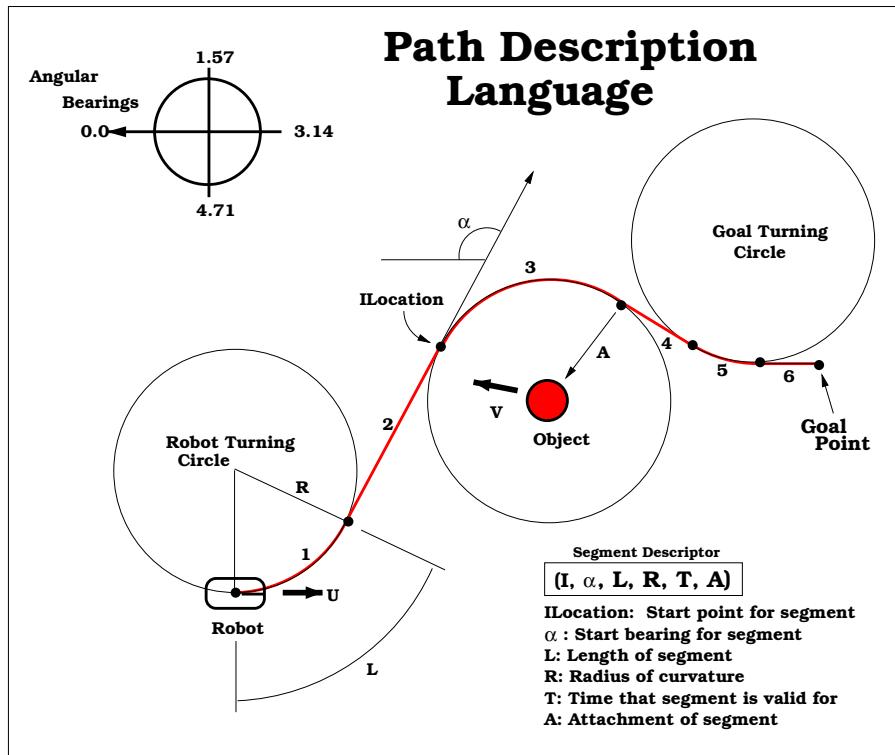


Fig. 2. Path Description Language

Objects are attached and detached from path segments dynamically. If an obstacle moves too close to a path, the path is split at the closest point to the

obstacle and a new segment is inserted which is attached to the object. As the object continues to move the attached segment will move as well. Once the object is too far from the path, the object is detached from the path segment.

3.2 Repair Strategies

Any movement in the domain results in attached path segments being moved as well, and thus may create a discontinuity, a so-called disjunction, in the path (see Fig. 3). The assumption is that repairing these disjunctions is cheaper than creating a new path from scratch.

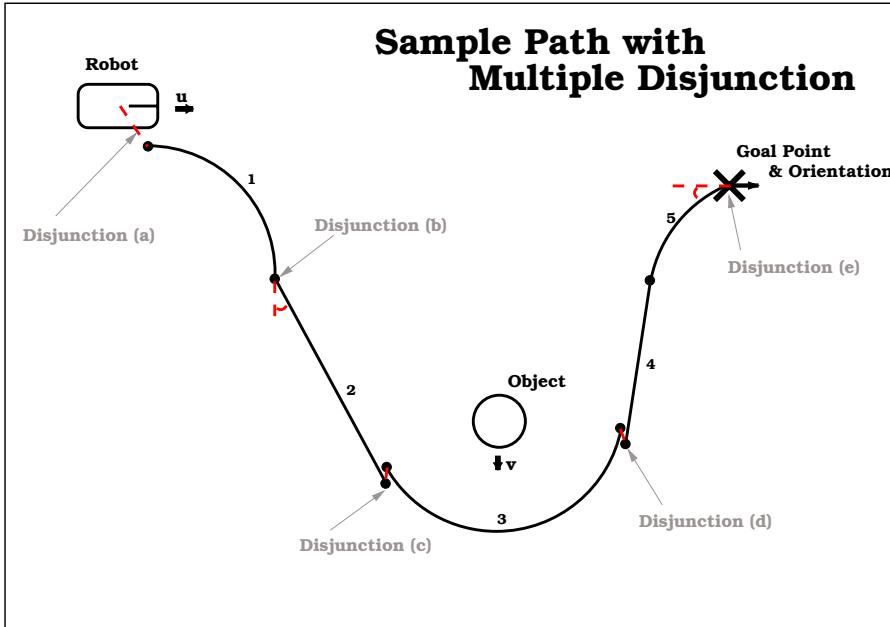


Fig. 3. An Object Movement that Results in a Disjunction

There are two types of disjunctions: distance disjunctions (break in the path) and angle disjunctions (continuous path, but discontinuity in the first order derivative).

The adaptive path planner starts with the largest disjunction and attempts to adapt the plan to fix the disjunction. In this process, new, but smaller disjunctions may be introduced into the path. The repair methods are applied using a standard A^* search algorithm that uses the complexity of the repair as a heuristic function. For example, changing the length of a segment is easier than translating a segment to a different location.

To fix these disjunctions, the adaptive path planner contains a library of path repair methods. These path repair methods are highly specific repairs that have a high chance of success and of reducing the size of the disjunction.

The following classes of repair methods are implemented in the adaptive path planner:

- **P ositional Adjustment:** Changes in the start and end position and bearing of a segment. By themselves these repair strategies are not very useful, but in combination with others (e.g., shape adjustment) they can fix many plans.
- **Shape Adjustment:** These repair methods change the length or curvature of a segment. For example, a tight turn can be converted into a gentler turn.
- **T ypeAdjustment:** These repair strategies change the sign of the curvature, so a left turn can be converted into a straight line or a right turn. Also, the traversal direction of a circle can be swapped.
- **Segment Structure Adjustment:** These repair methods use segment insertion, segment breaks, and segment deletion to change the structure of the plan.
- **Plan Justification Adjustment:** These repair methods remove unnecessary plan segments or object attachments from the plan and thus are an optimizing post-processing step.

A shape adjusting adaptation is shown in Fig. 4. An angular disjunction is fixed by simultaneously rotating and stretching the line segment and shortening the circle.

A complete list of adaptation methods can be found in [3].

4 Evaluation

To evaluate the performance of the adaptive path planner, we compared the cumulative runtime and the success rate of the adaptive path planner against that of two popular non-holonomic path planners.

The evaluation was based on a set of case studies, which were derived from situations during robotic soccer competitions. Figure 5 shows a case study of a situation that was inspired by the RoboCup competitions. It shows how the adaptive path planner changes the path to compensate for a moving ball and an interfering object. The robot starts at the left side of the field. The goal location is on the right side of the field. The goal initially moves upwards until it rebounds and starts moving downwards. An obstacle moves upwards in the center of the field. There are also two static obstacles in the domain.

Figure 5 shows a graph of the planning time of the adaptive path planner versus that of an optimized Bicchi planner. As can be seen, the adaptive planner uses very little time in most cases. The adaptive planner uses more time in cycle 103. That is because at this moment the obstacle intrudes into the path and a new segment attached to this object needs to be inserted. Correspondingly the peaks at cycle 200 and 300 correspond to the path planner having to change the traversal direction around the obstacle as well as avoiding the upper obstacle.

The repair strategies had a very high success rate (> 99%), since the adaptive path planner only failed in six of the 1000 planning episodes. Should the adaptive planner fail, it may or may not call the static planner in the current cycle. This

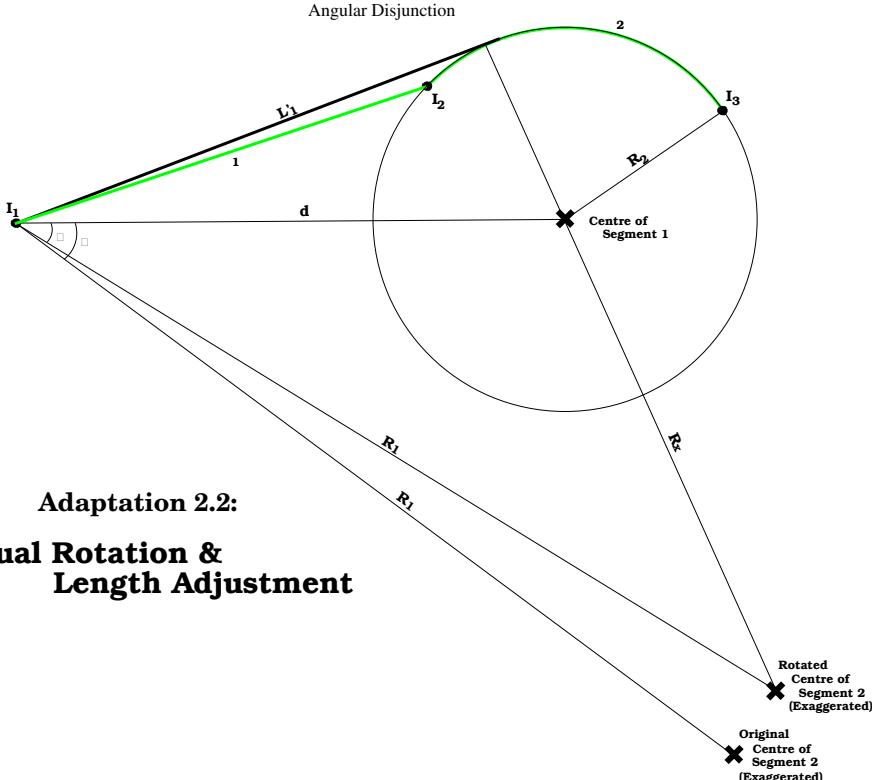


Fig. 4. A Shape Adjustment Adaptation

decision depends on the amount of computation that was done in the current cycle and the expected cost of repairing the plan.

Nevertheless, the cumulative running time for the path planner in this problem is greatly reduced. To evaluate the cumulative runtime further, we randomly created a set of real world problems similar to the one shown above. After the problem was set up using our mobile robots, the path planning activity of the robot was recorded until it reached the goal. These tests used a complex domain with 11 objects.

Table 1 shows that the adaptive path planner is significantly faster (a factor of 2.7) than the optimized Bicchi planner. Often the adaptive path planner is orders of magnitude faster. Only in situation 7, did the adaptive path planner perform worse.

The length of the plan is similar for the adaptive planner and the optimized Bicchi planner. Note that the optimized Bicchi planner uses heuristics to prefer simple path (few reversals, few segments), and not necessarily the shortest path as the original Bicchi planner.

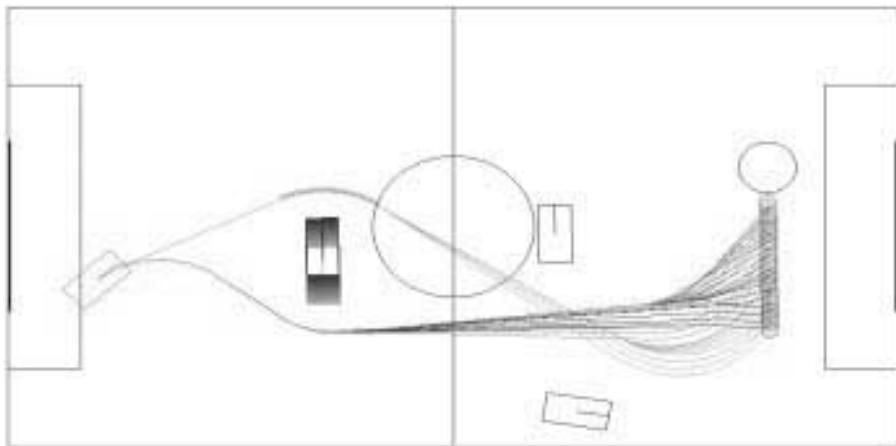


Fig. 5. Case Study Derived From Observed RoboCup Competition

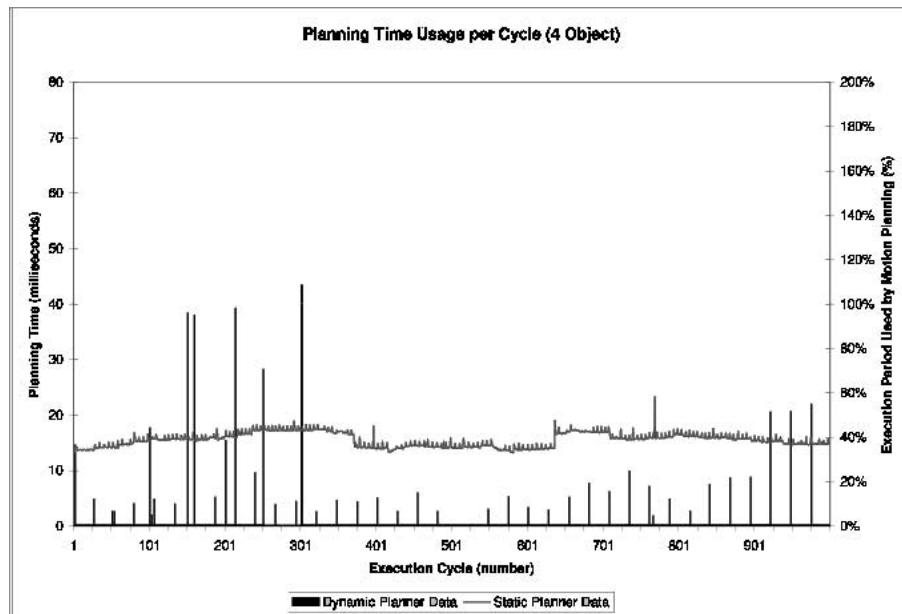


Fig. 6. A comparison of the planning times for the episode shown in Fig. 5

5 Conclusion

This paper describes a novel approach to path planning for car-like mobile robot, that attempts to reuse previous planning work as much as possible when planning for a new situation.

Sit.	OPTIMIZED BICCHI PLANNER			ADAPTIVE PLANNER		
	Time (ms)	Outcome	Length (mm)	Time (ms)	Outcome	Length (mm)
1	276ms	Success	2750mm	13ms	Success	1912mm
2	93ms	Success	1480mm	12ms	Success	1484mm
3	72ms	Success	988mm	2ms	Success	993mm
4	304ms	Success	2287mm	2ms	Success	2282mm
5	189ms	Success	2538mm	10ms	Success	2107mm
				23ms	Success	2151mm
				18ms	Success	2161mm
6	264ms	Success	2347mm	59ms	Success	1678mm
7	43ms	Success	2023mm	243ms	TimeOut	1880mm
				40ms	Success	1866mm
8	281ms	Success	1758mm ¹	12ms	Success	2758mm
				4ms	Success	2755mm
9	381ms	Success	2755mm ¹	242ms	TimeOut	3022mm
				8ms	Success	3147mm

Table1. Cumulative Runtime for randomly generated real world examples with 11 objects.

¹This plan uses both forward and backward movement by the robot to complete.

A set of adaptation methods, a set of highly specific repair strategies, has been developed that have a high rate of success.

An empirical evaluation using a set planning episodes derived from real world problems shows that the adaptive planner is significantly faster than other state of the art non-holonomic planners.

The basic idea of adaptive planning is independent of the actual robot control. The only change required is the plan representation and the associated set of adaptation methods.

References

1. Antonio Bicchi, Giuseppe Casalino, and Corrado Santilli. Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1349–1354, 1995.
2. Kristian J. Hammond. *Case Based Planning*. Academic Press Inc., 1989.
3. Nicholas Hildreth. Adaptive path planning for real-time systems. Master's thesis, University of Auckland, July 2000.
4. J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
5. J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2), 1990.
6. Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*, chapter 20, pages 598–624. Prentice-Hall Inc., Englewood Cliffs, New Jersey 07632, 1995.

Communication and Coordination among heterogeneous Mid-size players: ART99

Claudio Castelpietra¹, Luca Iocchi¹, Daniele Nardi¹, Maurizio Piaggio²,
Alessandro Scalzo² and Antonio Sgorbissa²

¹ Dipartimento di Informatica e Sistemistica Università “La Sapienza”, Roma, Italy

² Dipartimento di Informatica, Sistemistica e Telematica Università di Genova, Italy

Abstract. Distributed coordination among *robotic soccer agents* has been considered in the recent years within the framework offered by the RoboCup competitions, mostly in the simulation and F-180 leagues. In this paper we describe the methods and the results achieved in coordinating the players of the ART team participating in the F-2000 league. The team is formed by several heterogeneous robots having different mechanics, different sensors, different control software, and, in general, different abilities for playing soccer. The coordination framework we have developed has been successfully applied during the 1999 official competitions allowing both for a significant improvement of the overall team performance and for a complete interchangeability of all the robots.

1 Introduction

Distributed coordination of *robotic agents* [7] has been considered as one of the central research issues in the RoboCup competitions [2, 4]. In a highly dynamic and uncertain environment such as the one provided by RoboCup games, the centralized coordination of activities underlying much of the work in Robotics does not seem to be adequate. In particular, the possible communication failures as well as the difficulty of constructing a global reliable view of the environment, require full autonomy on each robot.

The coordination problem in the context of RoboCup was first faced in the simulation league [6, 8], where it plays a central role because of the high number of players (11). In the small size league coordination can take advantage of the availability of global information on the game status, since a centralized vision system and elaboration is used [12]. In the F-2000 (middle size) league, although the number of players is 4 (including the goal keeper), coordination among the players is still a critical issue because the dynamics of the game make it necessary to avoid interferences among players’ actions. However, the distinguishing feature of the F-2000 league is the difficulty of reconstructing global information about the environment and thus coordination needs to be achieved without laying down drastic prerequisites on the knowledge of the single players.

The implementation of cooperative strategies for a team of robots can be addressed by relying on explicit communication [13] or by exploiting implicit

communication and emergent behaviors [5]. Our choice has been to rely on explicit communication for implementing cooperative strategies; however, due to the frequent communication failures the robots must depend neither on communication, nor on information provided by other robots.

The Azzurra Robot Team (ART) [9] of robotic soccer players, participating in the RoboCup competitions in the F-2000 category, is composed of different players (both in the hardware and in the software) developed in various Italian universities. Because of this kind of organization, coordination among the ART players requires not only a distributed coordination protocol, but also a very flexible one, that allows the coach to accommodate the various configurations that can arise by forming teams with different basic features. Moreover, results on the coordination between players and the goal-keeper are described in [1].

Summarizing the hypotheses underlying the coordination problem are:

1. *Communication-based coordination*: exploit the use of communication among the players to improve team performances, allowing the robots to acquire more information and to self-organize in a more reliable way.
2. *Autonomy in coordination*: the players are capable to perform their task, possibly in a degraded way, even in case of lack of communication.
3. *Heterogeneity in the multi agent system*: the players are heterogeneous both from hardware and software viewpoints.

Besides the above constraints, coordination in ART has been designed to deal both with roles (defender, attacker etc.) and with strategies (defensive, offensive). While the strategic level is currently demanded to an external selection (the human coach), roles are dynamically assigned (see [12]) to the various team elements during the game. In the following sections we describe the communication infrastructure and the coordination protocol, we discuss the experimental results on coordination of the ART players and finally draw some conclusions.

2 Communication infrastructure

The communication infrastructure is strictly related to the ETHNOS [10] software architecture whose protocol was at the base of inter-robot communication in ART. ETHNOS exploits a message based communication protocol called EIEP (Expert Information Exchange Protocol) which deals transparently both with inter process communication within a single robot and with inter-robot communication. In the EIEP the robots are allowed to subscribe to communication clubs. For example, we may envisage a single club to which the different players belong or different clubs, to which only players which are performing a cooperative activity belong. Messages are exchanged with a publish/subscribe technique in which subscription and publication can thus be distinguished in internal, external in a specific club or external in all clubs. Whenever a message is published ETHNOS transparently and dynamically distributes the messages to the appropriate subscribed receivers. Figure 1 shows an example configuration that we have tested. In particular we are allowing the robots to communicate in a single

club the team club - (to which all of them have subscribed) and with an external supervisor (the coach) which monitors the activity of all the players for displaying and debugging the team activity during a match.

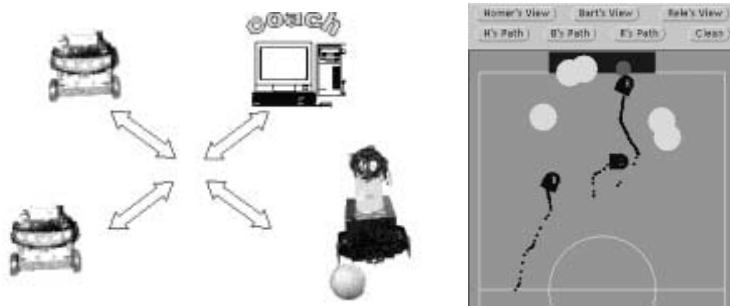


Fig. 1. Left: example of multi-robot ETHNOS configuration, Right: monitoring the team activity.

It is worth noticing that, because of EIEP protocol, whenever we want to add (remove) a player to (from) the team, it is not necessary to explicitly inform each player about the modifications in the team's composition. In fact, the players just have to agree about the type of information they are ready to send and receive: ETHNOS automatically updates the database of each club's members and consequently dynamically dispatches messages from senders to the appropriate receivers. This has been very important in ART in which there were more than four types of robots available, with different characteristics of play, and thus different team compositions were selected for the single matches and also modified during the matches to better contrast the opponent. Moreover since in the RoboCup (and in general in mobile robotics) network communication is often wireless (i.e. radio link, Wavelan(c), etc.), due to noise or interference transmission packets are sometimes lost. In this context, both TCP-IP and UDP-IP based communication cannot be used: the former because it is intrinsically not efficient in a noisy environment; the latter because it does not guarantee the arrival of a message. For this reason we have also designed a protocol for this type of applications, called EEUDP (Ethnos Extended UDP) because, based on the UDP, it extends it with the necessary properties. The EEUDP allows the transmission of messages with different priorities. The minimum priority corresponds to the basic UDP (there is no guarantee on the message arrival) and should be used for data of little importance or data that is frequently updated (for example the robot position in the environment that is periodically published). The maximum priority is similar to TCP because the message is sent until its reception is acknowledged. However, it differs because it does not guarantee that the order of arrival of the different messages is identical to the order in which they have been sent (irrelevant in ETHNOS applications because every message is independent of the others), which is the cause of the major TCP overhead. Different in-between priorities allow the re-transmission of a message until its reception is acknowledged for different time periods (i.e. 5 ms, 10 ms, etc.).

3 Coordination

The major issues that we have addressed in the coordination protocol are the dynamic assignment of roles and the team strategy. We adopted a *formation/role* system similar to the one described in [12, 11]. A *formation* decomposes the task space defining a set of *roles*. Each robot has the knowledge necessary to play any role, therefore robots can switch roles on the fly, if needed. However, the implementation choices are different due to the difference in the application domain: in the simulation league the focus is on communication failures, while in the F-2000 league the use of this kind of coordination is needed to avoid that more than one robot of the same team tries to perform the same action (e.g. to go to the ball) and to occupy properly various parts of the field.

Therefore, the basic formation of an F-2000 team requires that one robot takes the role of going to the ball, another that of defending and another that of supporting the attack. However, other formations are possible depending on the kind of strategy adopted (offensive, defensive) and on the need to handle special situations such as for example the malfunctioning of the goal keeper.

In [13] a coordination method used in the F-2000 league is presented: the basic idea is that all the robots try to get the ball and the first one who reaches it asks the other robots to stay away from the ball. We believe that the dynamic roles' assignment is equally effective in gaining ball possession while it allows the robots to better occupy the various parts of the field.

3.1 Coordination protocol

Inside each robot, the coordination module takes its input from the coordination messages. The output is the formation that the team will adopt and the role assigned to the robot.

The computation for the coordination protocol is distributed. The protocol is robust because it relies on a little amount of transmitted data. The coordination protocol includes the following two steps:

Step 1: Formation selection The robots have at their disposal a set of pre-computed formations that can be used depending on the current state of the environment. The automatic generation of these formations from a declarative specification of the robots' abilities is described in detail in [3]. The robots also possess the rules to determine which formation should be adopted on the basis of the environment configuration. But, considering that each robot's data do not necessarily coincide with those of the others, the robots may determine different formations. Therefore each robot adopts the formation that gets the majority of votes among those proposed by the team.

In order to avoid the risk of oscillations in the common decision relevant to the formation due to varying numerical parameters or to the message transmission delays, each robot has to moderate the frequency of changes in proposed formation. The main way to do it is to use some kind a decision-stabilization method as discussed below.

Step 2: Roles selection This step implements dynamic role assignment through explicit communication of the “utility values”. Such values indicate the usefulness, with regard to the whole team, of each role to be assigned to each robot. This is achieved by the definition of a number of *utility functions* (specific for every role) that every robot evaluates given its current local information about the environment. By comparing these values, each member of the team is able to establish the same set of assignments (robot↔role) to be immediately adopted.

More specifically, suppose we have n robots $\{R_1, \dots, R_n\}$ and m roles $\{r_1, \dots, r_m\}$. The roles are ordered with respect to importance, i.e. in the current formation assigning r_i is more important than assigning r_{i+1} . This means that if $n < m$ then only the first n roles will be assigned, while if $n > m$ then $m - n$ robots will not be assigned any task. In the RoboCup scenario we always guarantee that $n \leq m$, so that every robot will always be assigned a role.

Let $f_j(i)$ be the value of the utility function, computed by robot R_i for the role r_j and $A(i) = j$ denote that the robot R_i is assigned the role r_j .

The method for dynamic role assignment requires that each robot R_p computes the following:

1. For each role r_j compute $f_j(p)$ and broadcast the computed value
2. For each robot R_i ($i \neq p$) and for each role r_j , collect $f_j(i)$
3. $\mathcal{L} = \emptyset$ (Empty the list of assigned robots)
4. For each role r_j do
 - (a) $h =$ the robot R_i ($i \notin \mathcal{L}$) such that $f_j(i)$ has the higher value
 - (b) if $h = p$ then $A(p) = j$ (my role is r_j)
 - (c) $\mathcal{L} = \mathcal{L} \cup \{h\}$

It's easy to see that every role is assigned to only one robot and that every robot is assigned to only one role. The reason is that on every cycle of the algorithm a different assignment $A(i) = j$ is done: j changes at each cycle and robots already included in the set \mathcal{L} of assigned robots cannot be chosen for further assignments.

3.2 Critical factors for effective implementation

In order to obtain an effective application of the above method a number of issues need to be dealt with properly: the stability of decisions, the recognition of situations where a robot cannot successfully accomplish the task associated with the assigned role and communication failures.

It is important for all the high level decisions taken by the robot, including those regarding coordination, to be stable with respect to possible oscillations of the numerical parameters upon which they depend (see [6] par.5.2).

We have chosen a method of stabilizing decisions by means of *hysteresis* (see Fig. 2), which amounts to smoothing the changes in the parameter values.

This technique prevents a numerical parameter's oscillation from causing oscillations in high level decisions. In the case of coordination, for instance, if

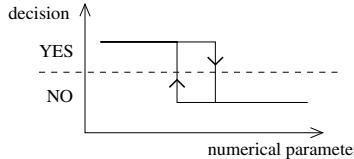


Fig. 2. The hysteresis mechanism in decisions.

at a certain instant robot R_i covers role r_j , its utility function $f_j(i)$ for role r_j returns a higher value.

Another critical factor for the correct operation of coordination is the capability of each element to realize a sudden difficulty in performing its task. For instance, a robot that is moving towards the ball can get stuck on its way. Once it has realized this circumstance, all its utility functions must return low values so that the important roles can be assigned to other robots.

Finally, if all the robots possess the same data (i.e. communications are working correctly), they will compute the same assignment, but in case of a great loss of transmitted data due to interferences, the robots may have slightly inconsistent data. Therefore, there could be roles temporarily assigned to more than one robot or not assigned at all. However, holes in data transmission last in general fractions of a second. So, if we assume that the values of the "utility functions" do not change sharply, the correct use of the hysteresis system described guarantees that the roles will be correctly assigned almost always (as shown by the experimental data we have collected during the games and that are discussed in the next section).

3.3 Example

As an example, we will analyze the numerical values of the utility function used to determine which player must move towards the ball. For convenience, the output is expressed as an estimated time, in seconds (consider that in this case the role is assigned to the one robot that obtains the lowest value):

- Base value: distance in millimeters from the ball, divided by the average robot speed, in mm/s.
- If the ball cannot be directly seen by the robot: add 1 second.
- If there are obstacles between the robot and the ball: add 1.4 seconds.
- If the ball must be passed to shoot in goal: add 180 degrees divided by the average rotational speed in deg/s.
- If the robot is stuck and cannot move: add 10 seconds.
- *Hysteresis*: add 1.8 seconds if at the moment one does not have the role for moving toward the ball.

4 Experimentation

The heterogeneity of the ART robots makes the experimental phase particularly demanding, since the exchanged information are computed and interpreted differently by each element of the team. As an example, consider the evaluation of reachability of the ball: each robot may have a mobile base of different capabilities and a set of behaviors with peculiar speed characteristics and, that notwithstanding, robots must calculate comparable numerical values.

A successful coordination of the team depends on a correct implementation of the coordination protocol and on a suitable calibration of a number of numerical parameters, such as the hysteresis and the utility functions parameters.

The experimentation of the coordination protocol must be done in stages which require the use of different tools. In particular, in order to realize an effective implementation of this approach and with the aim of evaluating this method, we have relied upon the use of different tools: a simulator, data analysis without play and during actual games, analysis of log files after the games.

The first and easier experimental setting is provided by a simulator. While a simulator, in general, cannot provide a precise characterization of all the aspects that influence the performance of the robot in the real environment, it can provide useful feedback to the design of the coordination system for actual robots. Through a simulator one can verify both the correctness of the protocol and the intended behaviour of the robots in each of the roles in different scenarios.

First experiments involving the robots may be done without play, with steady robots and moving the ball. At this stage one can adjust the discrepancies arising from differences in robots' implementations. In particular, one can compare transmitted values, due to different implementations of the utility functions, and differences can thus be spotted (together with major failures in the sensing capabilities of each robot).

The other experimental phase involving robots consists in looking at the game and in singling out the failures of the coordination system. For example a typical task is that of identifying situations where the most suitable player does not take the role of moving towards the ball and adjust parameters to restore the expected behaviour.

Finally, an analysis of the log files generated during the games is useful for highlighting possible situations in which coordination has not given good results. With this respect we made use of a 3D viewer for experimental data that allows for displaying several information about one or more robots.

4.1 Evaluation

The performance of the ART team at RoboCup 1999 [4] have provided substantial evidence that basic coordination among the team players has been successfully achieved.

The reliability of communication with the EIEP was experimentally verified. ETHNOS system allocates a maximum guaranteed and dedicated time to network communication. Since ETHNOS schedules all tasks in real-time according

to the Rate Monotonic scheduling policy, the dedicated time value is computed automatically on the basis of the schedulability analysis so that the real time execution of the whole set of tasks (i.e. user-defined and communication tasks) is guaranteed. Thus clearly the value depends on the computational load of the tasks in execution as well as the processor speed.

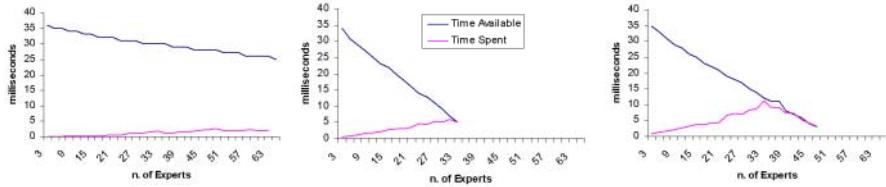


Fig. 3. Network Communication in ETHNOS. From left to right: the monitor, Relé, Homer (upper line = time available ; lower line = time spent).

In figure 3 the three graphs represent different machines (with different processing power) corresponding to two robots (Relé, and Homer) and a monitor, connected using Wavelans(c). The top line in the graph indicates the calculated time available each 100 ms for communication purposes. Clearly this value decreases as the number of tasks in execution increase (and so the computational load). The bottom line indicates the time spent in communication which also increases with the number of experts (this is because for this experiment we have assumed that the activity of every expert involves either transmission or reception of messages). In this way it is always possible to determine a priori whether the system is capable of both communicating information and executing in real time. For example, if we consider Relé, the limit situations in which the two lines converge is also the limit beyond which the schedulability analysis fails. Instead, if we consider Homer, real time scheduling is possible also beyond the intersecting point but only if we accept communication degradation. Degradation is in particular interesting. In fact we noticed that when communications were not reliable, the lack of coordination negatively impacts on the overall performance of the team, but it never happened that robots were stuck without knowing what to do, due the coordination protocol. For instance, in case two robots do not communicate with each other they will both assume the most important role for the current formation (i.e. they both will try to go to the ball or to tackle the opponent robot).

The coordination protocol has been evaluated by defining some measures that have been thought to be interesting in the RoboCup environment: the coverage of the field during the game, the average number of robots going to the ball at every time, role switching rate, etc. The following analysis has been computed from the data acquired during the games of European RoboCup 2000.

In Table 1a) we show the percentage of time in which at least one of the robot has occupied a zone of the field. Notice that the field has been properly occupied, even if there is not an explicit subdivision of the field in competence areas assigned to the robots.

Match	Forward	Middle	Backward		Match	GoToBall	Support	Defend
1	78.8%	68.7%	97.8%		1	82.9%	39.5%	98.2%
2	65.4%	68.8%	98.6%		2	84.6%	98.0%	80.8%
3	53.2%	54.5%	99.3%		3	87.6%	38.5%	90.0%
4	23.5%	80.4%	93.1%		4	89.5%	81.8%	96.9%
5	64.1%	72.1%	98.9%		5	93.5%	84.1%	98.9%
Avg.	57.0%	68.9%	97.5%		Avg.	87.6%	68.3%	93.0%

Table 1. a) Robot position in the field. b) Robots' roles.

Table 1b) refers to the roles covered by the players during the games. Also in this case roles have been covered in an effective way, in particular the most important roles *GoToBall* and *Defend* have been executed almost at every time. With respect to a static assignment of roles, dynamic assignment still provides a good distribution of roles, but with the advantage of selecting the most appropriate robot for every role depending on the current situation of the environment.

In addition, the other statistical data have been computed:

- During the game there is in the average one role switch every 7 seconds.
- Due to occasional loss of transmitted data, we noticed that about 1/10 of the role switches generates roles' oscillation, lasting about 300 ms before stabilization.

This analysis shows the effectiveness of our distributed coordination that has significantly contributed to the overall performance of the team during the last competitions. In fact, in general the players smoothly switched role and managed to get ball possession without obstructing each other and they generally have occupied the field in a satisfactory way.

5 Conclusions

The distributed coordination method presented in this paper has been successfully employed by all the members of the ART team during the RoboCup 1999 and European RoboCup 2000 competitions [4]. The effectiveness of the method has been proved by the fact that we were always ready to substitute any robot with another one, without affecting coordinate behaviour of the overall team.

The goal of coordinating through a distributed protocol a multi agent system, formed by heterogeneous components, not only has been achieved, but actually provided a substantial contribution to the overall performance of the ART team. While the effectiveness of coordination has been addressed in the RoboCup environment, the techniques and the tools can be successfully exploited in other multi-robot domains, where similar assumptions are verified. A key step that made coordination successful has been the experimental work carried out in order to attain the desired coordinated behaviour. We have used several measures and tools for tuning and evaluating the effectiveness of the coordination protocol.

From a technical perspective the proposed protocol is based on the explicit exchange of data about the status of the environment and is based on simple

forms of negotiations. Simplicity in the protocol stems from the need to make rather weak assumptions on each robot's capabilities. An increase of such capabilities would lead to more complex protocols. However, we believe that a major issue in coordination is to find a suitable balance between the robot individual capabilities and the form of cooperation realized.

Acknowledgments

We are grateful to the members of the ART team and acknowledge the support of "Consorzio Padova Ricerche", "Consiglio Nazionale delle Ricerche", Università di Roma "La Sapienza", Politecnico di Milano, Università di Padova, Università di Genova, Università di Palermo, Università di Parma, AI*IA, Vesta pneumatics, Sony Italia, Images, Tekno.

References

1. G. Adorni, S. Cagnoni, M. Mordonini, and M. Piaggio. Team/goal-keeper coordination in the robocup mid-size league. In *RoboCup-2000: Robot Soccer World Cup IV*, 2000.
2. M. Asada. The RoboCup physical agent challenge: Goals and protocols for Phase-I. In H. Kitano, editor, *Robocup-97: Robot Soccer World Cup I*, 1998.
3. C. Castelpietra, L. Iocchi, D. Nardi, and R. Rosati. Coordination in multi-agent autonomous cognitive robotic systems. In *Proc. of 2nd International Cognitive Robotics Workshop*, 2000.
4. S. Coradeschi, L. Karlsson, P. Stone, T. Balch, G. Kraetzschmar, and M. Asada. Overview of RoboCup-99. *A.I. Magazine*. To appear.
5. M. Ferraresco, C. Ferrari, E. Pagello, R. Polesel, R. Rosati, A. Speranzon, and W. Zanette. Collaborative emergent actions between real soccer robots. In *RoboCup-2000: Robot Soccer World Cup IV*, 2000.
6. M. Hannebauer, J. Wendler, P. Gugenberger, and H. Burkhard. Emergent cooperation in a virtual soccer environment. In *DARS-98*, 1998.
7. Y. Lesperance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter, and R.B. Scherl. A logical approach to high-level robot programing. In *AAAI Fall Symposium on Control of the Physical World by Intelligent Systems*, 1994.
8. F. Montesello, A. D'Angelo, C. Ferrari, and E. Pagello. Implicit coordination in a multi-agent system using a behavior-based approach. In *DARS-98*, 1998.
9. D. Nardi, et al. ART-99: Azzurra Robot Team. In *Proceedings of 3rd RoboCup Workshop*. Springer-Verlag.
10. M. Piaggio, A. Sgorbissa, and R. Zaccaria. A programming environment for real time control of distributed multiple robotic systems. *Advanced Robotic Journal*, 14, 2000.
11. P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence* 110(2), pages 241–273, June 1999.
12. M. Veloso and P. Stone. Individual and collaborative behaviors in a team of homogeneous robotic soccer agents. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 309–316, July 1998.
13. K. Yokota et al. Utter united: Cooperative team play based on communication. In *Proceedings of 2nd Workshop on Robocup*, 1998.

A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor

Carlos F. Marques*, Pedro U. Lima

Instituto de Sistemas e Robótica, Instituto Superior Técnico

Av. Rovisco Pais, 1 — 1049-001 Lisboa, PORTUGAL

{cmarques,pal}@isr.ist.utl.pt

<http://socrob.isr.ist.utl.pt/>

Abstract. *In this paper, a method for robot self-localization based on a catadioptric omni-directional sensor is introduced. The method was designed to be applied to fully autonomous soccer robots participating in the middle-size league of RoboCup competitions. It uses natural landmarks of the soccer field, such as field lines and goals, as well as a priori knowledge of the field geometry, to determine the robot position and orientation with respect to a coordinate system whose location is known. The landmarks are processed from an image taken by an omni-directional vision system, based on a camera plus a convex mirror designed to obtain (by hardware) the ground plane bird's eye view, thus preserving field geometry in the image. Results concerning the method's accuracy are presented.*

1 Introduction and Motivation

The navigation system is perhaps the most important sub-system of a mobile robot. In many applications, especially those concerning indoors well-structured environments, one important feature of the navigation system concerns the ability of the robot to self-localize, i.e., to autonomously determine its position and orientation (posture). Once a robot knows its posture, it is capable of following a pre-planned virtual path or to stabilize its posture smoothly[1]. If the robot is part of a cooperative multi-robot team, it can also exchange the posture information with its teammates so that appropriate relational and organizational behaviors are established[9]. In robotic soccer, these are crucial issues. If a robot knows its posture, it can move towards a desired posture (e.g., facing the goal with the ball in between). It can also know its teammate postures and prepare a pass, or evaluate the game state from the team locations.

An increasing number of teams participating in RoboCup's middle-size league is approaching the self-localization problem [2]. The proposed solutions are mainly distinguished by the type of sensors used: Laser Range Finders (LRFs), vision-based omni-directional sensors and single frontal camera. The CS-Freiburg and Stuttgart-Cops teams can determine their position with an accuracy of 1 and

* This work was supported by grant PRAXIS XXI /BM /21091 /99 of the Portuguese Foundation for Science and Technology

5 cm, respectively, using LRFs. However, LRFs require walls surrounding the soccer field to acquire the field border lines and, in a sense, correlate them with the field rectangular shape to determine the team postures. Should the walls be removed, the method becomes not applicable. RoboCup's Agilo team proposes a vision-based approach to the self-localization problem too. A single frontal camera is used to match a 3-D geometric model of the field with the border lines and goals line segments in the acquired image. Only a partial field view is used in this method. Several teams use a vision-based omni-directional hardware system similar to the one described here, but only for ball and opposing robots tracking. The robots of the Tuebingen team use omni-directional vision for self-localization, but only the distance to the walls is used.

In this paper, an omni-directional catadioptric (vision + mirror) system is used to determine the robot posture, with respect to (w.r.t.) a given coordinate system, from the observation of natural environment landmarks such as straight lines resulting from the intersection between the walls and the ground, as well as from *a priori* knowledge of the environment geometry. The correlation between the observed field and its geometric model is made in Hough Transform space. Iocchi and Nardi [8] also use a single frontal camera and match the lines with a field model using the Hough Transform. However, their approach considers lines detected locally, rather than a global field view, and requires odometry to remove ambiguities.

The paper is organized as follows: in Section 2, the proposed self-localization method is described in general terms, and its application to robotic soccer is detailed in Section 3. Results concerning accuracy of the postures obtained in this case study are presented in Section 4, together with a study of its dependence on robot field locations. Finally, some conclusions and a description of envisaged future work are drawn in Section 5.

2 The Self-Localization Method

2.1 Omni-Directional Catadioptric System

Most omni-directional catadioptric systems are based on one of two types of reflecting surfaces: spherical [10] or conic[3,11]. Those systems require a perspective unwarping made by software, a time-consuming process that may prevent real-time robot guidance, in applications where fast motion is required. The solution used here, although developed independently, is similar to the one described in [7], whose catadioptric system preserves the geometry of a plane orthogonal to its symmetry axis, in particular providing a bird's eye view of the ground plane. Figure 1 shows the mirror cross-section, obtained numerically as the solution of a differential equation.

2.2 Method Description

Even though the self-localization algorithm was designed motivated by its application to robotic soccer, it can be described in general terms and applied to other

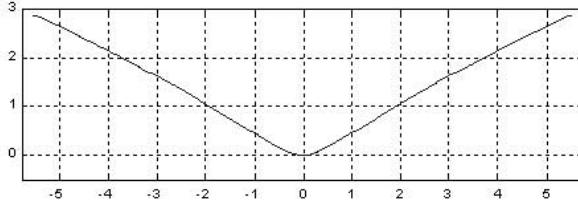


Fig. 1. Cross section of the mirror.

well-structured environments, with the assumption that the robot moves on flat surfaces and straight lines can be identified and used as descriptive features of those environments. An important requirement is that the algorithm should be robust to image noise. Given an image acquired from the catadioptric system, the basic steps of the algorithm are:

1. Build a set \mathcal{T} of *transition pixels*, corresponding to image pixel representatives of environment straight lines (e.g., intersection between corridor walls and ground, obtained by an edge detector).
2. For all transition pixels $p^t \in \mathcal{T}$, compute the Hough Transform using the normal representation of a line[6]

$$\rho = x_i^t \cdot \cos(\phi) + y_i^t \cdot \sin(\phi), \quad (1)$$

where (x_i^t, y_i^t) are the image coordinates of p^t and ρ, ϕ the line parameters.

3. Pick the q straight lines $(\rho_1, \phi_1), \dots, (\rho_q, \phi_q)$ corresponding to the top q accumulator cells resulting from the Hough transform described in the previous step.
4. For all pairs $\{(\rho_j, \phi_j), (\rho_k, \phi_k), j, k = 1, \dots, q, j \neq k\}$ made out of the q straight lines in the previous step, compute

$$\Delta\phi = |\phi_j - \phi_k| \quad (2)$$

$$\Delta\rho = |\rho_j - \rho_k|. \quad (3)$$

Note that a small $\Delta\phi$ denotes almost parallel straight lines, while $\Delta\rho$ is the distance between 2 parallel lines.

5. Classify, in the $[0, 100]$ range, the $\Delta\phi$ s and $\Delta\rho$ s determined in the previous step, for its relevance (function $\text{Rel}()$) using *a priori* knowledge of the geometric characteristics of the environment (e.g., in a building corridor of width d , only $\Delta\phi \simeq 0$, $\Delta\phi \simeq 180$ and $\Delta\rho \simeq d$ should get high grades). For each pair of straight lines, assign a grade in the $[0, 200]$ range to the pair, by adding up $\text{Rel}(\Delta\phi)$ and $\text{Rel}(\Delta\rho)$.
6. Pick up the most *relevant* pair of straight lines (i.e., the pair of largest $\text{Rel}(\Delta\phi) + \text{Rel}(\Delta\rho)$ in the previous step), and use it to extract some relevant feature regarding environment localization (e.g., the orientation θ of the robot w.r.t. the corridor walls, represented by the most relevant pair of parallel straight lines, in the example above).

7. Use the relevant feature from the previous step to proceed. For instance, assuming θ in the corridor example is such a feature, it is used to select columns from the accumulator cells matrix referred in Step 3. The idea is to correlate a number of actual straight lines, found in the image, sharing the same descriptive parameter (e.g., the angle ϕ corresponding to θ) with the expected straight lines obtained from an environment model (e.g., the building layout). To attain this, up to n ρ values from the accumulator matrix column corresponding to ϕ are picked up, corresponding to up to n straight lines found in the image. To handle uncertainty in ϕ , an even better solution is to pick up not only one column but a few columns surrounding the accumulator matrix column corresponding to ϕ , using the top n ρ values from those columns. Concatenate all these Hough space points in an array and call it $\hat{\rho}_\phi$.
8. Create an array ρ_ϕ similar to $\hat{\rho}_\phi$, but obtained from a geometric model of the environment. Actually, ρ_ϕ measures distances of environment straight lines to the origin of the world reference frame. Correlate ρ_ϕ and $\hat{\rho}_\phi$ by shifting one array over the other, and incrementing a counter for each matching $(\rho_\phi, \hat{\rho}_\phi)$ pair. The maximum of the correlation corresponds to the best match between up to n straight lines in the image and the n known environment straight lines. From this result and similar results obtained for other straight lines non-parallel to them (determined by the same procedure for different θ s), the image coordinates of environment feature points, whose location in the world reference frame is known, are determined and used to determine the robot position w.r.t. that frame, by a suitable transformation from image to world coordinates.

3 Application to Robotic Soccer

The self-localization of a middle-size league soccer robot, using the method described in the previous section, takes advantage of the soccer field geometry and of the different colors used for the field (green), the surrounding walls and the field lines (white). The field is a 9×4.5 m flat rectangle that can be almost fully observed by the robot catadioptric system from most field locations. To eliminate occlusion due to the use of mirror supports, the catadioptric system consists of an acrylic cylinder with a small color CCD camera inside and the mirror on the top. The camera is assembled in a 5 degree of freedom (3 of translation and 2 of rotation) support allowing the user to correctly position the camera with respect to the mirror. Acquired images are processed using the HSV color model[6].

Since all four robots in the team require the catadioptric system, the tradeoff between cost and final mirror machining quality had to be seriously considered. While it is true that image distortion due to poor mirror machining quality reflects in posture accuracy, the method here described was developed to be robust to considerable image distortion and to image noise due to irregular and poor illumination. As such, a non-expensive ($\simeq 150$ Euros) mirror was used. Image



Fig. 2. Catadioptric system assembled on the robot.

distortion is severe on the external mirror zone, hence processing is only made within a circle with a 220 pixel radius, centered with the camera (see Fig. 4). Figure 2 shows the catadioptric system assembled on a Nomadic SuperScout II.

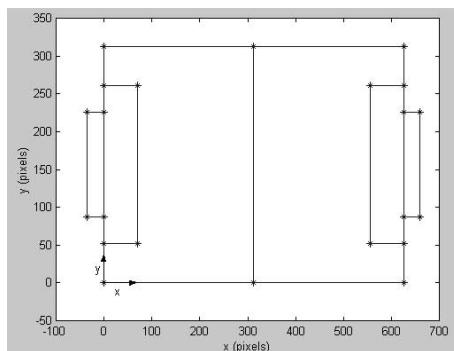


Fig. 3. Soccer field model (coordinates in pixels).

3.1 A Priori Knowledge

The bird's eye view of the soccer field, shown schematically in Fig. 3, shows 6 horizontal and 7 vertical straight lines (considering interrupted lines as only

one line). In this work, all horizontal lines and 5 of the vertical lines (excluding those corresponding to the back of the goals) were considered. Excluded lines were chosen because they are often occluded by the goalkeeper robots. All the distances between lines are known from RoboCup rules. Changes in the dimensions are parameterized in a table. The model reference frame is located at the bottom left of this field model.

3.2 Orientation Determination

Steps 1-6 of the algorithm described in Section 2 are followed to determine the initial robot orientation estimate (with a $\pm 90^\circ$ or $0^\circ/180^\circ$ uncertainty, to be solved later). The set \mathcal{T} of transition pixels is obtained by determining the white-to-green and green-to-white image transitions over 36 circles centered with the robot, shown in Fig. 4. The number of circles was determined based on a tradeoff between accuracy and CPU time.

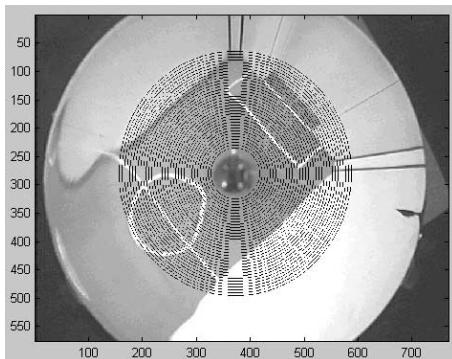


Fig. 4. Image obtained by the catadioptric system with the 36 circles used to determine transition pixels.

The Hough transform is then applied to the pixels in \mathcal{T} – a variable number from image to image, depending on the number and length of observed lines. In Step 3, $q = 6$ is used, based on experimental analysis of the tradeoff between CPU time and accuracy. The *relevance* functions for $\Delta\phi$ and $\Delta\rho$, used in Steps 5-6, are plotted in Figure 5. The latter reflects *a priori* knowledge of the environment, by its use of the known distance between relevant field lines that can be observed by the catadioptric system in one image.

The accumulator cells of the Hough transform in Step 2 are obtained by incrementing ϕ from 0 to 180° in 0.5° steps, leading to an image line slope resolution of $\tan 0.5^\circ$. ρ is incremented from 125 to 968 in steps of 1 pixel, corresponding to an actual field resolution of 6.95 mm. The $\pm 90^\circ$ or 180° ambiguity referred above results from the absence of information on which field lines lead to the most relevant pair. This information is obtained in Steps 7-8.

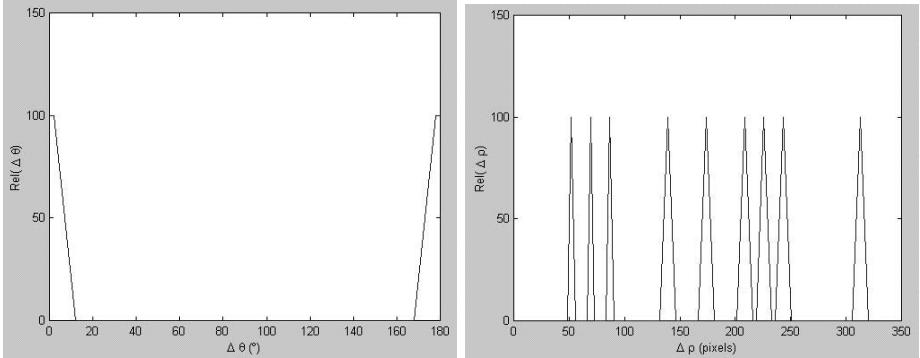


Fig. 5. Relevance functions for $\Delta\phi$ and $\Delta\rho$.

3.3 Position Determination

The final step in the self-localization process consists of determining the robot position coordinates in the soccer field. This is done together with the disambiguation of the relevant feature θ determined in Steps 1-6 of the self-localization method, by creating not only the ρ_ϕ and $\hat{\rho}_\phi$ arrays referred in Steps 7-8, but also their “orthogonal” arrays $\rho_{\phi+90}$ and $\hat{\rho}_{\phi+90}$. The correlation in Step 8 is made between all 4 possible pairs $(\rho_{\phi+90}, \hat{\rho}_{\phi+90})$, $(\rho_{\phi+90}, \hat{\rho}_\phi)$, $(\rho_\phi, \hat{\rho}_{\phi+90})$ and $(\rho_\phi, \hat{\rho}_\phi)$ with $n = 6$ (the maximum number of field lines that can be found in the image). The maximum of the 4 correlation maxima occurs for the array pair representing the best match between image and actual field lines. The array immediately identifies whether $\theta \pm 90^\circ$ or $\theta = 0^\circ \vee \theta = 180^\circ$ is the robot orientation. A companion array pair exists for each best pair. The 2 pairs uniquely identify 2 (approximately) orthogonal field lines, by checking the array positions where the maximum occurred (vertical field lines are numbered 1, ..., 5 from left to right and horizontal lines are numbered 1, ..., 6 from top to bottom). The intersection of the two lines is a reference point, whose coordinates are known in the world reference frame, from the field model.

The explanation above is summarized in the following table (the best and companion pairs positions can be exchanged):

Best Pair	Companion Pair	θ
$(\rho_\phi, \hat{\rho}_\phi)$	$(\rho_{\phi+90}, \hat{\rho}_{\phi+90})$	$\theta = \phi \pm 90^\circ$
$(\rho_\phi, \hat{\rho}_{\phi+90})$	$(\rho_{\phi+90}, \hat{\rho}_\phi)$	$\theta = \phi \vee \phi + 180^\circ$

The robot position is computed from a rotation of θ (one of the possible values is used, with no special criterion), followed by a translation that expresses the center of the image (i.e., the robot position in image coordinates) in the model reference frame, and another translation plus a scale factor f to express it in world coordinates. The world reference frame is located in the middle of the soccer field, with the x axis pointing towards the blue goal and the y axis is such

that a 3-D coordinate frame would have z pointing upwards. The orientation θ is measured from x to a pre-defined straight line passing through the robot center. The scale factor f depends on the geometry of the catadioptric system and can be calibrated experimentally. This transformation can be expressed by the following equation, using homogeneous coordinates:

$$\begin{bmatrix} x_f^r \\ y_f^r \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & x_i^{\text{ref}} + x_m^{\text{ref}} \\ -\sin \theta \cos \theta & y_i^{\text{ref}} + y_m^{\text{ref}} & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i^r \\ y_i^r \\ 1 \end{bmatrix} - \begin{bmatrix} 450 \\ 225 \\ 0 \end{bmatrix} \cdot f \quad (4)$$

where the subscripts i, m, f stand for the image, field model and actual field reference frames, and the superscripts ref and r stand for the reference point and the robot, respectively.

A further validation and disambiguation of the robot posture is required, since, when only two parallel lines are used to determine the position, and due to field symmetry, the robot side of the field is unknown, as well as its orientation. To solve this problem, two tests are made. First, the algorithm checks whether the robot position is not outside the field. The second test consists of using the current estimated posture to seek the nearest goal in the image.

This is achieved by selecting m points located inside one of the goals (blue or yellow) in the actual field and applying to each of those points of coordinates (x_f^g, y_f^g) the inverse transform of (4):

$$\begin{bmatrix} x_i^g \\ y_i^g \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & x_i^{\text{ref}} + x_m^{\text{ref}} \\ -\sin \theta \cos \theta & y_i^{\text{ref}} + y_m^{\text{ref}} & 1 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \left(\begin{bmatrix} x_f^g \\ y_f^g \\ 1 \end{bmatrix} + \begin{bmatrix} 450 \\ 225 \\ 0 \end{bmatrix} \right) \cdot f \quad (5)$$

where the superscript g stands for goal.

Should the majority of the corresponding pixels in the image have the same color of the field pixels, $\theta = 0^\circ$ and the estimated position is validated. Should they have the color of the opposing goal, $\theta = 180^\circ$ and the symmetrical coordinates of the current position estimate must be used for the robot position. When the majority of image pixels is green, the top maximum of the correlation process is removed and the whole process re-started using the second maximum, and if needed, the third one and so on until the actual posture is determined.

4 Experimental Results

The described self-localization algorithm has been implemented in C and used to self-localize a robot. The method was applied to a set of 90 images obtained by a catadioptric system mounted on a Super Scout II robot. The images were taken at different field spots, with several images taken at each spot, and were processed in about 0.5 second each, in a Pentium 233MHz with 64Mb of RAM, the Super Scout II on board computer. Table 1 shows the results of the 90

experiments. The first column gives the average accuracy μ in x , y and θ , the second the variance of the accuracy σ^2 and the third column the accuracy for one standard deviation. In Fig. 6, the histogram of the accuracy, for the x and y coordinates, is shown as well as an adjusted Gaussian function. The represented rectangle contains all the accuracies within one standard deviation from μ , i.e., 68,2% of the postures obtained have an accuracy of, e.g., 10 cm in X .

The accuracy was determined as the difference between the estimated values and the ones measured on the field, using pre-defined spots whose location is well known (e.g., the corner of the goal area). The precision (i.e., the difference between the measured value and the measurements average value for the same location) results are similar, and visual inspection made the average values seem trustable.

	μ	σ^2	$\mu - \sigma$
x	+3.2 (mm)	0.0099 (m^2)	10 (cm)
y	-18.0 (mm)	0.0084 (m^2)	9.18 (cm)
θ	0.22 °	3.14 ° ²	1.77 °

Table 1. Posture accuracy statistics (mean and standard deviation).

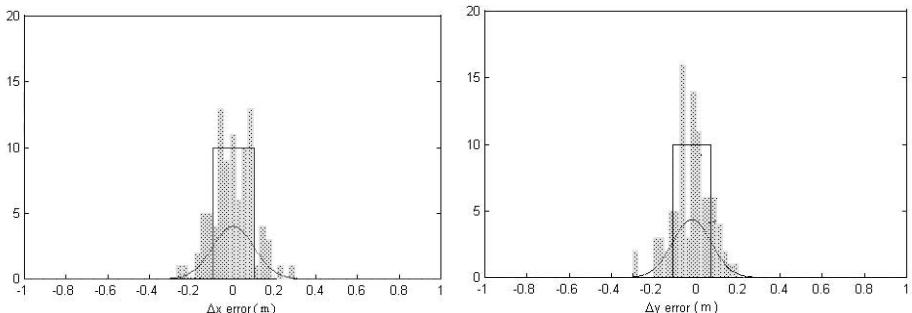


Fig. 6. Position error histogram.

Figure 7 shows an example of an image to be processed. The lines represented are the possible lines of the field. In this case, the $(\rho_\phi, \hat{\rho}_{\phi+90})$ pair achieved the top correlation value and position with an error of $\Delta x = +1$ cm, $\Delta y = +1$ cm and $\Delta \theta = +1$ °. Note that, in this test, the robot is close to one of the field walls, making harder the posture determination process, as, due to the limited image used, the other wall is not seen, and a relevant parallel line can not be found by the algorithm.

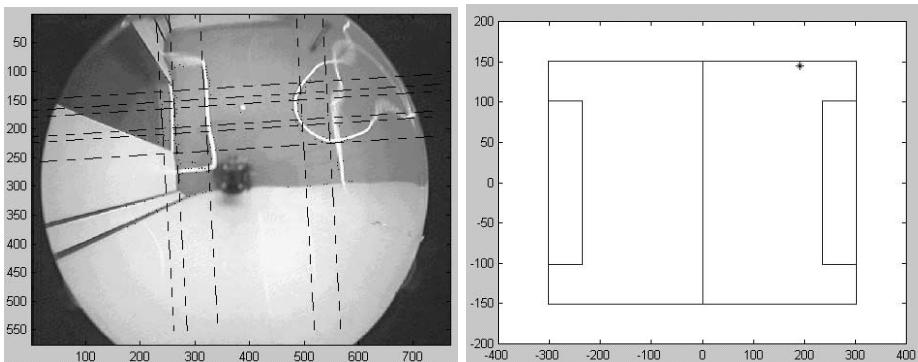


Fig. 7. Test image results.

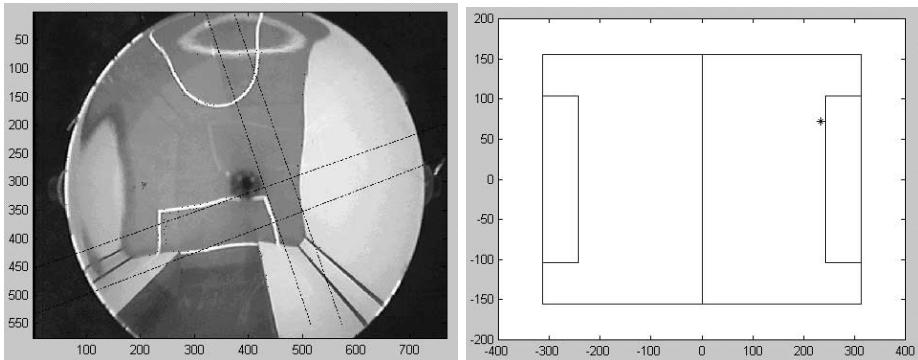


Fig. 8. Bad test image results.

One example of a bad image, is shown in Fig. 8. In this case, the position was computed with an error of $\Delta x = +10$ cm, $\Delta y = +1$ cm and $\Delta\theta = +21^\circ$. Even though the results shown in Fig. 8 are considerably worse, they are acceptable for the due purposes (except the orientation), considering the large image distortion.

5 Conclusions and Future Work

A vision-based algorithm for robot self-localization was tested on images taken from a low-cost catadiotric system mounted on a Super Scout II soccer robot. The algorithm was designed for well structured environments, where *a priori* knowledge about the world model is available, and straight lines can be used to describe environment features.

In the robotic soccer application, promising results were obtained concerning posture accuracy and method robustness to image noise and distortion. The

method robustness meets the problem specifications, as the robot projection on the field is roughly a 40 cm radius circle and the ball diameter is 21 cm, and typical position errors ranged from 0 to 10 cm.

One way to further reduce the position error range is to use sensor fusion methodologies. The Tuebingen team fuses three different self-localization methods for the localization of their goalkeeper, using three sensors: an omnidirectional camera, a Laser Range Finder and a compass [2]. The method can be used only in regions near the field goals. However, we rely only on an omnidirectional vision system for self-localization which can be used everywhere in the field.

Recently, a new version of the norm-preserving mirror has been machined, with significantly better results, both concerning the reduction in distortion and the visible field range. The algorithm here described is currently being used by the ISocRob team to periodically reset odometry and keep the whole team knowledgeable of the teammate postures.

The integrated odometry + vision-based self-localization system is also being used as feedback for guidance control of the ISocRob team soccer robots. The guidance controller makes the robot acquire a desired posture (e.g., facing the goal with the ball in between) while using the SuperScout II sonars to avoid the other robots and the walls during its motion.

Information on all teammate postures is shared through a distributed blackboard where other important environment features are also stored, such as the ball position, as seen by each robot. We are currently working towards using Markov Localization methods [4,5] to refine each robot posture estimates by taking advantage of the global field view obtained by the catadioptric system and the features used in the current work, such as the field lines. This will also lead to a cooperative localization of relevant objects, such as the ball and opponent robots, with an estimate of the involved uncertainty.

Acknowledgements

The authors would like to thank Luis Custodio, José Santos-Victor and Rodrigo Ventura for the fruitful discussions about the subject of this paper.

References

1. C. Canudas de Wit, B. Siciliano, G. Bastin (Eds), *Theory of Robot Control*, CCE Series, Kluwer, 1996
2. S. Coradeschi, T. Balch, G. Kraetzschmar, P. Stone (Eds), *ROBOCUP-99, Small and Middle Leagues, Team Descriptions*, Stockholm, Sweden, 1999
3. L. Delahoche, C. Pégard, B. Marhic, P. Vasseur, "A Navigation System Based on an Omni-directional Vision Sensor", in *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 718-724, 1997
4. S. Enderle, M. Ritter, D. Fox, S. Sablatnog, G. Kraetzschmar, G. Palm, "Vision-Based Localization in RoboCup Environments", in *this book*, 2001

5. D. Fox, W. Burgard, S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments", *Journal of Artificial Intelligence Research*, 11, pp. 391-427, 1999
6. R. Gonzalez, R. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
7. R. A. Hicks, R. Bajcsy, "Reflective Surfaces as Computational Sensors", *in Proc. of CVPR'99, Workshop on Perception for Mobile Agents*, 1999
8. L. Iocchi, D. Nardi, "Self-Localization in the RoboCup Environment", *in Proc. of Sixteenth IJCAI 99, The III Int. Workshop on RoboCup*, pp 116-120, 1999.
9. P. Lima, R. Ventura, P. Aparício, L. Custódio, "A Functional Architecture for a Team of Fully Autonomous Cooperative Robots", *in RoboCup-99: Robot Soccer World Cup III*, Springer-Verlag, Berlin, 2000
10. B. Marhic, E. M. Mouaddib, C. Pegard, "A Localisation Method with an Omni-directional Vision Sensor Using Projective Invariant", *in Proc. of IEEE Int. Conf. on Intelligent Robots and Systems*, pp. 1078-1083, 1998
11. Y. Yagi, M. Yachida. "Real-time Generation of Environmental Map and Obstacle Avoidence using Omni-directional Image Sensor with Conic Mirror", *Trans. IEEE Robotics and Automation*, Vol. 10 No.1, pp. 11-22, 2/28 ,1991

Behavior Classification with Self-Organizing Maps

Michael Wünstel¹, Daniel Polani^{2,1*}, Thomas Uthmann¹, and Jürgen Perl¹

¹ Institut für Informatik, Johannes Gutenberg-Universität Mainz, Germany

² Institut für Neuro- und Bioinformatik, Universität Lübeck, Germany

{wuen, polani, uthmann, perl}@Informatik.Uni-Mainz.DE

Abstract. We describe a method that applies Self-Organizing Maps for direct clustering of spatio-temporal data. We use the method to evaluate the behavior of RoboCup players. By training the Self-Organizing Map with player data we have the possibility to identify various clusters representing typical agent behavior patterns. Thus we can draw certain conclusions about their tactical behavior, using purely motion data, i.e. logfile information. In addition, we examine the player-ball interaction that give information about the players' technical capabilities.

1 Introduction

The goal of our work is to present a method to detect characteristic features of trajectories. The method is illustrated by analyzing the actions of Robocup players on a purely behavioral level. Only logfile information and no knowledge about implementation or inner states of the players is used.

In our analysis we use *Kohonen's Self-Organizing Map*¹ (Kohonen, 1989). The SOM is a data analysis method inspired by the structure of certain cortex types in the brain. It is able to identify different clusters in high-dimensional data and to project ("map") the data of the different clusters to a two-dimensional grid respecting the topology, i.e. the neighborhood structure of the data. By this a visualization of the data can be easily obtained. The mapping and the visualization capability is an advantage of the SOM as compared to standard statistical methods; in particular, the SOM does not just separate different clusters, but it also resolves the inner structure of the clusters. Mathematically it is related to the principal surface models for data distributions known from statistics (Ritter et al., 1992), though, unlike the SOM, the latter are not designed to handle data sets decomposing into different clusters.

For a trained SOM, each data point from the high-dimensional space is projected onto an element of the two-dimensional SOM grid, a *SOM unit*. Nearby data points are projected to the same unit or a unit close by in the grid. In turn, every SOM unit represents a vector in the high-dimensional data space, such that the SOM can be regarded as an embedding of the two-dimensional SOM

* corresponding author

¹ in the following abbreviated by *SOM*

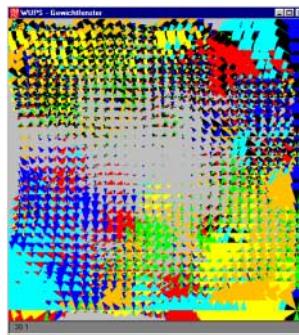


Fig. 1. SOM of 30×30 units trained with SFR-vectors in sector representation (see Secs. 2 and 2.2 for the definition of SFR-vectors and Sec. 3.1 for a description of the sector representation)

grid into the high-dimensional data space. Fig. 1 gives a general impression of the form of such an embedding for RoboCup player trajectories.

In this paper, each SOM unit will either represent a trajectory slice (i.e. a short section of a trajectory) of a single player or a trajectory slice of the combined motion of player and ball. Before the SOM can do that, it first has to be trained in order to learn to represent a set of given motion slices in an optimal way. This way we obtain a two-dimensional map of units, each unit encoding a trajectory slice. The trained map can be employed to classify the individual trajectory slices or to examine the behavior of a single player in general. One property of the trained SOM is especially valuable: neighboring units describe similar paths of motion. With the help of a graphical display it is then possible to get an overview over the behavior patterns of a single player. The paper will give an introduction into the SOM algorithm, introduce the representation for player trajectories and present its application to the analysis of player behavior in a game from the RoboCup 1999 competition at Stockholm.

2 Self-Organizing Maps and Trajectory Clustering

The problem of storing trajectories with the help of SOMs can be solved in different ways. One method is to examine the trace of the winning units like the trace of a elementary particle in a cloud chamber. One would then compare the order of activated units with some reference order. This method has been used successfully for example in (Carpinteiro, 1999) to recognize instances of a theme in a piece of music of J.S. Bach. It also can be used in speech recognition to trace the order of phonemes (Mehler, 1994; Kangas, 1994). Another way has been used by Chappell and Taylor (Chappell and Taylor, 1993) who used Leaky Integrator Units. These units can store the units' activations for a while and therefore are able to represent temporal information. The method presented in our paper,

however, directly stores a complete trajectory slice itself in each SOM unit. Here we use the *spatially focused representation* (*SFR*), where we consider a trajectory as a simple sequence of spatial data: The time intervals are fixed and discrete and the spatial data are continuous. With the so-called *enhanced spatially focused representation* (*ESFR*) we were able to combine two simultaneous trajectories (of one player and the ball) and thus examine the player-ball interactions. The data representation is similar to SFR. We use these methods to examine the short-term (micro-)behavior of individual RoboCup players.

2.1 The Basic SOM Model

A Self-Organizing Map with m inputs is a set \mathcal{N} of units, where every unit i is assigned a *weight vector* $\mathbf{w}_i \in \mathbb{R}^m$ (Kohonen, 1989; Polani and Uthmann, 1992; Ritter et al., 1992). Furthermore on \mathcal{N} there exists a metric $d_{\mathcal{N}}$ which defines a distance $d_{\mathcal{N}}(i, j)$ for each two units in \mathcal{N} . An *input (training) signal* is a vector $\mathbf{x} \in \mathbb{R}^m$. The similarity between an input signal \mathbf{x} and a unit's weight vector \mathbf{w}_i is given by their Euclidean distance $\|\cdot\|$. An input signal \mathbf{x} is said to *activate* a (typically unique) unit i^* , for which

$$\|\mathbf{w}_{i^*} - \mathbf{x}\| \leq \|\mathbf{w}_i - \mathbf{x}\|, \forall i \in \mathcal{N}$$

holds, i.e. that unit whose weight vector has the smallest Euclidean distance to the input signal. The training of the map involves three steps:

1. **Initialization:** the values of the weights \mathbf{w}_{il} , $l = 1, \dots, m$ of every unit i can be randomly chosen.
2. **Stimulus and Response:** choose an input vector out of a training set according to some probability distribution and detect the activated (*winning*) unit i^* .
3. **Adaption:** modify the weights \mathbf{w}_i of all the units in a certain \mathcal{N} -neighborhood of the winning unit i^* towards the winning unit i^* . The degree of the \mathcal{N} -neighborhood is determined by the distance function (metric) $d_{\mathcal{N}}$.

Steps 2 and 3 are repeated arbitrarily often. Formally, the learning rule for the adaption of the weights of a unit i at iteration t is given by:

$$\Delta \mathbf{w}_i(t) := \epsilon(t) \cdot h_t(i^*, j) (\mathbf{x}(t) - \mathbf{w}_i(t)) \quad (1)$$

$\mathbf{x}(t)$	the training input at iteration t
$i^* = i^*(\mathbf{x}(t))$	the winning unit at iteration t
$\epsilon(t)$	the learning rate at iteration t . The learning rate is a monotonously decreasing function with $0 < \epsilon(t) < 1$.
h_t	the activation profile at iteration t . Unit weights close to the winning unit i^* are more attracted to the winning unit i^* . An example for h is found in Eq. (2) (Sec. 3.1).

The weights of units are then changed according to $\mathbf{w}_i(t+1) := \mathbf{w}_i(t) + \Delta \mathbf{w}_i(t)$. If the units in \mathcal{N} are now arranged as a two-dimensional lattice, above

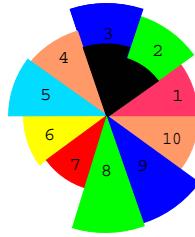


Fig. 2. Weight vector in sector representation

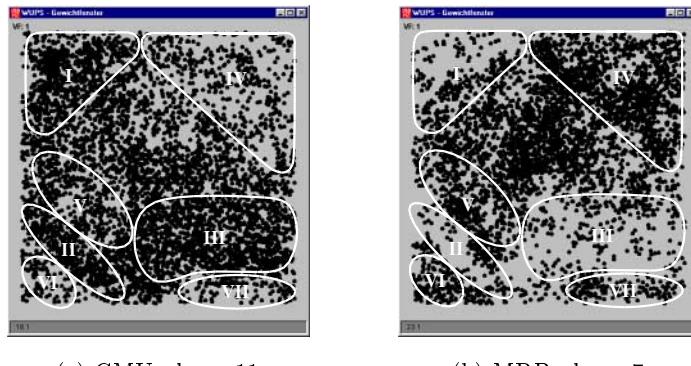


Fig. 3. SOM activations for SFR-vectors

algorithm generates a projection of the high-dimensional input space to the two-dimensional lattice. As neighboring units of a trained SOM represent similar weights, it is possible to identify clusters representing similar data. In Fig. 1 the similarity has been made visible by a direct plot of the units' weights². It can be seen that units with similar weights form certain areas (Kohonen, 1989). If the trained SOM is then fed with data, the frequency of the units' activations can help to draw conclusions about the examined data set.

In Figs. 3(a) and 3(b) a dot is plotted for every activation of a unit. Since the same unit may be activated multiple times, the dot positions are slightly perturbed for each activation event to obtain a visualization of the activation frequencies ("importance") of a certain region in the unit lattice.

2.2 Spatially Focused Representation (SFR)

The training input (a trajectory) is given as a sequence of τ difference ("velocity") vectors \mathbf{u} . Every difference vector is calculated from two successive position

² Sector representation is used as shown in Fig. 2 and described in detail in Sec. 3.1.

vectors \mathbf{p} of a RoboCup player:

$$\begin{aligned}\mathbf{p} &= (\mathbf{p}_t)_{t=t_0-t', t'=\tau, \dots, 0}, \quad \mathbf{p} \in \mathbb{R}^2 \\ \mathbf{u} &= (\mathbf{u}_t)_{t=t_0-t', t'=\tau-1, \dots, 0} \\ \mathbf{u}_t &= \mathbf{p}_t - \mathbf{p}_{t-1}\end{aligned}$$

In our experiments we use the sequence length $\tau = 5$. A training input $\mathbf{x}(t)$ at simulation time step $t = t_0$ is then given by

$$\begin{aligned}\mathbf{x}(t = t_0) &= (\mathbf{u}_{t_0-4}, \mathbf{u}_{t_0-3}, \mathbf{u}_{t_0-2}, \mathbf{u}_{t_0-1}, \mathbf{u}_{t_0-0})^T \\ &\quad (\mathbf{p}_{t_0-4} - \mathbf{p}_{t_0-5}, \\ &\quad \mathbf{p}_{t_0-3} - \mathbf{p}_{t_0-4}, \\ &= \quad \mathbf{p}_{t_0-2} - \mathbf{p}_{t_0-3}, \\ &\quad \mathbf{p}_{t_0-1} - \mathbf{p}_{t_0-2}, \\ &\quad \mathbf{p}_{t_0-0} - \mathbf{p}_{t_0-1})^T\end{aligned}$$

We call such a vector a *SFR-vector*. This method has already been introduced in (Boll, 1999; Wünstel et al., 1999).

2.3 Enhanced Spatially Focused Representation (ESFR)

Whereas in the SFR a training input vector is built exclusively by the difference vectors \mathbf{u} of a RoboCup player, here ball-ball and player-ball difference vectors are used. Thus the trajectory slices of the player and the ball are completely described save a possible translation about the origin. The starting point of a trajectory is the player-ball difference vector at time step t followed by pairs of ball-ball and player-ball difference vectors of the next time steps:

$$\begin{aligned}\mathbf{p}_{\text{ball}} &= (\mathbf{p}_{\text{ball}_t})_{t=t_0-t', t'=\tau-1, \dots, 0}, \quad \mathbf{p}_{\text{ball}} \in \mathbb{R}^2 \\ \mathbf{p}_{\text{player}} &= (\mathbf{p}_{\text{player}_t})_{t=t_0-t', t'=\tau-1, \dots, 0}, \quad \mathbf{p}_{\text{player}} \in \mathbb{R}^2 \\ \mathbf{u}_{\text{ball}} &= (\mathbf{u}_{\text{ball}_t})_{t=t_0-t', t'=\tau-2, \dots, 0} \\ \mathbf{u}_{\text{ball}_t} &= \mathbf{p}_{\text{ball}_t} - \mathbf{p}_{\text{ball}_{t-1}} \\ \mathbf{u}_{\text{player}} &= (\mathbf{u}_{\text{player}_t})_{t=t_0-t', t'=\tau-1, \dots, 0} \\ \mathbf{u}_{\text{player}_t} &= \mathbf{p}_{\text{player}_t} - \mathbf{p}_{\text{ball}_t}\end{aligned}$$

The training input $\mathbf{x}(t)$ at time step $t = t_0$ is given by³

$$\mathbf{x}(t = t_0) = \begin{pmatrix} \mathbf{u}_{\text{player}_{t_0-\tau+1}} \\ \mathbf{u}_{\text{ball}_{t_0-\tau+2}} \\ \mathbf{u}_{\text{player}_{t_0-\tau+2}} \\ \vdots \\ \mathbf{u}_{\text{ball}_{t_0-1}} \\ \mathbf{u}_{\text{player}_{t_0-1}} \\ \mathbf{u}_{\text{ball}_{t_0}} \\ \mathbf{u}_{\text{player}_{t_0}} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{\text{player}_{t_0-\tau+1}} - \mathbf{p}_{\text{ball}_{t_0-\tau+1}} \\ \mathbf{p}_{\text{ball}_{t_0-\tau+2}} - \mathbf{p}_{\text{ball}_{t_0-\tau+1}} \\ \mathbf{p}_{\text{player}_{t_0-\tau+2}} - \mathbf{p}_{\text{ball}_{t_0-\tau+2}} \\ \vdots \\ \mathbf{p}_{\text{ball}_{t_0-1}} - \mathbf{p}_{\text{ball}_{t_0-2}} \\ \mathbf{p}_{\text{player}_{t_0-1}} - \mathbf{p}_{\text{ball}_{t_0-1}} \\ \mathbf{p}_{\text{ball}_{t_0}} - \mathbf{p}_{\text{ball}_{t_0-1}} \\ \mathbf{p}_{\text{player}_{t_0}} - \mathbf{p}_{\text{ball}_{t_0}} \end{pmatrix}$$

³ Note that there is always one more player-ball entry than ball-ball entry.

We call such a vector an *ESFR-vector*.

Example 1. With $T = 49$, $\tau = 4$ we have:

$$\begin{aligned}
 \mathbf{p}_{\text{player}_{46}} &= (18, 5)^T & \mathbf{p}_{\text{ball}_{46}} &= (22, 7)^T \\
 \mathbf{p}_{\text{player}_{47}} &= (25, 7)^T & \mathbf{p}_{\text{ball}_{47}} &= (34, 11)^T \\
 \mathbf{p}_{\text{player}_{48}} &= (37, 7)^T & \mathbf{p}_{\text{ball}_{48}} &= (41, 12)^T \\
 \mathbf{p}_{\text{player}_{49}} &= (40, 7)^T & \mathbf{p}_{\text{ball}_{49}} &= (46, 11)^T \\
 \\
 \mathbf{u}_{\text{player}_{46}} &= (-4, -2)^T & \mathbf{u}_{\text{ball}_{47}} &= (12, 4)^T \\
 \mathbf{u}_{\text{player}_{47}} &= (-9, -4)^T & \mathbf{u}_{\text{ball}_{48}} &= (7, 1)^T \\
 \mathbf{u}_{\text{player}_{48}} &= (-4, -5)^T & \mathbf{u}_{\text{ball}_{49}} &= (5, -1)^T \\
 \mathbf{u}_{\text{player}_{49}} &= (-6, -4)^T & &
 \end{aligned}$$

$$\mathbf{x}(t=49) = (-4, -2, 12, 4, -9, -4, 7, 1, -4, -5, 5, -1, -6, -4)^T$$

3 Experiments and Results

In our experiments we test the SFR method with the isolated data of two opposing RoboCup players. We choose two offensive players with approximately equivalent roles from the game Carnegie Mellon United (CMU) versus Mainz Rolling Brains (MRB) at the RoboCup WorldCup 1999 in Stockholm (simulation league): CMU left wing forward player 11 and MRB left wing forward player 7. For the experiment with the ESFR method we use ESFR-vectors of all players of the same game. Only those ESFR-vectors were used whose player-ball distance vector value at the beginning of the trajectory is smaller than 30 simulator units, since we are only interested in player behavior related to the ball.

In this procedure, the SOM solves the task of organizing the data in a way suitable to projection to 2-dimensional space. This representation has then to be “interpreted”, that is, to be translated into a human-readable notion. The units belonging to the different clusters have to be identified; this step can be done in a semi-automated fashion based on distances of unit weight vectors (Boll, 1999), which can not be described here due to space limitations. The different clusters were then tagged by human inspection. This processing step was not automated since that would have required a linguistic concept for the treatment of geometric notions outside the scope of our work.

Here it is important to emphasize the following point: in principle all possible movement classes could have been enumerated and the vectors falling into those classes according to some suitable criterion could have been counted instead of using the SOM. However, the SOM is able to detect clusters that do not belong to a clear-cut linguistic category, and, in addition, it does not introduce unnecessary classes. Above that, for every class, the SOM maps the data variability to areas on the grid instead of just breaking down the data to a number of events of a certain class.

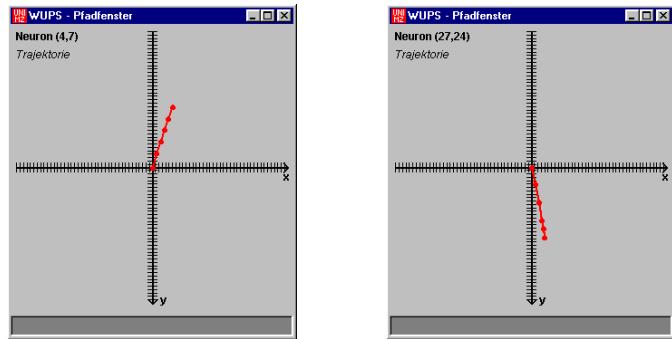


Fig. 4. Player trajectories reconstructed from a weight vector

3.1 Spatially Focused Representation

The SOM consists of a lattice of 30×30 units. The distance between two units i, j is given by their Euclidean distance, i.e.

$$d_N(i, j) = \| \mathbf{i} - \mathbf{j} \|_2 ,$$

where \mathbf{i}, \mathbf{j} are the position vectors of unit i and j in the lattice⁴.

The activation profile h_t (see Eq. (1)) is a cone function, i.e.

$$h_t(i, j) = \begin{cases} 1 - \frac{\| \mathbf{i} - \mathbf{j} \|_2}{r_h(t)} & \text{if } \| \mathbf{i} - \mathbf{j} \|_2 < r_h(t) \\ 0 & \text{else} \end{cases} . \quad (2)$$

The activation radius r_h is given by: $r_h(t) = 40(1 - 0.0006)^t$, with the constraint that $r_h(t) \geq 1.7$.

The learning rate $\epsilon(t)$ is given by $\epsilon(t) = 0.65(1 - 0.001)^t$, with the constraint that $\epsilon(t) \geq 0.15$. After the SOM is trained with the SFR-vectors of both players, the weight vector of every unit represents one SFR-vector which describes a trajectory slice. We use the sequence length $\tau = 5$. Shorter sequences do not provide enough structure to distinguish interesting separate behaviors, and in longer sequences several micro-behaviors tend to be combined into sequences that lead to unnecessary multiplication of classes.

In Fig. 1 the weight vectors of the units after the training are plotted as groups of sectors. Every weight vector is represented by 10 sectors (see Fig. 2), one sector for every component of the unit's weight vector. The radius of each sector represents the value of that component, negative values are marked by a black component, showing the self-organization of the SOM clusters.

⁴ Note that the metric here is taken on the two-dimensional grid space which is distinct from that of the vectors \mathbf{w} from Sec. 2.1.

In Fig. 4 the trajectory slices for two different example weight vectors are shown. The beginning of a trajectory is placed in the origin. A trajectory is hence the translation of a part of the corresponding movement of a RoboCup player to the origin. The dots represent the player position at successive simulator time steps.

In the next step the different behavior of the players will be examined. Therefore the SOM gets activated with all the SFR-vectors of one of the players. Every activation of a unit is plotted as a dot. It is slightly perturbed for easier identification of multiple unit activations. This procedure is repeated for the other player. The Figs. 3(a) and 3(b) show the resulting SOM activations. After the SOM clusters and maps the input data, the tagging of the clusters (as translation into human-understandable notions) has to be done by a human.

If one compares the activation zones with trajectories like in Fig. 4 one can draw some player specific conclusions: a large part of the forward movements (zones II, V, VI) of the CMU player are of intermediate speed (zone II). His sideways movements (zones I & III) are also carried out mainly at intermediate speed. The back movements hardly took place. This can be explained with the superiority of CMU.

The forward movements of MRB player are carried out slowly (zone V) or rapidly (zone VI). Sideways movements are rare (only few activations in the zones I & III). A large part of the right sideways movements are fast (zone VII). Back movements here are frequently happening (zone IV). This can either be explained again with the superiority of CMU or with a reduced action radius (as the MRB team plays with 5 offensive players in a row) which let him go back to a waiting position. While the SOM extracted all above information from purely behavioral (i.e. logfile) data, inspection of the agent code confirms that many of the found behavior patterns are indeed implemented in the agent. Our trajectory analysis therefore provides a possibility to reconstruct behavior patterns without knowing implementation details. In addition, it provides a possibility to detect behavior patterns not explicitly implemented but emerging from the collective agent dynamics.

3.2 Enhanced Spatially Focused Representation

The SOM again consists of a lattice of 30×30 units. The distance between two units i, j is given by their Euclidean distance in the lattice. The activation profile is a cone function (Eq. (2)) where the activation radius r_h is given by $r_h(t) = 40(1 - 0.00005)^t$, with the constraint that $r_h(t) \geq 1.7$. The learning rate is given by $\epsilon(t) = 0.35(1 - 0.00005)^t$, with the constraint that $\epsilon(t) \geq 0.07$. The SOM is trained for 100,000 iterations with the ESFR-vectors from the same RoboCup match as before. In Fig. 5 the weight vectors of the units after the training are plotted again as groups of sectors. Every weight vector is represented by 14 sectors, one sector for every component of the unit's weight vector. The radius of each sector again represents the value of that component. In Fig. 6 two example trajectories of two weight vectors are presented. The sequence length is $\tau = 4$.

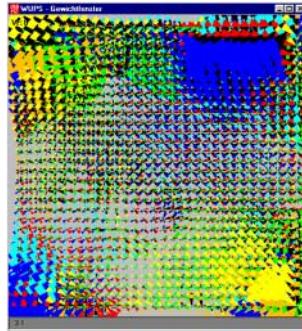


Fig. 5. SOM of 30×30 Units trained with ESFR-vectors

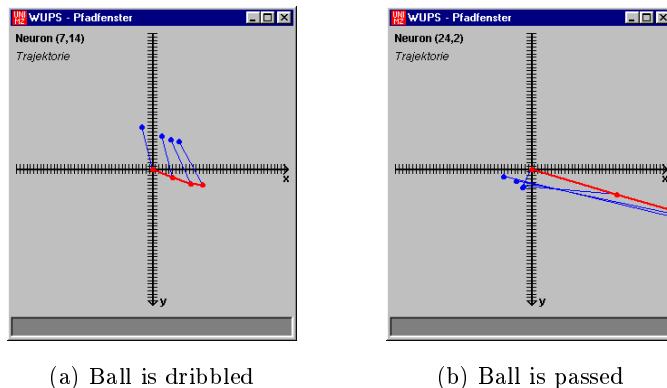


Fig. 6. Player-ball trajectories reconstructed from a SOM weight vector. Note that the ball and not the player is initially placed in the origin. The player is denoted by dark grey, the ball by light grey dots. Player and ball positions corresponding to the same simulator time step are connected by a line.

To examine the technical capabilities of the teams, the trained SOM is activated with the activation vectors of the teams. Every activation of a unit is again plotted as a dot. Figs. 7(a) and 7(b) show the activation distributions. Inspection of Fig. 5 and of player-ball trajectories like in Fig. 6 allows again to draw some conclusions about the players' capabilities:

The trajectory analysis shows very explicitly that the CMU team has much more ball contacts than the MRB team. We distinguish between the techniques dribbling, passes and "near-ball" game. Dribbling trajectories can be identified whenever player and ball are in phase and the distance between ball and player is only few units. In pass trajectories the player only slightly changes his position and the ball leaves the player rapidly. Near-ball game consists of trajectories

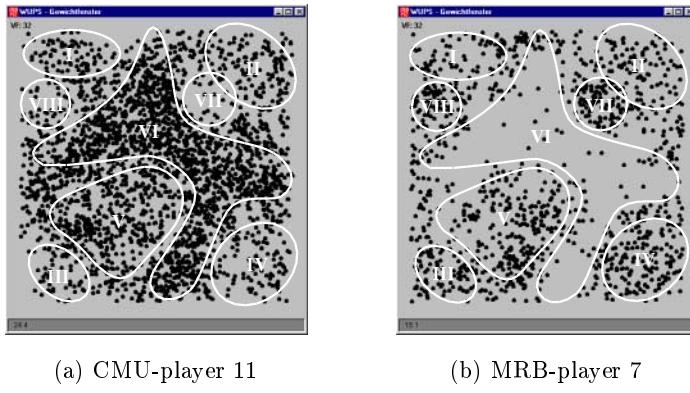


Fig. 7. SOM activations for ESFR-vectors

where the player changes his position for only few units and the ball remains close to the player.

If we examine the dribbling zones (zone VI, VII & VIII) we recognize that CMU controls dribbling in a extraordinary way (zone VI). In contrast to that MRB dribbling occurs seldom and is of lower quality (zones VII and VIII). The activation frequency for the two players differs in zones representing passes (pass right: zone I, pass forward: zone II, pass back: zone III & pass back: zone IV) are examined. The near-ball game (zone V) of CMU is more evenly distributed, at MRB it is more focused on certain patterns.

4 Conclusion and Outlook

The paper introduced and examined the use of SOMs for the examination of the micro-behavior and motion of individual RoboCup players. As opposed to other SOM trajectory analysis methods, in our method each SOM unit encodes not just single states of a trajectory slice, but a complete trajectory slice instead.

Requiring only behavioral (i.e. logfile) data, the method proved very useful to identify short-term behavior patterns explicitly implemented in the agents or emerging during the game. The method shown is easily generalizable to tasks outside of RoboCup or collective agent systems. Its potential covers a wide selection of applications, wherever time-dependent patterns have to be analyzed. In the future, we plan to extend the approach to the analysis of more complex RoboCup scenarios and to other collective agent systems. Further we wish to investigate the potential of the method to uncover causal dependencies between behavior patterns.

4.1 Acknowledgements

We would like to express our gratitude to the anonymous reviewers for their very detailed and helpful comments.

References

- Boll, M., (1999). *Analyse von Verhaltensprozessen mit Hilfe Neuronaler Netze*. Master's thesis, Johannes Gutenberg-Universität Mainz.
- Carpinteiro, O. A. S., (1999). A hierarchical self-organizing map model for sequence recognition. *Neural Processing Letters*, 9:1–12.
- Chappell, G. J., and Taylor, J. G., (1993). The Temporal Kohonen Map. *Neural Networks*, 6:441–445.
- Kangas, J., (1994). *On the Analysis of Pattern Sequences by Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Espoo, Finland.
- Kohonen, T., (1989). *Self-Organization and Associative Memory*, vol. 8 of *Springer Series in Information Sciences*. Berlin, Heidelberg, New York: Springer-Verlag. Third edition.
- Mehler, F., (1994). *Selbstorganisierende Karten in Spracherkennungssystemen*. Dissertation, Institut für Informatik, Johannes Gutenberg-Universität Mainz.
- Polani, D., and Uthmann, T., (1992). Neuronale Netze – Grundlagen und ausgewählte Aspekte der Theorie. Technical Report 2/92, Institut für Informatik, Universität Mainz.
- Ritter, H., Martinetz, T., and Schulten, K., (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Reading, MA: Addison-Wesley.
- Wünstel, M., Boll, M., Polani, D., Uthmann, T., and Perl, J., (1999). Trajectory Clustering using Self-Organizing Maps. In Sablatnög, S., and Enderle, S., editors, *Workshop RoboCup at KI'99 in Bonn, Germany*, Report 1999/12 ISSN: 1438-2237, 41–46. SFB 527 Ulm University.

Flexible Synchronisation within RoboCup Environment: a Comparative Analysis

Marc Butler, Mikhail Prokopenko, Thomas Howard

Artificial Intelligence in e-Business Group
CSIRO Mathematical and Information Sciences
Locked Bag 17, North Ryde, NSW 1670, Australia

Abstract. Synchronisation between an agent and the environment it resides in, is without a doubt, an important aspect of a more generic problem of agent interaction with the environment. A systematic comparative analysis of alternative approaches to the synchronisation problem remains an open challenge, despite numerous successful implementations of RoboCup teams in the past. The underlying reasons appear to be a multiplicity of software platforms, implementation changes in the Simulator itself, and sometimes a methodological bias of designers driven by a particular agent architecture. In this paper we describe alternative methods of agent-environment synchronisation, introduce a simple software tool for analysing RoboCup games via server log files, and compare the proposed synchronisation alternatives with respect to certain quantitative metrics. This comparative analysis is conducted without varying situated, tactical or strategic agent skills, highlighting purely synchronisation features.

1 Introduction

A distinction between a softbot (synthetic agent) and a software program (module or subroutine) has been extensively studied in context of multi-agent systems. Agent characteristics such as self-containment, temporal continuity, reactivity, pro-activeness, autonomy, etc. are often used to illustrate properties that are unique to agency, as opposed to software modules. In general, agents are supposed to make their decisions and update their behaviour on the basis of local, rather than global, information. Multi-agent interactions lead to emergent patterns in overall system behaviour. In general, emergent behaviour cannot be predicted or even envisioned from knowledge of what each component does in isolation [1]. We believe that one of the most significant aspects making an agent something more than just a software module, is its existence in an environment, its interactions with the environment, and ultimately its adaptation to the environment.

The Simulation League of the RoboCup provides a standard competition platform where teams of software agents play against each other [2]. As in the real game, “players” have only fragmented, localised and imprecise information of the field, and must respond to actions and events in limited time. Recent

RoboCup literature investigated various aspects of RoboCup simulation, including learning of basic and tactical skills [8], genetic evolution of agent behaviors [3], different levels of agents reasoning abilities [6], cooperation between agents [9], opponent modelling [10], etc. The Simulator which creates the environment has also been comprehensively described on both implementation and semantic levels [2, 5]. A particular problem of agent's synchronisation with the environment has been addressed as well, for example in [4]. However, we believe that a systematic comparative analysis of different approaches to the synchronisation problem remains an open challenge. We feel that such analysis could shed more light on the nature of agents interactions with the RoboCup environment.

In this paper we describe alternative methods of agent-environment synchronisation, introduce a simple software tool for analysing RoboCup games via log files, and compare the proposed synchronisation alternatives with respect to certain quantitative metrics. It is worth pointing out that this comparative analysis is conducted without varying situated, tactical or strategic agent skills, highlighting purely synchronisation features.

2 Synchronisation Alternatives

The RoboCup Simulation League provides essentially a pseudo real-time environment. This platform exposes the inherently problematic nature of timing which, as a consequence, results in many engineering problems. Client host platforms typically vary greatly in the available time resolution at both the hardware and in system library level. In addition there is the issue of network speed and reliability, and available resources such as CPU time and memory.

One of the implementation challenges posed by this is synchronisation with the Soccer Server, which we will refer to as the “server”. By default the server simulates time periods of 100ms known as “server cycles”, in which a client may send a command. Contiguous to this, sense_body information (kinematic parameters, stamina, etc) is sent at 100ms intervals, and visual information at 150ms intervals. As documented, the server will execute one and only one command per cycle received from any single client, hence the challenge is to ensure that at most one command is sent. We call the detrimental situation where a client sends two or more commands in a server cycle a “clash”. Another potentially harmful situation is where no commands are sent in a server cycle, even though commands are sent in each adjacent server cycle; we call this a “hole”. For example we would expect a client chasing the ball to have an unbroken sequence of dash and turn commands until it reaches the ball. Occurrences of holes generally slow the agents, and hence disadvantage them in intercepting and running for the ball.

We consider only the default situation ignoring the complexities introduced by the alternative visual information delivery periods provided at the cost of either cone of vision, quality or both. Most obviously we are faced with the problem that 50 % of all visual information is delivered mid server cycle, and every third cycle a client may receive no current information at all. Secondly, we

also find that there is some elasticity in the server cycles. That is, the windows may not always be 100ms, they are generally never less than 100ms, but according to the resources available and calculations required they may grow, by multiples of 10ms. From an implementation perspective, inaccuracies in the system timing routines are due to timer resolution, limited resources, and the process-scheduling algorithm employed. There is some latency (though it may be negligible), in the distributed multi-process environment in which the simulation takes place. Lastly, packets may be irrevocably lost in UDP transmission; fortunately this is very uncommon. In summary, synchronisation with “elastic” server cycles in the presence of asynchronous visual information becomes a challenging task.

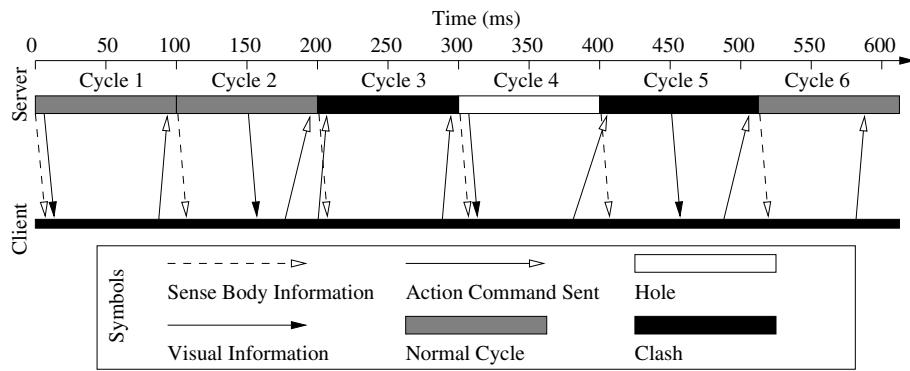


Fig. 1. Synchronisation problem.

Our clients are multi-threaded, where a single thread is dedicated to sensing (watching the UDP port) and reasoning, and another is for acting (dispatching commands, and subsequently synchronisation). Upon receiving visual information the client determines which type of commands and in what sequence they should be executed. The number of commands inserted typically falls within the range of 0 to 4. These are then placed in a thread-safe queue. The commands are extracted one at a time from the queue as determined by the acting thread. We have found many possible synchronisation variations, and have distilled these into four categories, which seem to address the dominant aspects of the problem.

These scheduling schemes can be conveniently categorized according to certain traits. Firstly we say the timing mechanism is either, external or internal. External timing can be thought of as observation of change in the environment, which is expressed conveniently as the sense_body information in this domain. Internal timing can be thought of as a biological clock or regulation mechanism, realized by our use of the operating system timing routines. Separate to the timing mechanism, we may further classify a synchronisation schema as “On Demand”. “On Demand”, may be thought of as “as immediately needed” or “as immediately required”.

2.1 Internal Basic

The internal basic scheme is the simplest, and appears to provide the least optimal performance. The acting thread counts out intervals of 100ms, and then attempts to send a command to the server if one is available. This works on the premise that if commands are sent only at 100ms intervals, these adjacent time points will occur within adjacent server cycles. This methodology does not directly address the issue of mid cycle information or cycle elasticity. It does indirectly, in a probabilistic way, address the issue as the time point may occur anywhere within the window. However, this illuminates an additional problem. Even given the situation of inelastic cycles, commands dispatched very close to the end of a server cycle (typically within the last 10ms) may arrive in either the current cycle or next in stochastic fashion; somewhat like an erratic pendulum (see, for example, cycle 4 in Figure 2). In short, the internal basic scheme may work only under certain assumptions, achieving good synchronisation on probabilistic average. However successful expression of some behaviors may demand more precise synchronisation.

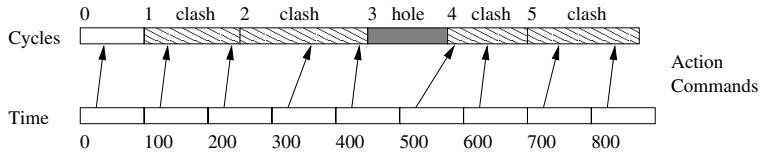


Fig. 2. Synchronisation with internal basic scheme.

2.2 Internal On-Demand

This scheme varies from the Basic implementation in two ways. Firstly, in addition to counting the 100ms intervals, the acting thread now receives notice of the insertion of commands into the queue. Secondly, if no command was sent at the last interval, and there is no command to be sent at the current interval, the thread will sleep until a command is inserted in the queue. This may advantage the agents by providing them with immediate response to sudden events. However, the internal on-demand scheme also suffers from an inability to “guess” a server cycle’s start point and duration. Neither of the internal schemes adequately addresses the issue of mid cycle information or cycle elasticity and both appear to be greatly limited in utility.

2.3 External Basic

In this schema the sense_body information is utilized to indicate the commencement of a new server cycle. This information allows us to use the concept of a “window”. This is a time period that indicates the “window of opportunity” in which the client may reason and subsequently place commands in the queue during which time the acting thread will not attempt to dispatch the first command from the queue. This directly addresses the issue of elasticity in server cycles, as the sense_body information is also subject to this condition (i.e., if server cycle expands, the sense_body interval follows). The issue of mid-cycle information

may also be addressed by selecting a window size of 0.5 of a server cycle plus the average time needed to generate new commands (as during cycle 5 in Figure 3). Obviously, if the time taken exceeds the average, the mid-cycle information will not be utilized (see, for example, cycle 2 in Figure 3).

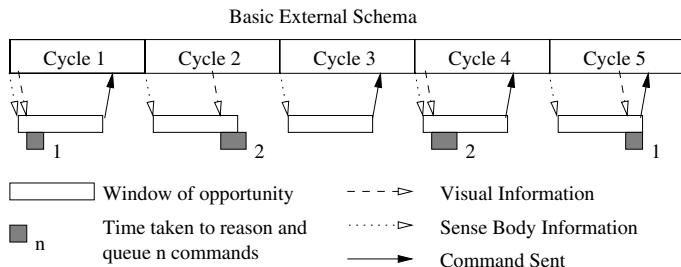


Fig. 3. Synchronisation with external basic scheme.

2.4 Flexible Windowing

In order to attempt to realize the optimal situation as often as possible, Flexible Windowing attempts to fuse the concept of “On-Demand” with that of “Windowing”. This schema is almost identical to External Basic Windowing with the exception, that the amount of time before dispatching a command is varied according to the state of the queue. If the queue was emptied in the previous window, the action thread may select to sleep for a different (typically shorter) duration, than it would if the queue still contains commands — see, for instance, cycle 2 in Figure 4, where the mid-cycle information is utilized and a command is dispatched at the end of a (possibly) shorter window. A special case is where the time to wait in both circumstances is equal, and we call this “Fixed Windowing”. In this case the state of the queue is irrelevant.

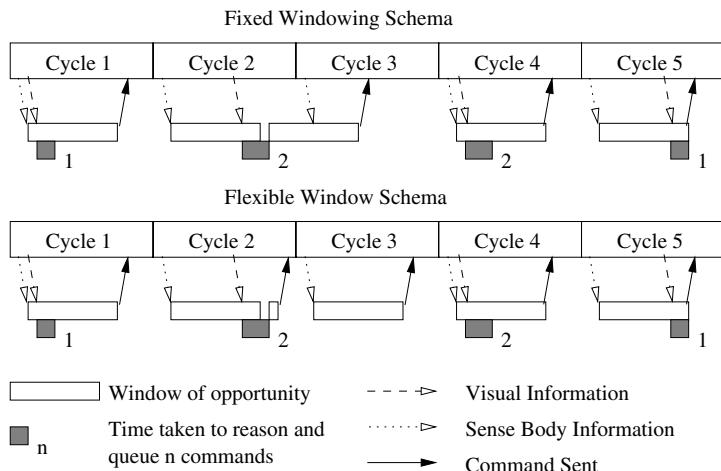


Fig. 4. Flexible synchronisation.

3 Analysis of Game Logs

The AGL (Analysis of Game Log) tool was initially developed to provide an empirical measurement of the success of our alternative synchronisation schemes. This is made possible by an option in the Soccer Server configuration file, which enables the server to maintain a log of the commands sent by each client. We will refer to this file as the “Game Log”. The game log layout is simple to parse, yet contains considerable information. Each line consists of the id of the server cycle, the player id and the command sent verbatim. The following is an extract from a game log, selecting commands received from a particular client, in this case number 3 of the “Fixed Windowing” team:

```
2080 Recv fixedwin_3 : (turn - 29.76)
2082 Recv fixedwin_3 : (change_view normal high)
2082 Recv fixedwin_3 : (turn 17.52)
2083 Recv fixedwin_3 : (dash 35.00)
2085 Recv fixedwin_3 : (dash 35.00)
2086 Recv fixedwin_3 : (dash 35.00)
2089 Recv fixedwin_3 : (dash 100.00)
2089 Recv fixedwin_3 : (dash 100.00)
2091 Recv fixedwin_3 : (dash 100.00)
```

It is easy to see, for example, a “clash” occurring at server cycle 2089, followed by a potential “hole” at 2090.

Currently the AGL is implemented as a Perl 5 script, which not only parses and collates data from the log file, but also provides a terse analysis. The AGL tool is sufficiently flexible to be able to handle partial games. Many (though not all) measurements may be made with any variable number of players from 1 to a full 11. In addition to the obvious comparison of goals scored, the AGL calculates a number of measurements, provided both as a count (raw data), and as a percentage. The meaning of the percentage varies between measurements and is described separately with each formula.

It is well-documented that some of the client commands can be executed in parallel during one server cycle (for example, “say” or “change view”). Others, like “dash”, “turn”, “kick” and “catch”, are executed only once per cycle. We are mostly interested in synchronising the latter command type, which we shall refer to as “action” commands, or “actions”. Let us introduce the following notation:

- $\alpha_{i,j}$ is a function returning the number of all action commands received by the server from a player i at cycle j ;
- $\lambda_{i,j}$ is a function returning 1 if the server received one or more action commands from a player i at cycle j , and 0 otherwise;
- $\delta_{i,j}$ is a boolean function returning *true* if $\alpha_{i,j} = 0$, and *false* otherwise;
- $\gamma_{i,j}$ is a boolean function returning *true* if the server received a “dash” action command from a player i at cycle j , and *false* otherwise;
- $\theta(a)$ is a function returning 1 if a boolean expression a is true, and 0 otherwise.

In short, $\lambda_{i,j}$ is 1 if the cycle j was used by the player i (even if more than once), while $\delta_{i,j}$ represents unused cycles.

“Activity” measurement is given by the formula:

$$\frac{\sum_{i=1}^n \sum_{j=1}^m \alpha_{i,j}}{n * m}$$

where n is a number of players, and m is a number of server cycles.

An activity level of 100 % indicates that each client in the team sent an action command in every cycle. Activity provides some interesting information. In particular, we have observed that a winning side often has a lesser activity, possibly making a losing side chase the ball while denying them a good passing game.

“Clashes ratio” measurement is given by the formula:

$$\frac{\sum_{i=1}^n \sum_{j=1}^m (\alpha_{i,j} - \lambda_{i,j})}{\sum_{i=1}^n \sum_{j=1}^m \alpha_{i,j}} = 1 - \frac{\sum_{i=1}^n \sum_{j=1}^m \lambda_{i,j}}{\sum_{i=1}^n \sum_{j=1}^m \alpha_{i,j}}$$

where n is a number of players, and m is a number of server cycles.

Basically, this ratio represents the frequency of clashes with respect to all action commands, and is primarily used to determine the quality of synchronisation between the clients and the server.

“Holes ratio” measurement is given by the formula:

$$\frac{\sum_{i=1}^n \sum_{j=2}^{m-1} \theta(\gamma_{i,j-1} \wedge \delta_{i,j} \wedge \gamma_{i,j+1})}{\sum_{i=1}^n \sum_{j=1}^m \alpha_{i,j}}$$

where n is a number of players, and m is a number of server cycles.

Intuitively, this ratio represents the sum of all unused cycles j occurring between the adjacent cycles $j - 1$ and $j + 1$ in which the client sent “dash” commands, calculated for the whole team in proportion to the total number of action commands sent (including clashes) for players on the same team. Holes (more precisely, dash holes) are used in conjunction with clashes to determine the quality of synchronisation between the clients and the server.

Here is an example report produced by the AGL for two teams (after a 6000 cycles game):

Team: flexwin		Team: fixedwin			
All commands..	48276	(73.15%)	All commands..	46414	(70.32%)
Activity.....	41443	(62.79%)	Activity.....	39988	(60.59%)
Holes.....	164	(0.34%)	Holes.....	259	(0.56%)
Clashes.....	2544	(5.27%)	Clashes.....	1340	(2.89%)
Kicks.....	479	(0.99%)	Kicks.....	462	(1.00%)
Dashes.....	22785	(47.20%)	Dashes.....	21323	(45.94%)
Turns.....	18079	(37.45%)	Turns.....	18181	(39.17%)
Catches.....	100	(0.21%)	Catches.....	22	(0.05%)

The simple format of the game log requires low resource overhead to generate during runtime, thus minimising impact on the experiment. The time required by AGL to generate a report such as the one above is quite small, though proportional to the size of the log. This provides information promptly after the completion of the game, allowing an experimenter to quickly confirm their impressions. The log also contains an identical depth of information for both teams participating in the game. This enables an analysis of games involving heterogeneous teams, provided they use the same version of the Soccer Server protocol.

4 Preliminary Comparative Analysis

In this section we briefly describe an experiment which exemplifies how the AGL tool is used in comparative analysis of alternative synchronisation schemes summarised in section 2. Four schemes were selected for the experiment: internal on demand, basic external, flexible windowing and fixed windowing. Consequently, four different teams were compiled, and played 15 round-robin tournaments (first 3 tournaments with an aggressive tactical formation 3-5-2, and the next 12 tournaments with a more conservative 5-3-2 formation), resulting in 45 games played by each team (9 in the first experiment and 36 in the second). The overall competition performance is summarised in the following tables (a win earns 3 points, and a draw 1 point).

	Internal	Basic External	Flex Windowing	Fixed Windowing
Wins	0	5	7	4
Losses	8	3	1	4
Draws	1	1	1	1
Goals For	14	36	42	30
Goals Against	34	23	20	25
Total Points	1	16	22	13

Table 1. Results with the 3-5-2 formation.

	Internal	Basic External	Flex Windowing	Fixed Windowing
Wins	8	16	16	9
Losses	18	8	8	15
Draws	10	12	12	12
Goals For	22	37	39	26
Goals Against	36	26	30	32
Total Points	34	60	60	39

Table 2. Results with the 5-3-2 formation.

These results clearly rule out the “internal on demand” synchronisation — the corresponding team managed to get only one draw in 9 games, losing 8 times, with the 3-5-2 formation, and finished clear last with the 5-3-2 formation.

Three other teams were quite competitive — no one escaped without losing to some other team in the first short experiment, and the next experiment has shown some close scores as well. At the end, “flexible windowing” edged ahead overall, and “fixed windowing” finished clear third. “Basic external” proved to be a solid performer. In fact, it shared the first place with “flexible windowing” in the second experiment. It should be pointed out, however, that more games with alternative formations should be played to get more statistically significant results, and other placements of these three external-based schemes are possible.

At this stage, we intended just to demonstrate how tournament results can be complemented by the AGL tool. The following table contains average and standard deviation measurements for “activity”, “clash” and “dash holes” ratios.

Name	Clashes		Holes		Activity	
	Average	Std. Dev	Average	Std. Dev	Average	Std. Dev
Internal	3.67	0.70	0.21	0.03	68.05	3.61
Basic External	3.01	2.14	0.57	0.11	58.49	4.45
Flexible Windowing	5.26	0.73	0.46	0.08	61.12	2.59
Fixed Windowing	3.01	1.82	0.59	0.10	58.92	3.47

Table 3. AGL statistics with the 3-5-2 formation.

Name	Clashes		Holes		Activity	
	Average	Std. Dev	Average	Std. Dev	Average	Std. Dev
Internal	0.31	0.22	0.16	0.03	61.45	3.67
Basic External	0.03	0.03	0.64	0.09	52.70	3.47
Flexible Windowing	1.19	0.44	0.50	0.09	55.00	4.02
Fixed Windowing	0.08	0.06	0.60	0.10	53.81	3.59

Table 4. AGL statistics with the 5-3-2 formation.

It should be noted that the second experiment was conducted in a better networking environment. It appears that the tournament winner has, in fact, a higher percentage in the “clash ratio” — but with a significantly lesser standard deviation in the first experiment, making this scheme more stable and robust under less “friendly” conditions. In addition, its “holes ratio” is smaller among three best teams in both experiments. The “internal” scheme exhibited quite comparable ratios, but the team was too much active — reflecting the fact that a large portion of commands arrived at incorrect or non-optimal cycle, and the players had to chase the ball more than opponents. Again, more games would definitely provide statistically better results.

5 Conclusions

We felt that mechanical analysis of game logs added greatly to comparison of goals scored, and observations by what are now “expert eyes” amongst the devel-

opment team. In developing AGL, we have continued to find that more involved analysis of the log may provide us with a multitude of parameters by which we can compare teams from various stages of development with one another, as well as other teams.

Our preliminary comparative analysis was carried out without varying situated, tactical or strategic agent skills and highlighted purely synchronisation issues. The agents used in the experiments were developed in accordance with the Deep Behaviour Projection agent architecture, which provided a systematic support for design and full implementation of the team Cyberoos [7]. In general, agent-environment synchronisation is only one factor contributing to a teams overall performance. The precise nature of relationships and dependencies between synchronisation schemes and the agent architecture is a subject of future research.

References

1. John L. Casti. *Would-be Worlds. How Simulation is Changing the Frontiers of Science*. John Wiley & Sons, 1997.
2. Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela M. Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda and Minoru Asada. The RoboCup Synthetic Agent Challenge. In Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997.
3. Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson and James Hendler. Co-evolving Soccer Softbot Team Coordination with Genetic Programming. In RoboCup-97: Robot Soccer World Cup I (Lecture Notes in Artificial Intelligence No. 1395), H. Kitano, ed. Berlin: Springer-Verlag, 398–411, 1998.
4. Kostas Kostiadis and Huosheng Hu. A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition, In Proceedings of the IJCAI'99 RoboCup Workshop, Stockholm, August 1999.
5. Mikhail Prokopenko, Ryszard Kowalczyk, Maria Lee and Wai-Yat Wong. Designing and Modelling Situated Agents Systematically: Cyberoos'98. In Proceedings of the PRICAI-98 Workshop on RoboCup, 75–89. Singapore 1998.
6. Mikhail Prokopenko and Marc Butler. Tactical Reasoning in Synthetic Multi-Agent Systems: a Case Study. In Proceedings of the IJCAI-99 Workshop on Non-monotonic Reasoning, Action and Change, 57–64. Stockholm 1999.
7. Mikhail Prokopenko, Marc Butler and Thomas Howard. On Emergence of Scalable Tactical and Strategic Behaviour. To appear in Proceedings of the RoboCup-2000 Workshop, Melbourne 2000.
8. Peter Stone and Manuela Veloso. Team-Partitioned, Opaque-Transition Reinforcement Learning. In Proceedings of the Third International Conference on Autonomous Agents (Agents'99) and in "RoboCup-98: Robot Soccer World Cup II", M. Asada and H. Kitano (eds.), Springer Verlag, Berlin, 1999.
9. Peter Stone and Manuela Veloso. Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. In Artificial Intelligence, volume 100, number 2, June 1999.
10. Peter Stone, Patrick Riley, and Manuela Veloso. Defining and Using Ideal Teammate and Opponent Agent Models. In Proceedings of the Twelfth Innovative Applications of AI Conference (IAAI-2000).

Extended Q-learning : Reinforcement Learning using Self-Organized State Space

Shuichi ENOKIDA, Takeshi OHASI,
Takaichi YOSHIDA and Toshiaki EJIMA

Kyushu Institute of Technology, Dept. of Artificial Intelligence,
680-4 Kawazu, Iizuka, Fukuoka, 820-8502, Japan
{enokida, ohashi, takaichi, toshi}@mickey.ai.kyutech.ac.jp

Abstract. We propose Extended Q-learning. To accommodate continuous state space directly and to improve its generalization capability. Through EQ-learning, an action-value function is represented by the summation of weighted base functions, and an autonomous robot adjusts weights of base functions at learning stage. Other parameters (center coordinates, variance and so on) are adjusted at unification stage where two similar functions are unified to a simpler function.

1 Introduction

Reinforcement Learning has received a great attention as a method that trains an autonomous robot to make an appropriate action aimed at accomplishing a task with little or no premise knowledge. Q-learning[6], a well-known reinforcement learning, tries to learn an action-value function in order to find optimal policy with respect to some object functions.

Typically, even if an autonomous robot has continuous sensor values, they are quantized to reduce learning time. As a result, learning algorithms which work on the discrete space are easily designed and analyzed from a point of optimal control. However, it is costly to design a state space to accomplish a task smoothly. The method which constructs state space automatically by off-line learning is proposed by Asada and al[1]. This is, without a good quantization of state space, reinforcement learning algorithms including Q-learning suffer from lack of robustness and lack of generalization.

To overcome the above, we propose Extended Q-learning(EQ-learning). EQ-learning is designed to accommodate continuous sensor space without quantization of it and to improve generalization capability of Q-learning. EQ-learning represents the action-value function by the summation of weighted base functions which is generally used to estimate a continuous value[3][5].

An EQ-learning algorithm has two stages called by learning stage and unification stage. A robot adjusts only weights of base functions through interactions at learning stage. In addition to that, at unification stage, a robot makes an effort to unify two similar base functions into one base function to obtain a simpler representation of action-value function. That is, at unification stage, based on

two similar base functions, a new base function is calculated and two functions are replaced with it. A simpler representation caused by the unification leads to a robust model which works well even in a noisy environment as well as a general model which also works well in a slightly different environment.

Accordingly, our EQ-learning leads to a generalization of Q-learning and promises a much better learning performance capability than ordinary learning methods, including Q-learning. We performed simulation of two learning methods and preliminary results verified the higher performance capability of EQ-learning than Q-learning.

1.1 Q-learning

Q-learning algorithm is generally used as a method that give us a sophisticated solution to reinforcement learning problems.

If a robot transits from a state $s \in \mathcal{S}$ to a state $s' \in \mathcal{S}$ by selecting an action $a_i \in \mathcal{A}$, this algorithm updates the utility $Q(s, a_i)$ as,

$$Q(s, a_i) \leftarrow Q(s, a_i) + \alpha(r(s, a_i) + \gamma M(s') - Q(s, a_i)), \quad (1)$$

$$M(s) = \max_{j \in \mathcal{A}} Q(s, a_j), \quad (2)$$

where α is the learning rate and γ is the discounting factor.

The reinforcement signal $r(s, a_i)$ is accorded to a robot by an environment, when a robot accomplished a task. Thus, the reinforcement signal is added to the utility in the state which transited to “goal” by one action. The Q-learning algorithm updates the utility to minimize a temporal-difference of two utilities in current state and next state. As a result after a sufficient number of iteration, selecting an action a_i which has the maximum value of the utility $Q(s, a_i)$ in a state s is the optimal policy.

2 Extended Q-learning

2.1 Continuous action-value function

EQ-learning extends Q-learning in order to accommodate continuous sensor space directly. As a result, EQ-learning is able to minimize errors caused by the quantization of continuous sensor values.

EQ-learning represents the utility function by the summation of weighted base functions, as follows,

$$U(\mathbf{x}, a_i) = \sum_{m=1}^N W_m(a_i) B_m(\mathbf{x}). \quad (3)$$

To estimate the optimal utility (action-value) function, it is necessary to adjust these parameters (weight, variance, mean, and so on).

We consider that the reinforcement learning in the quantized sensor space is equivalent to applying the rectangular function as a base function. EQ-learning with rectangular function is equivalent to Q-learning.

$$B_m(\mathbf{x}) = \begin{cases} 1 & |\boldsymbol{\mu}_m - \mathbf{x}| \leq \text{Width}_m, \\ 0 & |\boldsymbol{\mu}_m - \mathbf{x}| > \text{Width}_m, \end{cases} \quad (4)$$

where Width_m is the width of a rectangular function. The Gaussian function is applied as the base function too.

$$B_m(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_m)^t \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_m)\right). \quad (5)$$

Because these two functions have a locality feature, we choose them as a base function for EQ-learning.

2.2 Adjustment of weights of base functions

Generally, learning parameter increasing will also increase learning time[7]. The number of weight parameters is proportional to the number of base functions. But the number of center coordinate parameters and variance parameters are proportional to the number of base functions and sensor dimensions. Thus, in EQ-learning, an autonomous robot adjusts only weight of base functions in each step to decrease learning time.

The EQ-learning algorithm updates weights of base functions through interactions with a given environment and minimizes the temporal-difference. A robot transits from a state $\mathbf{x} \in \mathbf{X}$ to $\mathbf{x}' \in \mathbf{X}$ by selecting an action $a_i \in \mathbf{A}$, the temporal-difference $E(\mathbf{x}, a_i)$ is,

$$\begin{aligned} E(\mathbf{x}, a_i) &= (r(\mathbf{x}, a_i) + \gamma M(\mathbf{x}') - U(\mathbf{x}, a_i))^2, \\ &= (r(\mathbf{x}, a_i) + \gamma \max_{a_k \in \mathbf{A}} U(\mathbf{x}', a_k) \end{aligned} \quad (6)$$

$$- \sum_{m=1}^N W_m(a_i) B_m(\mathbf{x}))^2. \quad (7)$$

We obtain a vector \mathbf{e} which decreases the temporal-difference by partial derivative of $E(\mathbf{x}, a_i)$ with respect to $W_j(a_i)$, j is 1 to N , as follows,

$$\mathbf{e} = \left(\frac{\partial E(\mathbf{x}, a_i)}{\partial W_1(a_i)} \quad \frac{\partial E(\mathbf{x}, a_i)}{\partial W_2(a_i)} \quad \dots \quad \frac{\partial E(\mathbf{x}, a_i)}{\partial W_N(a_i)} \right)^t. \quad (8)$$

Therefore, EQ-learning algorithm updates all of weights of base functions as,

$$\begin{aligned} W_m(a_i) &\Leftarrow W_m(a_i) + \\ &N_m(\mathbf{x})(r(\mathbf{x}, a_i) + \gamma M(\mathbf{x}') - V(\mathbf{x}, a_i)), \end{aligned} \quad (9)$$

where,

$$N_m(\mathbf{x}) = \alpha \frac{B_m(\mathbf{x})}{\sum_j B_j(\mathbf{x})}, \quad (10)$$

$$M(\mathbf{x}) = \max_{k \in A} V(\mathbf{x}, k), \quad (11)$$

where α is the learning rate and γ is the discounting factor.

2.3 Adjustment of other parameters

The more number of base functions is used, the more exactly approximation model can be obtained. However, increasing learning parameter will also increase learning time as well. Thus, in order to obtain an appropriate and simple model, a robot adjusts other parameters (the number of base functions, center coordinates, and variances) automatically by unification of two similar base functions into a single base function. We define that the similarity is calculated by a **K-L information number** and a **Maharanobis distance**.

The probability distributions in center coordinates of two base functions B_n and B_m are,

$$P(\boldsymbol{\mu}_n) = \{p(\boldsymbol{\mu}_n, a_1), \dots, p(\boldsymbol{\mu}_n, a_K)\}, \quad (12)$$

$$P(\boldsymbol{\mu}_m) = \{p(\boldsymbol{\mu}_m, a_1), \dots, p(\boldsymbol{\mu}_m, a_K)\}. \quad (13)$$

Thus, the K-L information number $I(P(\boldsymbol{\mu}_n); P(\boldsymbol{\mu}_m))$ of $P(\boldsymbol{\mu}_n)$ related to $P(\boldsymbol{\mu}_m)$ is calculated as,

$$I(P(\boldsymbol{\mu}_n); P(\boldsymbol{\mu}_m)) = \sum_{i=1}^K p(\boldsymbol{\mu}_n, a_i) \log \frac{p(\boldsymbol{\mu}_n, a_i)}{p(\boldsymbol{\mu}_m, a_i)}. \quad (14)$$

The Maharanobis distance $d_n(\mathbf{x})$ from $\boldsymbol{\mu}_n$ to \mathbf{x} is calculated as,

$$d_n^2(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_n)^t \Sigma_n^{-1} (\mathbf{x} - \boldsymbol{\mu}_n). \quad (15)$$

We define the similarity $s_n(m)$ from the base function B_n to the base function B_m as,

$$s_n(m) = \exp(-aI(P(\boldsymbol{\mu}_n); P(\boldsymbol{\mu}_m)) - bd_n^2(\boldsymbol{\mu}_n - \boldsymbol{\mu}_m)), \quad (16)$$

where a and b are constants. During the learning, the robot unifies two base functions that have high similarity into a single base function to obtain a simpler representation of the utility function. If the parameter $S_{nm}(t)$ is larger than the threshold th , the unification of two base functions (B_n and B_m). The parameter $S_{nm}(t)$ is calculated as,

$$S_{nm}(0) = 0. \quad (17)$$

$$S_{nm}(t+1) = (1 - \beta)S_{nm}(t) + \beta s_n(m), \quad (18)$$

where $0 \leq \beta \leq 1$ is the summation rate and t is increased when the robot is accorded with the reinforcement signal.

Based on two similar base functions, a new base function B_{new} is calculated and two base functions are replaced with it, as follows,

$$\boldsymbol{\mu}_{new} = \frac{d_m^2(\boldsymbol{\mu}_n)}{D_{nm}} \boldsymbol{\mu}_n + \frac{d_n^2(\boldsymbol{\mu}_m)}{D_{nm}} \boldsymbol{\mu}_m, \quad (19)$$

$$D = d_n^2((\boldsymbol{\mu}_m)) + d_m^2((\boldsymbol{\mu}_n)), \quad (20)$$

$$\boldsymbol{\Sigma}_{new} = \boldsymbol{\Sigma}_n + diag(\Delta x_i^2), \quad (21)$$

$$\Delta x_i = \mu_{new}(i) - \mu_n(i). \quad (22)$$

Weights of bases functions are calculated, as follows,

$$W_{new}(a_i) = W_n(a_i)B_n(\boldsymbol{\mu}_n) + W_m(a_i)B_m(\boldsymbol{\mu}_m). \quad (23)$$

By these algorithms, a robot can estimate the utility function by simpler functions which are obtained automatically.

2.4 Action selection rule

The simplest action selection rule is to select an action which has the highest action-value in each state. However, in learning phase, this rule has no chance to obtain a better policy than the current one. To overcome the above, generally, mapping a state \mathbf{x} to an action a_i is stochastically, as follows,

$$\mathbf{P}(\mathbf{x}) = (p(\mathbf{x}, a_1), p(\mathbf{x}, a_2), \dots, p(\mathbf{x}, a_K)). \quad (24)$$

$$0 \leq p(\mathbf{x}, a_i) \leq 1, \quad (25)$$

$$\sum_{i=1}^K p(\mathbf{x}, a_i) = 1. \quad (26)$$

We decide that the robot is in a state \mathbf{x} , an action $a_i \in \mathbf{A}$ is selected stochastically according to Boltzmann distribution, as follows,

$$p(\mathbf{x}, a_i) = \frac{\exp(U(\mathbf{x}, a_i)/T)}{\sum_{j=1}^K \exp(U(\mathbf{x}, a_j)/T)}, \quad (27)$$

where T is the scaling constant and utility function $U(\mathbf{x}, a_i)$ is the utility of selecting an action a_i in a state \mathbf{x} .

3 Simulation and result

In our simulations, we train an autonomous robot to get a ball faster which is allocated randomly in a given environment. Therefore, the robot creates the utility function to get the ball through EQ-learning. In this simulation, the robot can identify the ball through the camera and select one of the given five actions (see Figure 1). The reinforcement signal accorded by the environment is,

$$r(\mathbf{x}, a_i) = \begin{cases} 1 & \text{if the robot gets the ball,} \\ 0 & \text{otherwise.} \end{cases} \quad (28)$$

The other parameters for robot learning are, as follows,

$$\alpha = 0.25, \quad \gamma = 0.99, \quad T = 0.1. \quad (29)$$

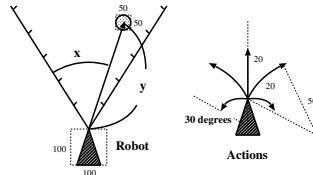


Fig. 1. Robot parameters in our simulation.

In order to assess the capability of our EQ-learning, we report results of a robot that can speed up to get a ball in several environments by checking average steps to carry out the task. These assessment run 1000 times at each number of trials(10, 20, 30, 40, 50, 100, 150, 200) to calculate these average steps to get a ball allocated randomly.

3.1 Simulation experiment

To assess our EQ-learning, we have two set of experiments. For the first set of experiment, a robot adjusts only weights of rectangular functions. This algorithm is equivalent to an ordinary Q-learning algorithm. At the second set, through EQ-learning, the robot adjusts weights and unifies base functions automatically. Therefore, the robot creates the utility function approximation by adjusting all parameters (weights, center coordinates, variances, and the number of base functions). In the first case, 16 (4×4) base functions are allocated in the sensor space, and the robot learns and adjusts these parameters. In the second set, the Gaussian function is used as the base function. In this experiment, parameters for the unification are, as follows,

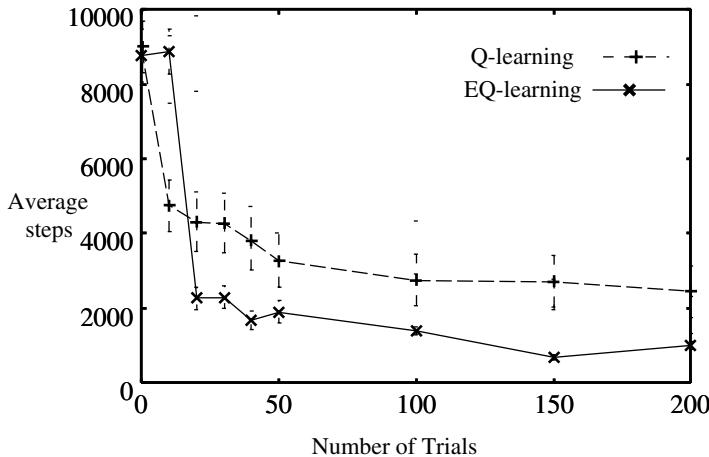
$$a = 1, \quad b = 0.5, \quad \beta = 0.1, \quad th = 0.25. \quad (30)$$

3.2 Result

Average steps and the example of unification of base functions are plotted against trial number in Figure 2. This figure shows that EQ-learning is useful to estimate the utility function. Because of unification, parameters that a robot must adjust decrease in an appropriate number. That is, the robot can get a simpler and appropriate model for learning to get the ball. An example of learning model is shown below in Figure 2. The figure shows the allocation of base functions. Through EQ-learning, an autonomous robot creates the policy which is equivalent to the natural hand-designed solution (if ball is on left, go left; if on right, go right; if straight ahead, go straight).

4 Experiment for the real robot and the result

RoboCup[4] provides a common task for AI and robotics. We try to train a real robot(Figure 3) to learn the behavior of a goalie robot (intercepting space



Number of trials	0	50	100	200
Average numbers of base functions	16	4.2	2.8	2.5
Example of the unification process				

Fig. 2. Result of the reinforcement learnings and the number of base functions at each number of trials (above), and the example of unification (below).

between the ball and the goal) by EQ-learning. The goalie robot has one video camera to identify and track colored objects (ball, goal, and other robots). There is a problem that the goalie should watch not only front, but also left and right side. There are some solutions, however, we decide to equip the goalie with omnidirectional vision system. The goalie is able to select an action out of five actions, go left(and right) slowly or quickly, and stop.

4.1 Experiment

In this experiment, we observe that an appropriate model for keeping the goal is obtained automatically by EQ-learning. The goalie extracts the regions of a goal and a ball from the input image by using color information. There are two vectors, \mathbf{v}_b and \mathbf{v}_g , one is from the center of input image (goalie robot) to the center of the ball's region and the other is to the goal's one. We then define that the angle made from two vectors is state space. If the angle is,

$$0 \leq x \leq 90, \text{ or } 270 \leq x \leq 360, \quad (31)$$

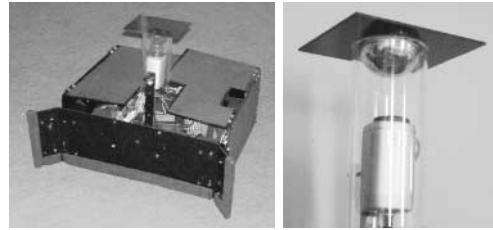


Fig. 3. Real keeper robot which has one omni-direction camera.

it assumes that the ball is in the goal. Thus, the goalie doesn't learn in this state. The reinforcement signal accorded by the environment is,

$$r(x, a) = \begin{cases} 1 & |180 - x(t)| < 20, \\ 0 & \text{otherwise.} \end{cases} \quad (32)$$

In this experiment, parameters for the unification are, as follows,

$$a = 1, \quad b = 0.5, \quad \beta = 0.1, \quad th = 0.25. \quad (33)$$

First allocation of base functions is shown in Figure 5.

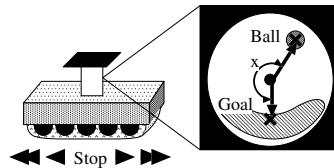


Fig. 4. The goalie can select an action and sense the environment to keep the goal.

4.2 Result

Through EQ-learning, the allocation of base functions is changed into simpler one as shown in Figure 6 and Figure 7. The goalie robot creates an appropriate model and policy which are equivalent to the natural hand-design solution for keeping a goal. The robot has the following policies: (1)go left quickly, (2)go left slowly, (3)go right slowly, (4)go right quickly, which correspond to four base functions in Figure 7 from left to right respectively. As the result, it is expected that the cost to create the learning model and learning time will reduce.

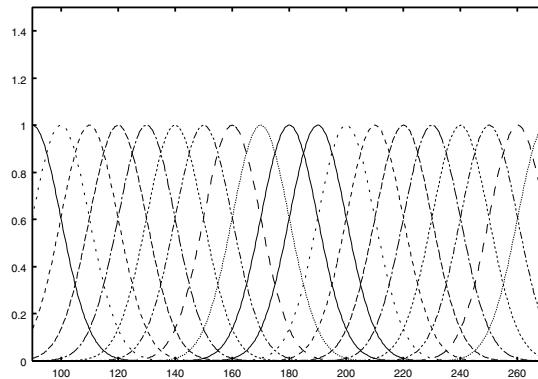


Fig. 5. First allocation of base functions.

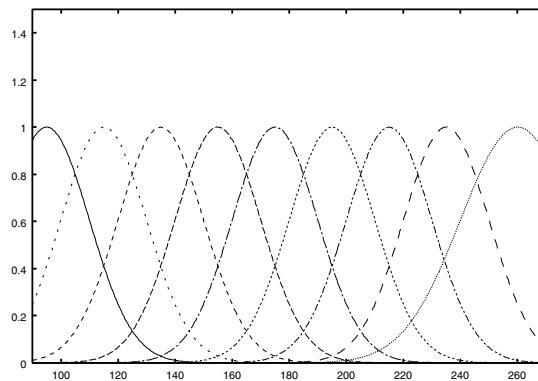


Fig. 6. Simple learning model for keeping a goal.

5 Conclusion and future work

We propose Extended Q-learning that estimates an action-value function by weighted base functions and leads appropriate parameters (weights, center coordinates, variances, and the number of base functions) automatically through the interaction with the environment. As a result, EQ-learning can deal with reinforcement learning algorithm with no quantization of continuous sensor values, and estimate a utility function much better than the ordinary one. Since EQ-learning has the robustness against errors that occurred from the noise of robot sensor parameters, we believe that EQ-learning can deal much better with real robot learning.

Through EQ-learning, we can train a robot to keep the goal. It, however, is fairly simple behavior. The behavior of the soccer needs more complex policy. Therefore, it is not easy for an applied EQ-learning to get the behavior of soccer

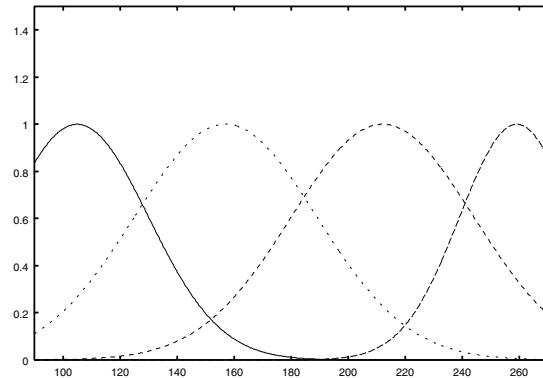


Fig. 7. Simple learning model for keeping a goal.

directly. Thus, we think that hierarchical control structures[2] are useful for this problem. We constitute hierarchical control structures which consists of modules called a subtask and rely on the programmer to design a hierarchy of modules. We train a robot to carry out each subtask by reinforcement learning so that the robot learns the behavior of soccer. We also design the hierarchical control structures to obtain high performance behavior of soccer.

References

1. M. Asada, S. Noda and K. Hosoda : Action Based Sensor Space Segmentation For Soccer Robot Learning. *Applied Artificial Intelligence*, Vol.12, No.2-3 (1998) 149–164.
2. Digney, B. L. : Learning Hierarchical Control Structures for Multiple Tasks and Changing Environments. *From animals to animats 5 : SAB 98*,(1998).
3. Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson : Neuronlike Adaptive Elements that can solve difficult learning control problems. (1983) 834–846.
4. Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara, : RoboCup : A Challenge Problem for AI and Robotics. *Lecture Notes in Artificial Intelligence* Vol.1395 (1998)1–19.
5. Norihiko Ono, Yoshihiro Fukuta : Learning Coordinated Behavior in a Continuous Environment. *Lecture Notes in Artificial Intelligence* Vol.1221 (1997) 73–81.
6. C.J.C.H. Watkins : Learning from delayed rewards. PhD thesis, University of Cambridge (1989).
7. Whitehead : A complexity analysis of cooperative mechanisms in reinforcement learning. *Proc. AAAI-91* (1991) 607–613.

And The Fans are Going Wild! SIG plus MIKE

Ian Frank ¹, Kumiko Tanaka-Ishii ², Hiroshi G. Okuno ^{3,4}, Junichi Akita ⁵
Yukiko Nakagawa ⁴, K. Maeda ⁶, Kazuhiro Nakadai ⁴, Hiroaki Kitano ^{4,7}

¹ Complex Games Lab, ETL, Tsukuba, Ibaraki 305-8568, Japan, ianf@etl.go.jp

² University of Tokyo, Tokyo 113-8656, Japan, kumiko@ipl.t.u-tokyo.ac.jp

³ Science University of Tokyo, Chiba 278-8510, Japan, okuno@nue.org

⁴ Kitano Symbiotic Systems Project, ERATO, JST, Tokyo 150-0001, Japan,
{yuki,nakadai}@symbio.jst.go.jp

⁵ Future University, Hakodate, Japan, akita@fun.ac.jp

⁶ Chiba University, Chiba 263-8522, Japan, kmaeda@cogsci1.chiba-u.ac.jp

⁷ Sony Computer Science Laboratories, Inc., Tokyo 141-0022, Japan,
kitano@csl.sony.co.jp

Abstract. We present an implemented commentary system for real-world robotic soccer. Our system uses an overhead camera to pass game information to the simulator league commentary program MIKE, and uses the humanoid robot SIG to provide a physical embodiment for the commentary. The use of a physical robot allows us to direct audience attention to important events by looking at them, provide more realism to the interaction between MIKE and other humans in the domain, and even set up a dialogue with the audience as part of the commentary itself. Our system combines the multi-modal input of audio and video information to generate a multi-modal commentary that brings the extra dimension of body language to the social interaction that characterises the overall commentary task.

1 Introduction

Good TV or radio commentary of soccer is hard. The action moves fast and the play itself is accompanied by many other events on and around the field. The referee is human and may make mistakes, the players may have rivalries and past histories that result in off-the-ball incidents, the players may argue amongst themselves or with the officials, the attitude of the managers and the team benches may be significant, and the reactions of the crowd can also have an impact on the game.

To date, RoboCup has been successfully used for research on automated commentary by restricting attention to the simulation league [1]. This is still a hard problem, but is much simpler than real-world soccer: information on the player and ball locations is complete and noise-free, there are no linesmen, the referee is just a disembodied decision program, and fouls are limited to free kicks (there are no penalties). In this paper, we describe the first implemented commentary

system for the RoboCup robot leagues. This system works by combining two existing research projects: MIKE and SIG.

MIKE is a commentator system developed for the RoboCup's simulator leagues [2]. We adapted MIKE to take as input the log produced by a vision recognition system that processes the video stream from an overhead camera [3]. This 2D representation of the game enables much of the original MIKE code to be retained. In robotic soccer, however, there are many events (such as humans entering the field of play to replace robots) that are too difficult to recognise automatically. We therefore re-designed MIKE as an *assisted* commentary system, which functions as part of a team with one or more humans. MIKE produces commentary by itself as long as it can follow the flow of the game, but when it has difficulty interpreting the scenes, it passes control of the commentary to a human. MIKE is capable of handling a basic dialogue with a human (by simply monitoring voice levels to identify interruptions and determine when it should speak or be quiet), but for our public demonstrations at RoboCup 2000 we used the simpler alternative of supplying MIKE with a repertoire of 40 *episodes* that it could use verbatim to fill in any breaks in the game.

To actually embody MIKE's commentary and situate the computer commentator in the RoboCup environment, we use SIG, a humanoid robot [4]. SIG has binaural microphones and binocular vision and can locate, and direct its gaze at, objects and sounds in 3D space [5]. Using a physical robot to personify the commentary increases the possibilities for realistic commentary. For example, SIG can direct audience attention by looking at the ball or towards the sound of the referee's whistle when it blows, and can add meaning to the commentary (or the speech of the referee or human commentators) by using head movements. It can also monitor, and react to, the level of excitement shown by the crowd watching the game (although this ability was not implemented in time for Melbourne).

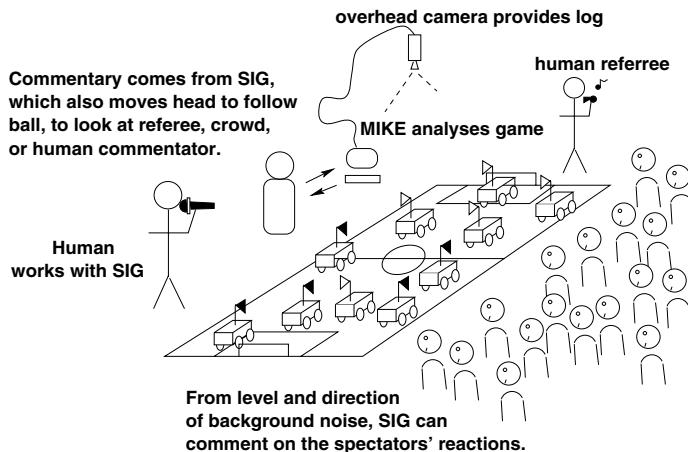


Fig. 1. Pictorial overview of soccer commentary system for real-world robots

Figure 1 shows the overall concept of our commentary system. The rest of this paper describes how we adapted MIKE and SIG towards realising this vision.

2 Overall Design

We work on the principle that the referee, robots, team designers, other commentators, and audience members are all significant actors in the drama of a RoboCup game. To produce an integrated system combining SIG and MIKE with all these other agents, we designed the simple architecture of Figure 2. In this figure, the ball and players' positions are assumed to come from the global vision system initially developed by [3]. Our improved version of this system can capture 30 frames per second on a PentiumIII at 450MHz. We identify robots by finding the five most promising matches for the ping-pong balls of each team, but also carry out error reduction by only looking for robots within the *detection radius* (usually, less than 50cm) of the positions where robots were successfully identified in the previous frame. Our implementation also identifies the extra markers of the robots (without limiting the detection radius) and humans entering the pitch (via an increase in the number of extra markers).

MIKE produces the basic commentary for a game on the basis of the information from this global vision system. Although MIKE was primarily designed to work with the logs of simulated games, a large amount of the basic code was retained (we discuss the changes in §3). The data from the global vision system is also passed to SIG to allow it to easily implement ball and robot tracking. Implementing such tracking using SIG's own cameras is future work.

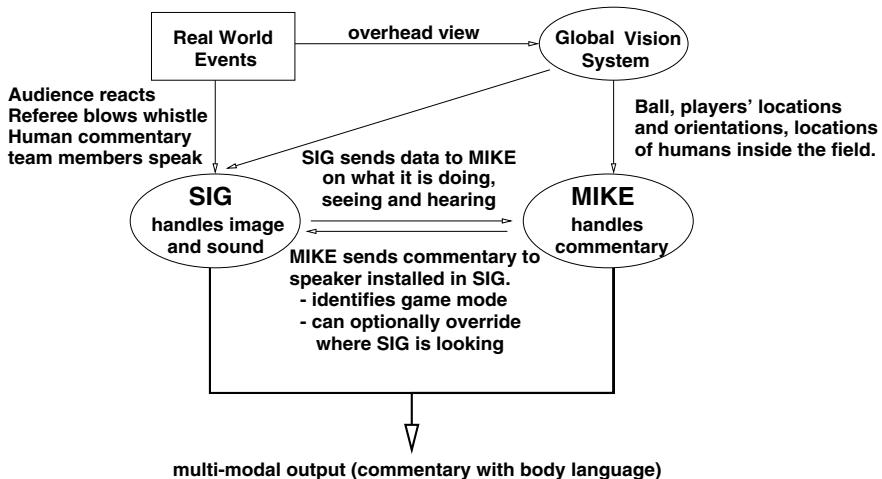


Fig. 2. Architecture combining MIKE and SIG to produce an embodied commentator

	Play mode	Stoppage mode
SIG	<ul style="list-style-type: none"> • follow ball • look towards sound of whistle • look at speaking commentator • tell MIKE what it does, sees, hears 	<ul style="list-style-type: none"> • look towards sound of whistle • look at speaking commentator • nod and look interested in what human commentator says • look at any people in field of play • tell MIKE what it does, sees, hears
MIKE	<ul style="list-style-type: none"> • send commentary to speaker • (optional) tell SIG where to look • tell SIG when mode changes to stoppage mode 	<ul style="list-style-type: none"> • Make the decision when to switch mode back to play mode, and inform SIG

Table 1. The roles of SIG and MIKE in play and stoppage modes

To maintain coordination between SIG and MIKE, we separated the roles of the two systems, as summarised in Table 1. At the basic level, SIG is expected to be capable of deciding for itself what it should do as part of a commentary team. However, we define the two distinct modes of *play* and *stoppage*, that affect its behaviour. In play mode, SIG concentrates on following the ball, but may also look at the source of any sounds that it detects (in our current implementation, the referee's whistle). In stoppage mode, however, SIG ignores the ball and looks around at other things in the domain. Since we didn't use human commentators for our demonstrations, SIG could not look towards them, but in future demonstrations we hope to show how SIG can react when human commentators speak, signalling its attention by nodding or tilting its head occasionally.

The responsibility of identifying the correct play mode is given to MIKE. This is actually a complicated task that can be viewed as a first step along the path to automated refereeing. With the ability to recognise whistles (using SIG's microphones) and to identify when humans enter the field of play (from the global vision system), MIKE can identify the game mode reasonably well. During our demonstrations, however, we gave a human operator the option of overriding MIKE's decisions. MIKE also has the task of refining the overall coordination of the commentary by passing SIG explicit directions to focus on specific features of domain. For example, if MIKE decides to make a comment about the level of the crowd noise, it is better for SIG to be looking at the crowd for the duration of this comment.

Overall, the interaction between MIKE, SIG and the other actors in the domain creates an unusual example of human-computer interaction. This interaction is multi-modal in both input and output. First, to perceive the world, data from 3D sound processing is integrated with visual data. Then, the commentary generated from this data is complemented by interaction with humans in the domain. Since we use a physical robot to embody the computer commentary, we

can use body language as a mode of communicating information to the audience. Effectively, each actor in the domain combines with SIG and MIKE in a social interaction that ultimately includes the audience itself as part of the event.

3 Descriptive Content: The nature of the robotic leagues

To apply MIKE — a commentator system for simulated soccer — to a game played by robots, we had to take account of the different characteristics of the real world domain. Table 2 gives an overview of these differences, the most important of which we can summarise below.

- The real robot league is surrounded by a wall.
- The ball tends to move much faster, both because the wall keeps the ball in play and because many teams have developed specialised hardware for kicking the ball hard.
- The higher ball speeds tend to result in the ball bouncing off players more often, and less actual passing and control of the ball
- The robot league has more complicated rules to deal with situations such as robots or the ball becoming stuck. Humans are allowed to enter the pitch, and there can be delays for repairs of robots.
- Robot games can have penalty kicks, with attendant lengthy pauses for the team developers to reposition robots.

When play is actually in progress, the higher ball speed means that it is more effective for MIKE to produce commentary that concentrates more on team formations and on overall indications of which team is attacking. However, to cope with long delays before penalty kicks and for repair of robots, we found that the only reliable way to produce robust commentary was to rely on a collaboration with a human counterpart.

Feature	Simulation league	Small-sized robot league
location data	complete	includes errors
orientation data	complete	includes errors
ball speed	bounded at reasonable level	can move very fast
environment	2D field without wall	3D field surrounded by wall
rules	simple refereeing almost 100% automated no penalty kicks	complicated human referee penalty kicks
time	no stoppages	potentially long stoppages
frames	varies (10 per sec produced by vision system we utilise [3])	30 per sec

Table 2. Differences between simulation and small sized robot leagues

3.1 Assisted Commentary & Episodes

Many of the events surrounding a robot soccer game are too complex to contemplate identifying automatically with today's technology. For example, a referee may show a yellow card to one of the teams, or may decide to explain a decision in English. Since MIKE cannot hope to recognise these events by itself (or even with SIG's assistance), we developed an *assisted* commentary mode for MIKE. In assisted commentary, a human works with MIKE to commentate a game. If MIKE at any time feels that it does not understand what is happening, it can pass control to the human. MIKE then expects the human to explain the situation until the game reaches a state that is easier to understand.

To do this, we extended a recently developed version of MIKE that already works with two commentators. In [6], we describe the general advantages of dividing up the explanation task between multiple agents in real-time domains, and give as an example a version of MIKE that explains soccer games with two separate voices: an *expert* and an *announcer*. We expected the general approach we used for passing commentary between two computer-controlled commentators to also hold when one of the commentators was a human. We therefore developed a simple system that MIKE could use to determine when to speak: monitor a microphone to determine when the human commentator is speaking, then keep silent unless there is a two second pause. Using simple templates such as "Over to you" and "Thanks" to pass the conversation backwards and forwards produced a reasonable effect, but during our public demonstrations we decided to fall back on the more reliable alternative of using MIKE to speak the entire commentary, but allowing the human commentator to assist MIKE at difficult points by selecting *episodes* for it to say.

The episodes that we designed for MIKE were primarily designed to fill the long pauses that occur when teams call time-outs. They can therefore be quite long. However, there are also some short examples that can be used to interrupt the commentary of a game (*e.g.*, "Please, no flash photography", or "The referee awarded a penalty"). There are 41 episodes in total, and a longer example is shown in Figure 3 (commented lines represent commands to SIG, such as pause, nod, or look at a pre-specified location, as described in the following section). A video of SIG speaking the episode of Figure 3 during a break in play at Melbourne is available from the MIKE web pages [7].

4 SIG, the Humanoid

Figure 4 and Figure 5 show the physical appearance of SIG. To give an idea of scale, the height of upper torso is approximately 70cm, and weighs 15Kg. When using SIG to commentate for RoboCup games, it is raised on a platform so that it is slightly lower than the height of an average person. This elevation gives SIG a good view of the game field (on the floor) and also exaggerates the effect of a change of focus from the pitch to other objects such as the referee, crowd, or a human commentator.

```

This seems to be a long break in the game here,
so I'm going to take the chance to advertise myself!
#<PAUSE>
I am making the commentary on this game thanks to the work
of three research groups. First, the good people at E T L and Tokyo
University in Japan gave me a voice and the ability to follow a game
of soccer and understand it. Then, the people at ERATO in Tokyo gave
me a body that can move...
#<PAUSE>
Sometimes!
#<PAUSE>
Finally, we hooked up with some cameras provided by people
at the Future University in Japan to actually watch these robot games.
#<PAUSE>
In my first incarnation in 1997, I was called MIKE,
and I could just commentate on the RoboCup simulation league.
Now I'm really glad to have this body
#<NOD>
#<PAUSE>
and to be able to talk to all of you
#<AUDIENCE_L>
about todays
#<AUDIENCE_R>
soccer game. I hope you are enjoying my commentary today.
If not, please don't tell my developers.

```

Fig. 3. Part of an episode used by MIKE

The surface of SIG is covered with urethane. Functionally, this covering reduces noise from SIG's internal mechanical and electrical components, as well as simply protecting them. This noise reduction results in a simplification of the processing for the binaural inputs. Aesthetically, FRP was chosen because it was found to be a good compromise between the functional requirements and the need to appeal (both statically and dynamically) as a believable actor in physical situations.

SIG uses both *active vision* and *active audition* [5] in the sense that it moves its body or changes camera parameters to perceive objects or sound sources better. SIG has a total of four degrees of freedom in moving its head and neck, which it manipulates using four DC motors with potentiometers. In turn, each eye has two degrees of freedom (pan and tilt), and also a focus and zoom control. For audio information, four microphones are configured as a couple of binaural microphones and a set of internal microphones, located just inside of the ear recesses. The visual and auditory processing abilities of SIG can be summarised as follows [8]:

1. The ambiguity in the sound source direction obtained by vision and audition is $\pm 1^\circ$ and $\pm 10^\circ$, respectively. Thus, visual information on a sound source is



Fig. 4. The humanoid, SIG



Fig. 5. Close-up of SIG's head

used in preference to auditory information when possible. Note that a typical human's auditory capability is about $\pm 8^\circ$.

2. When an exact direction of sound source is available, a direction-pass filter separates sounds originating from the specified direction.
3. When a sound source is out of sight or occluded, its auditory direction is used for improving visual tracking.

SIG was designed to tackle one of the main challenges of active audition: the separation of target sounds from the overall mixture of sounds reaching the robot. However, in our soccer domain, such 'sound source separation' is not the primary goal. Rather we are happy simply with the ability to identify auditory events such as the referee's whistle, and audience applause. For the demonstration in Melbourne, we implemented the following abilities:

1. Move two eyes and/or its body to track the ball or some robot player based on information from the overhead camera (this was easier than using SIG's own visual tracking system) or based on commands from MIKE specifying what SIG should do to best convey the current commentary.
2. Move two eyes and/or its body to face pre-programmed locations, such as the left or right side of the audience (see examples in Figure 3).
3. Identify when the whistle is blown (using a template of the whistle sound).
4. Nod and tilt its head based on commentary content.

Although these abilities ignore the input of the audience and human commentators (we did not use human commentators in our demonstrations) they were still enough to allow us to substantially increase the level of audience involvement in a game. For example, one of the episodes used by MIKE was "Are there any *{Team}* fans in the audience?" After directing SIG to look at the audience for a short while, MIKE would then randomly pick an assessment of the

audience reaction (*e.g.*, “Good”, “I see”, “OK”, “Much Better”). This is a very simple level of interaction, but the audience enjoyed these exchanges, even when (or, especially when) MIKE’s comments did not match the level of cheering in the audience response. This was a simple but convincing demonstration of the potential of the “SIG plus MIKE” combination to become a genuine participant in a social interaction involving a very large number of agents.

5 Related work

In terms of soccer commentary itself, we mentioned in the Introduction that all the efforts within RoboCup itself have concentrated on the simulation league. However, SOCCER, the forerunner of one of these RoboCup systems was actually designed to interpret short sections of video recordings of real soccer games [9]. To tackle the machine vision and scene interpretation problems involved in this task, SOCCER used *geometrical scene descriptions* [10] to understand the images. SOCCER also employed multimedia presentation techniques to combine text, graphics and video in its presentations of these recordings.

Our work differs from SOCCER in that our commentator has to handle *entire* matches from start to finish, rather than just short sections. This means there are many more problems of discourse continuity, management of repetition, and the handling of stoppage time within a game. Our commentator is also physically embodied and actually situated in a real soccer game, thus also increasing the number and type of domain events to comment on, and necessitating the use of both audio information and visual information to understand a game.

One novel consequence of our use of an embodied commentator is the potential it offers for social interaction with the other actors in the domain, such as the audience members, other commentators, and referee. Brooks [11], for instance, has pointed out that “robots with humanoid form are a tool for investigating and validating cognitive theories”. The SIG plus MIKE model can allow us to investigate how social interactions can be managed — and how they are perceived — in a system with a large number of agents. For instance, in terms of the actual interaction between SIG and human commentators, we could hope to extend the results reported by the developers of Waseda University’s robot that controlling gaze is a way to produce “confirmation of communication” [12]. The use of gaze can help not only during dialogues between SIG and other humans, but can also be used (as well as other forms of body language) to actively pass information to the audience. We especially hope to implement an ‘interview’ mode for our system, where SIG is placed *between* two human speakers, so that it can turn its head to face each one as they take turns to talk. This kind of interaction would be ideal for increasing the impact when a human commentator interviews one of the ‘personalities’ connected with a game, such as a developer of one of the competing teams.

6 Conclusions

We have shown how SIG and MIKE can be combined to process multiple modes of information and unify some of the many threads of a robotic soccer game into a single, multi-modal, interactive commentary. We demonstrated our implementation of this system at four separate games of the 2000 RoboCup in Melbourne, where we received favourable feedback (and made the audience laugh). There is still much work to be done to produce a genuine autonomous commentator for this domain, but our work represents a significant first step, and proves the feasibility and the value of the challenge.

References

1. E. Andrè, K. Binsted, K. Tanaka-Ishii, S. Luke, G. Herzog, and T. Rist. Three RoboCup simulation league commentator systems. *AI Magazine*, 21(1):57–66, Spring 2000.
2. K. Tanaka-Ishii, I. Noda, I. Frank, H. Nakashima, K. Hasida, and H. Matsubara. MIKE: An automatic commentary system for soccer. In *Proceedings of ICMA-98*, pages 285–292, 1998.
3. J. Akita, J. Sese, T. Saka, M. Aono, T. Kawarabayashi, and J. Nishino. Simple soccer robots with high-speed vision system based on color detection hardware. In *RoboCup-99 Workshop, Stockholm, Sweden*, pages 53–57, 1999.
4. H. Kitano, H. G. Okuno, K. Nakadai, I. Fermin, T. Sabish, Y. Nakagawa, and T. Matsui. Designing a humanoid head for robocup challenge. In *Proceedings of Agent 2000 (Agent 2000)*, page to appear, 2000.
5. K. Nakadai, T. Lourens, H. G. Okuno, and H. Kitano. Active audition for humanoid. In *Proceedings of AAAI-2000*, page to appear, Austin, TX, 2000.
6. K. Tanaka-Ishii and I. Frank. Multi-agent explanation strategies in real-time domains. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 158–165, Hong Kong, 2000.
7. MIKE web page. <http://www.etl.go.jp/etl/suiron/~ianf/Mike>.
8. H.G. Okuno, Y. Nakagawa, and H. Kitano. Integrating auditory and visual perception for robotic soccer players. In *Proceedings of International Conference on Systems, Man, and Cybernetics (SMC-99)*, volume VI, pages 744–749, Tokyo, Oct. 1999. IEEE.
9. E. Andrè, G. Herzog, and T. Rist. On the simultaneous interpretation of real world image sequences and their natural language description: The system soccer. In *Proc. of the 8th ECAI*, pages 449–454, Munich, 1988.
10. B. Neumann. Natural language description of time-varying scenes. In D.L. Waltz, editor, *Semantic Structures: Advances in Natural Language Processing*, pages 167–207. Lawrence Erlbaum, 1989. ISBN 0-89859-817-6.
11. R. A. Brooks, C. Breazeal, R. Irie, C. C. Kemp, M. Marjanović, B. Scassellati, and M. M. Williamson. Alternative essences of intelligence. In *Proceedings of AAAI-98*, pages 961–968, Madison, WI, 1998.
12. H. Kikuchi, M. Yokoyama, K. Hoashi, Y. Hidaki, T. Kobayashi, and K. Shirai. Controlling gaze of humanoid in communication with human. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 255–260, Victoria, Canada, 1998. IEEE.

Fast and Accurate Robot Vision for Vision based Motion

Pieter Jonker, Jurjen Caarls, Wouter Bokhove

Pattern Recognition Group, Department of Applied Physics,
Delft University of Technology
Lorentzweg 1, 2628 CJ delft, The Netherlands
pieter@ph.tn.tudelft.nl

Abstract. This paper describes the vision module from the soccer playing robots of the Dutch Team. Fast vision is necessary to get a close coupling with the motion software in order to allow fast turning and dribbling with the ball without loosing it. Accurate vision is necessary for the determination of the robot's position in the field and the accurate estimation of the ball position. Both fast and accurate are necessary for the goalkeeper, but also when one robot passes the ball to another. While the Dutch team has pneumatic kicking devices that allows catching a ball smoothly, fast an accurate vision is mandatory. We use lens undistortion, a new color segmentation scheme and a shape classification scheme based on linear and circular Hough transforms in regions of Interest. We use a severe calibration procedure to get very good distance and angle measurements of the known objects in the field of view of the robot. For the keeper robot we use a Linear Processor Array in SIMD mode, that is able to execute the entire robust vision algorithm within 30ms. However the same software was programmed for the other robots with a WinTV framegrabber on the on-board Pentium of the robot. With optimizing for speed we also remained within 25ms, however, omitting the circular Hough transform for the ball and processing in a separate thread the Linear Hough transforms for self-localization on lower rate of about 50msec. The angular errors at 0 °, 20 ° and 30° heading are about 0.6 °, 0.5° and 0.4° up to 4,5 meters. The distance error at 0° heading is 5% up to 3 meters.

1 Introduction

The robot that is mainly used in the Dutch Team is the Nomad Scout robot [1]. with a Pentium 233MHz running Linux. The robots are equipped with a WinTV PCI framegrabber. In this case the Pentium performs the image processing. As tracking the ball, team-mates and competitors was one of the most difficult tasks in past RoboCup matches, we equipped the goalkeeper with an IMAP-Real-time Vision System from NEC's Incubation Center [2]. The cameras used are NTSC color cameras with fixed zoom manual iris lenses with a (wide) opening angle of 94°. The IMAP-RVS is a Linear Processor Array with 256, 8 bits data processors in SIMD mode, color framegrabbing hardware and a RISC control processor on a PCI board. It is a parallel architecture specially made for real-time image processing, where a single column of an image is mapped onto one data processor. The system is programmed in

1DC, C extended for data parallel processing, simplifying to make equivalent software for robots equipped with and without IMAP. The modules with and without IMAP have the same software interface to the other modules.



Fig. 1. The IMAP-Vision System board

The software architecture of the soccer robots (Figure 2) shows three identical robots connected to each other via a communication module. The basic layer of the software consists of virtual devices. The middle layer contains the intelligent basic skills of the robot and the World Knowledge Module, in a sense the robot's memory of its past, present and its assumptions about its direct future world. Within the top level, intelligent decisions are made to adapt the robot to the changes in the game [3].

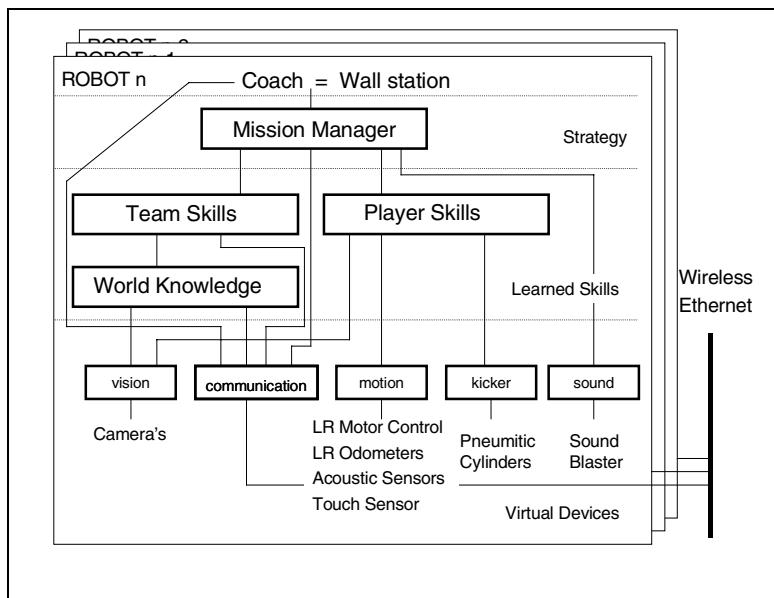


Fig. 2. Software architecture of the autonomous soccer playing robots

2. Vision for object recognition and tracking

This paper deals with the Vision Module of the robots of the Dutch Team. This module provides measurements to the Player Skills Module and the World Module and through that indirectly to the Team Skills Module. The Vision Module provides angles and distances of

known objects, like ball, goals, mates, opponents, corners, white lines and field borders. The operations that we use to realize this within the vision module are:

- Specular reflection reduction, white balancing and lens undistortion
- Segmentation in color-space; label each pixel according to YUV-information
- Search for the ball shape using a Circular Hough Transform
- Search for white lines (and wall/field-transits) using a Linear Hough Transform
- Find the corners and goal - wall transits for auto positioning using a re-segmented image

2.1 Specular reflections, white balancing, lens distortions

In order to get rid of most specular reflections a Polaroid filter is used on all cameras. To delude the auto white balance of the camera, a piece of white is visible in its VOF. Blooming is reduced by the dark filter, and the Polaroid itself takes away some of the reflections itself. Due to the wide opening angle of 94 ° of the color camera, all images are distorted. This distortion can be found by a suitable calibration method. We use two ways to circumvent the effect of the distortion: To do the image processing operations and then compensate the Regions of Interest (ROIs), features and points found for the known distortion, or to undistort the entire image and then do all subsequent operations. Using Tsai's [4] algorithm for camera calibration, as explained later, a shift in X- and Y-direction of each pixel can be calculated. These X and Y shifts are stored in Look-Up-Table images that are used to undistort the image. However for some pixels in the output image no data exist (figure 3b). Those are filled with the majority vote of neighbor data. The result is shown in figure 3c.

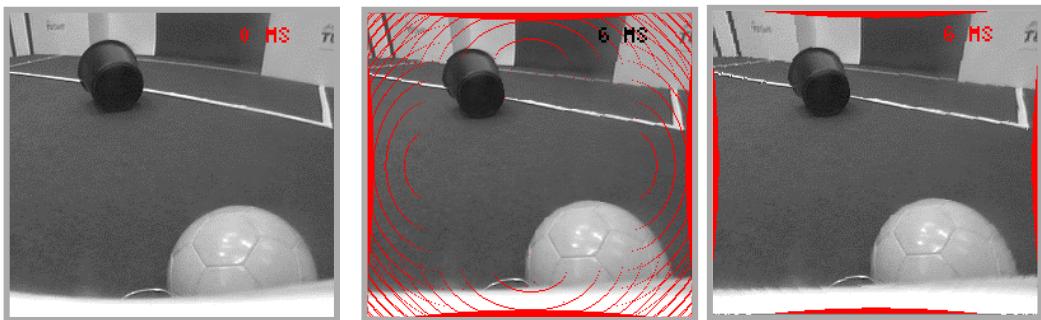


Fig. 3. Lens undistortion: a) original , b) Corrected image, c) Positions with missing data filled in.

2.2 Segmentation in color-space

For the segmentation of objects in color space we use YUV images. The benefit of the YUV space is that the UV space contains almost all color-information, meaning that a 2D image can be used to do the color segmentation. Moreover, many framegrabbers support YUV extraction in hardware. In principle the transformation from RGB to YUV is given by (Y, U, V) below,

however if the framegrabber is not supporting this, the pseudo space (Y' , U' , V') can be used, as only color differences important.

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \frac{1}{128} \begin{pmatrix} 38 & 75 & 15 \\ -22 & -42 & 64 \\ 64 & -54 & 10 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} + \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix}$$

$$\begin{pmatrix} Y' \\ U' \\ V' \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Figure 4a shows a typical scene. Figure 4b and 5 show the UV space of this scene with in greyscale overlay the intensity of the pixels found at that position in the UV space. So very white spots in e.g. the red area are reddish pixels with high intensity. Two methods can be used to do the color segmentation.

In figure 4b the method is shown that is often used, see e.g. [5], in which in U and V direction, a region of interest is made for each object to be found. This is fast but not robust.

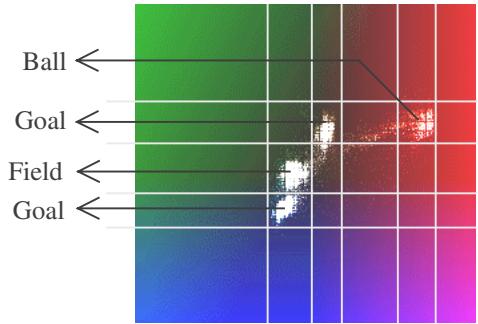
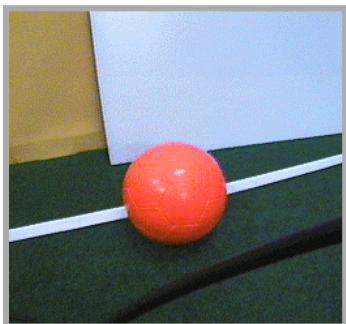


Fig. 4. a) Original typical scene, b) UV Colorspace with square ROIs to color classify objects

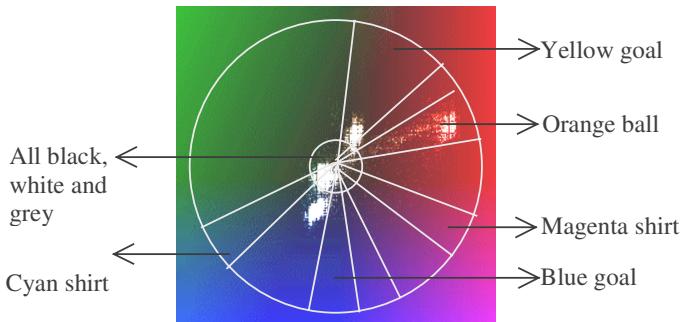


Fig. 5. Color classification method based on a simplified HSI bounding box, or pizza slices in YUV

Figure 5 shows our method, based on the observation that similar colors can be found at the same angle from the center. We simplified the original HSI colorspace, so that no multiplications are needed when represented in YUV. The majority of the pixels from the orange ball can indeed be found in the Region Of Interest spanned in Figure 4b, however the pixels in the tail also belong to the ball.

Typically, these are the pixels on top and at the bottom of the ball, where the color is less saturated due to the illumination from above and the shadow beneath. For the correct determination of the distance, e.g. based on the position where the ball hits the floor, these pixels are important. When segmenting the UV space in pizza slices, more pixels can be classified correctly and robustly to the same color, as the blob of pixels of each color class will move along the radiant during the many lighting conditions in the various game situations. In addition an inner circle is used for each color class to exclude all pixels that have a tendency too much to the grey, white, or black. For black and white objects we do an additional classification based on thresholding in the intensity image. Our color calibration procedure is based on drawing a region of interest around the object of interest in the input image, looking at its mapping in UV space and moving the two hands and inner circle and examining the result in the segmented image. This can be repeated for several play situations, until a robust segmentation is obtained. From this color calibration procedure we derive two Look-Up Tables, one for Y and another for (U,V), whereas each known object with its color class (or color segment) is assigned a bitplane in this LUT. In this way we can determine in one pass the color class of a pixel. Hence it remains possible that a pixel can be assigned to more than one colorclass. We perform:

$$\text{Class} = \text{LUT}_Y [Y] \& \text{LUT}_{UV} [U, V], \quad \text{with } \& \text{ a bitwise logic AND}$$

Per image from the grabber we first classify all pixels in this way. During this classification we form on the fly a projection of the classes on that column (i) and row (j) of the image:

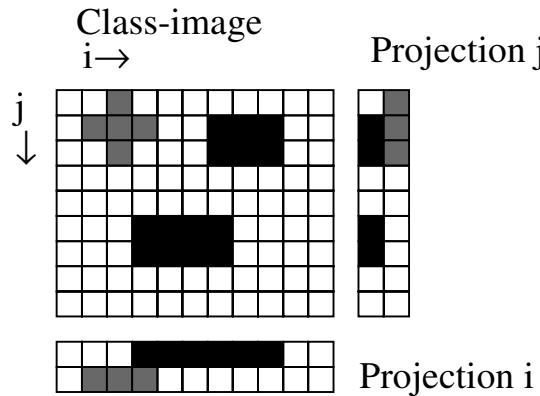


Fig. 6. Fast pixel classification procedure allowing initially multiple class labels

Then to detect each object separately, we use subsequent ROI passes in which we consider the projection boundaries as boundaries of a Region of Interest that we investigate. E.g. a red object on $j=(0-2)$ and $i=(1-3)$, a black object on $j=(1-2)$ $i=(3-8)$ and / or $j=(5-6)$ $i=(3-8)$. For each found ROI we re-project in order to separate objects until we are done. This results in bounding boxes around each object of a certain color class.

In order to be more robust against noise, we count in the ROI projections per row and column the pixels belonging to a certain color class. E.g. for a row we perform:

$$count_j = \sum_i (class_image[i, j] \text{ has } class_bit == 1)$$

Only the columns and rows that exceed a certain threshold are considered further. Moreover, we use minimum values for ROI sizes to filter out small noise objects. From the bounding boxes found the point at the bottom in the middle is used to determine the distance (this point is expected to be on the ground) and the middle-points of the side edges are used to calculate the angle of the object, using transformation formulas with calibrated parameters.

2.3 Searching and tracking the ball using a circular Hough Transform

The ball is found by fitting a circle through the edges of the orange blob, which is found during color segmentation. This feature makes it possible to estimate the correct position of the ball even if it is up to 70% occluded by other robots. Fitting a circle is done using a Circular Hough Transform in three dimensions [6]. This tries to find the three parameters by which a circle is determined: the center of the circle (X_C , Y_C) and its radius R :

$$(x - X_C)^2 + (y - Y_C)^2 = R^2$$

The 2D Circular Hough Transform uses a fixed search-radius and tries to find the center of a circle with this radius. Therefore only a 2D Hough space is enough to find the correct circle. We use a Sobel edge detector on the orange blob, which also provides directional information. This is used to allow votes only to the X_C and Y_C in the direction of the edge. As all edge-points vote to the correct center, a peak-detection-filter will find the correct center and hence the correct circle. The value of the fixed radius is determined by a recursive loop. The first time an orange blob is located in the image, using the calibration as described below, a radius of the ball for each position in the image can be found. This is used as start value for the recursive loop, which in general only needs one or two iterations.

2.4 Self-localization

For self-localization of the robot fixed markers are needed. The white lines and the wall/field transition in the field are good candidates. For a very fast Hough Transform we can't use all the points in the image. So in a pass from the top to the bottom of the image we look for white \leftrightarrow non-white transitions. We select the first and last 2 points per column resulting in $4*320=1280$ points to be transformed. The camera image suffers from lens distortion, so straight lines in the Field of View of the camera aren't straight in the image. Therefore we first transform all points using the formulas that define a virtual camera as described in 2.5. With all those points in Hough Space we search for peaks. Those peaks mark the lines that can be seen in the image. Note that here the IMAP Linear Processor Array and the WinTV grabber implementations differ in the sense that lens distortions are only compensated in the WinTV version for the image lines that are needed, whereas in the IMAP-RVS the whole image is undistorted. To find the corners and goals in an easy way, we have to use coordinates that are corrected for the camera tilt. To be fast, we only transform the first 20 lines of the image, as

the horizon lies within this area. With a histogram on the horizontal axis of the image, we can look for transitions from white to green and vice versa. The angle to those transition lines can be found. For an accurate repositioning algorithm for the keeper we need as many transition as can be seen. So we find each corner pole pair, as well as the wall-goal, goal-wall transitions.

2.5 Calibration procedure for lens distortion, angle and distance.

For the calibration of the lens distortion parameters we use the algorithm of Tsai [4]. First we go from pixel-coordinates (X_i, Y_i) to image-coordinates in mm: (X_d, Y_d), d from distorted.

$$\begin{aligned} X_d &= d \cdot x \cdot (X_i - C_x) && \text{with} & d \cdot x &= d_x \frac{N_{cx}}{N_{fx}} \\ Y_d &= d \cdot y \cdot (Y_i - C_y) \end{aligned} \quad (1)$$

(X_d, Y_d) : coordinates in mm of the distorted CCD image plane. (0,0) is the optical axis
 d_x center to center distance between adjacent CCD sensors in the scanning direction X
 d_y center to center distance between adjacent CCD sensors in the Y direction; double this value when capturing only half of the video-frame.

N_{cx} number of sensor elements in the x direction

N_{fx} number of pixels in each line as they are acquired by the framegrabber.

(C_x, C_y) : center of the lens distortion in pixels.

We only take the quadratic distortion into account, to undistort:

$$\begin{aligned} X_u &= (1 + k_1 r^2) X_d && \text{with} & r^2 &= X_d^2 + Y_d^2 \\ Y_u &= (1 + k_1 r^2) Y_d \end{aligned}$$

The Tsai algorithm now calibrates C_x, C_y and k_1 . For this we use a calibration pattern as described below. The results are used to shift the pixels as explained above.

The camera on the robot has an angle of 30 degrees with the floor. To ease the calibration of the distance we first transform the image such that the resulting image is the image if the camera (C) would look straight ahead. We call this the virtual camera (VC) image. This is done by projection of the 2D image of C onto the image of VC. Coordinates in VC are expressed as (X_v, Y_v). Assuming $Y_u=0$ and $Y_v=0$ on the horizon H (see figure 7), we obtain:

$$Z_H = Z_Q = \frac{f}{\cos(\varphi)} \quad Z_P = \frac{f}{\cos(\varphi)} - Y_{uP} \cdot \sin(\varphi) \quad Y_{vP} = Y_u \cdot \cos(\varphi)$$

With:

$$\frac{Y_{vQ}}{Y_{vP}} = \frac{Z_Q}{Z_P} \Leftrightarrow Y_{vQ} = \frac{Y_u \cdot \cos(\varphi) \cdot \frac{f}{\cos(\varphi)}}{\frac{f}{\cos(\varphi)} - Y_{uP} \cdot \sin(\varphi)} = \frac{Y_u \cdot \cos(\varphi)}{1 - Y_{uP} \cdot \frac{\sin(\varphi) \cdot \cos(\varphi)}{f}}$$

For Xv_Q we have:

$$Xv_Q = \frac{Xu}{1 - Yu_p \cdot \frac{\sin(\varphi) \cdot \cos(\varphi)}{f}}$$

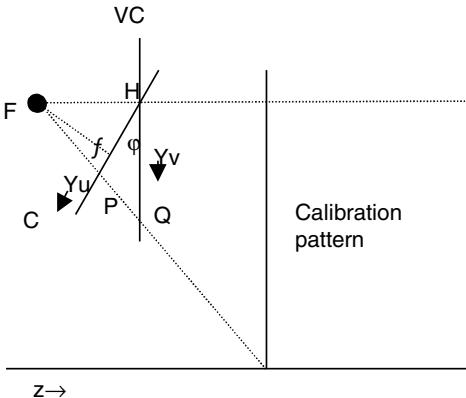


Fig. 7. Position of the real camera C en virtual camera VC in the world. H is the intersection of the horizon with the cameras. P and Q are the projections of the bottom side of the pattern onto C en VC respectively. f is the perpendicular distance of the focus point F to camera C.

These formulas are linear in the denominator and we can calibrate this using the camera on the robot and a dot pattern perpendicular to the floor, so that the VC should view a square grid of dots. With the known coordinates of the dots in the real world, the found coordinates in C and the knowledge that the dots form a regular grid, a least squares fit is used to find the denominator. Yv keeps an unknown scaling factor $\cos(\varphi)$, and if Yu_h and Yv_h are not equal 0, an extra scale factor in X and Y will appear and hence Y will obtain an additional shift. However, these linear transformations in Xv and Yv are of no concern in our additional calibration procedure! After transformation the image will look like Figure 8.

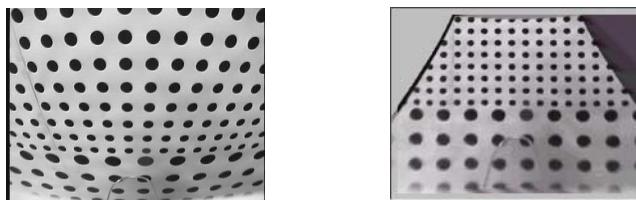


Fig. 8. Effect of calibrating for lens distortion and transformation to VC. The right image has a rectangular grid. It is not square because of the scaling $\cos(\varphi)$ in Yv

With the coordinates in VC (Xv, Yv) we are now able to calibrate the angle.

The formula for $\tan(\varphi)$, taken into account the unknown linear transformation from (X_v, Y_v) to world coordinates in the tilt-calibration, as can be deduced from figure 9, now is:

$$\tan(\phi) = \frac{X_p}{Z_p - Z_f} = \frac{X_Q}{f} = \frac{a \cdot Xv_Q + b}{f}$$

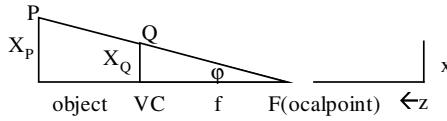


Fig. 9. Using these lines one can find the formulas for $\tan(\phi)$. f is the focal length. (x,z) are world coordinates

Using the known calibration points, this is again a linear formula, that can be found using a least squares method. Z_f can be found in a distance calibration procedure as described below. However, if $Z_p \gg Z_f$ then Z_f can be neglected.

The virtual camera concept can be used very well for the calibration of the distance. The formula that gives the relation between the distance and Yv is almost linear:

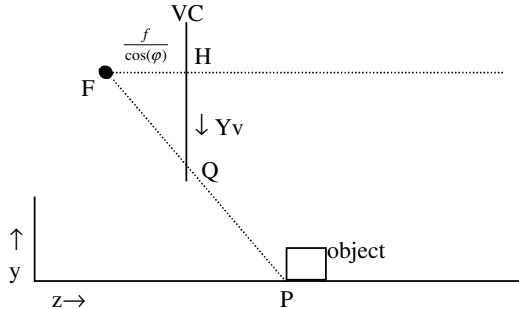


Fig. 10. f is the focal-length of C. (y,z) are world coordinates with a zero point on the floor

$$\frac{Z_p - Z_f}{Z_Q - Z_f} = \frac{Y_F}{a \cdot Yv_Q + b} \Leftrightarrow Z_p = Z_f + \frac{\frac{f}{\cos(\phi)} \cdot Y_F}{a} \cdot \frac{1}{Yv_Q + b},$$

This formula is only nonlinear in b' . The calibration with known points is done as follows:

- Choose a b' . Calculate with least squares the best fit. Do the same with $b'' = b' \pm h$.
- Choose the b'' with the least residual squared error, and step in that direction.

Use step halving on h , until the best b' is found. The distance now is:

$$Z = Z_{F,best} + a_{best} \cdot \frac{1}{Yv + b'_{best}}$$

The found $Z_{F,best}$ can be used in the angle calibration.

3 Speed and Accuracy

The Pentium 200MHz with IMAP and WinTV (foreground and background) vision tasks taking 256x240 and 320x240 images respectively:

Image Processing Routine	IMAP-RVS	WinTV (fg)	WinTV (bg)
Undistortion of the lens	6.5 ms		
Color classification of all objects	7.7 ms	10 ms	
Circular Hough Transform for the ball	3.9 ms		50 ms
Find objects (WinTV + lens undistortion)	2.6 ms	6 ms	
Linear Hough Transform for field lines	4.2 ms		50 ms
Find accurate corners and goals	2.1 ms	2 ms	
Total	27 ms	18 ms	100 ms

The Pentium load using respectively WinTV / IMAP is 27% / 3% at 15 / 30 frames/sec.

We measured that at:	1.5 m	3 m	4.5 m
The angular error at view angle 0 ° is:	0.0 °	0.6 °	1.0 °
The angular error at view angle + 20 ° is:	-0.3 °	0.0 °	0.5 °
The angular error at view angle + 30 ° is:	-0.8 °	0.4 °	-0.1 °

The distance error at	0.25 m	0.5 m	1m	1.5m	2m	2.5m	3m
and view angle 0 ° is:	-2 cm	-2 mm	1 cm	6 cm	-4 cm	-2.5 cm	+6 cm

This work was partially sponsored by the Dutch Foundation for Pure Research NWO.

4 References

- [1] <http://www.robots.com/nsuperscout.htm>
- [2] Y. Fujita et. al. 'A 10 GIPS SIMD Processor for PC based Real-Time Vision Applications', Proc. of the IEEE int. Workshop on Computer Architectures for Machine Perception (CAMP'97), pp22-32, Cambridge, MA, USA, 1997
- [3] J. Lubbers, R.R. Spaans, E.P.M. Corten, F.C.A. Groen, 'AIACS: A Robotic Soccer Team Using the Priority/Confidence Model', in 'Proceedings Xth Netherlands/Belgium conference on AI', 19 November 1998, p. 127-135.
- [4] Roger Y. Tsai "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses". IEEE Journal of Robotics and Automation, Vol. RA-3, NO. 4, August 1987
- [5] J.Bruce, T. Balch, M. Veloso, "Fast Color Image Segmentation using Commodity Hardware" see: <http://parrotfish.coral.cs.cmu.edu/fastvision/>
- [6] E.R.Davies, "Machine Vision: Theory, Algorithms, Practicalities" Academic Press, 1997. ISBN 01-12-206092-X
- [7] O. Faugeras, "Three-Dimensional Computer Vision", The MIT Press, 1996, ISBN 0-262-06158-9

Design of RoboCup-Rescue Viewers

– Towards a Real World Emergency System –

Yoshitaka Kuwata¹ and Atsushi Shinjoh²

¹ NTT DATA CORPORATION , JAPAN.

E-mail: kuwatay@nttdata.co.jp

² International Academy of Media Arts and Sciences, JAPAN.

E-mail: kaminari@iamas.ac.jp

Abstract. In this paper the design and implementation of a prototype viewer system for the RoboCup-Rescue framework (Version 0) is described. We discuss the requirements for the visualization of disaster information and propose a workflow model for the disaster mitigation process. A prototype of disaster information system has been built, based on this model, and here we discuss the system from the viewpoint of information visualizations.

1 Introduction

The RoboCup-Rescue project [3] was started by researchers in fields such as robotics and artificial intelligence. It is a grand challenge for disaster mitigation; the final goal is to build teams of rescue robots for disaster mitigation, and autonomous robot rescue teams are expected to be cooperating with human rescue teams by the year 2050. The project is divided into four parts: a simulation project, a rescue robot project, a system integration project, and an operation project. The goal of the simulation project is to design a set of software agents which model the various actors in the disaster mitigation process (rescue teams, fire-fighters, police, supervisors in fire department, and so forth.) Many kinds of simulators should be integrated as the bases of a multi-agent framework. For example, a traffic simulator, a fire-spreading simulator, and a building-collapse simulator, for example, should be working together. A prototype RoboCup-Rescue framework, called version 0, is under development by the RoboCup-Rescue technical committee. It is currently public to the RoboCup-Rescue community. The first RoboCup-Rescue competition is scheduled in August, 2001.

The importance of collecting disaster information and using it effectively was widely recognized in the aftermath of the great Hanshin-Awaji earthquake. In the early stages of the disaster mitigation process, the local emergency office could not control the rescue teams efficiently because the office did not have adequate information. The Japanese government now assists the local governments in the preparation of disaster information systems. [2]

The technologies and strategies developed in the RoboCup-Rescue framework are expected to be used in practical applications of disaster information systems in local emergency offices. Visualization techniques for disaster information are one of the most important parts of the mitigation systems on which humans rely.

In this paper we describe the design of RoboCup-Rescue (Version 0) and the architecture of the viewers as well as the requirements for the visualization of disaster information. Our analysis of the requirements is based on interviews of rescue teams and supervisors in emergency offices. This is because these requirements depend on the workflow of the mitigation process. A prototype system for the visualization of disaster information system has been based on this analysis.

2 Architecture of RoboCup-Rescue Framework

The architecture of RoboCup-Rescue framework (version 0) is described in detail in the simulator manual [5]. Figure 1 represents the architecture of RoboCup-Rescue framework in brief.

The architecture is based on the design of RoboCup-Soccer simulator. Thus, they are very similar except a few points; simulators and a Geographic Information System (GIS). As RoboCup-Rescue framework need to cooperate with many kinds of physical simulators which will be developed independently, protocols to exchange information between simulators and agents are defined. As geographic information is essential for both simulators and agents in this domain, a separate GIS is introduced.

The core of the framework is a kernel, that manages the status of the simulation world. It also takes charge of the communications between simulators and agents. A GIS component provides basic data such as the locations of buildings and road networks. It also provides the initial status of each agent. Several independent simulators, as well as several kinds of agents, are connected to the kernel. A traffic simulator, a fire-spreading simulator, a building-collapse simulator, a road-blockade simulator are currently available. One important goal for RoboCup-Rescue framework is to design a integrated disaster simulator by combining several independent simulators.

The other important goal is the modeling and design of agents. For example, action plans for rescue teams are developed and built into agents and tested in the framework. Good action plans are useful in practice. There are four kinds of agents exist in the RoboCup-Rescue framework. Fire brigade agents, police agents and rescue agents are introduced for the emergency response. Civilian agents represent victims.

The expected uses of the RoboCup-Rescue framework are the following:

1. Development of mitigation plans

The mitigation plans should be made before disasters occur. With RoboCup-Rescue simulators, specialists can check how their mitigation plan works. Urban planners, for instance, can run the simulators with or without fireproof buildings in their new city.

2. Disaster mitigation process

Viewers in emergency offices and support of rescue teams' communications are useful. We will discuss this issue in section 5 in details.

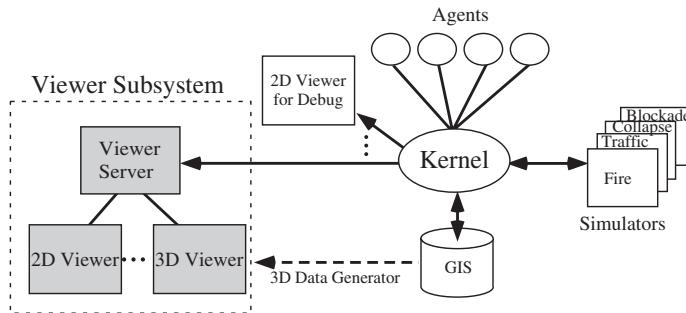


Fig. 1. An Architecture of RoboCup-Rescue Framework

3. Development of the simulation systems

Viewers are used in the development of simulators and agents. The designers of agent programs use viewers to verify their strategies. Researchers of disaster try their simulators with other simulators in order to check the complex results.

Detailed information is required for these purposes. For instance, the designer of rescue agents need to know exactly why their agents failed to rescue people in a burning house.

4. Demonstration

As in RoboCup-Soccer, RoboCup-Rescue competitions are planned. The main viewer (a 3D viewer) will be the most important component. It will be used to show the progress of the rescue competitions to the audience. It is very hard for audience to understand the progress and result of simulations. Thus, visualization should be intuitive, easy to understand, and attractive.

Two kinds of viewers are currently under development. One is a plain 2D viewer designed for researchers and developers. The other is a 3D viewer suitable for supervisors at emergency offices. The 3D viewer, also called the main viewer, provides a perspective view of the simulation world.

3 Design of Viewers for the RoboCup-Rescue Framework

3.1 3D main viewer

Like the main viewer in the RoboCup-Soccer Simulator, the main viewer for the RoboCup-Rescue framework (version 0) is based on client-server architecture. The server manages the information displayed in the clients according to the capability of the clients and the capacity of communication channel. Clients can thus be located on remote machines in distant locations. The viewer has an automated viewpoint movement built in, so that view points jump to the places where interesting events occur.

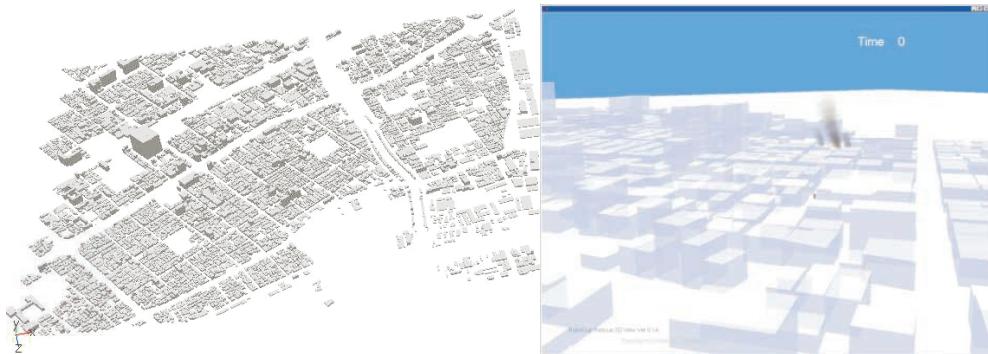


Fig. 2. Snapshot of a 3D data image of part of Nagata-Ward, Kobe(left) and a snapshot of 3D viewer(right) : The area shown is about 1.5 km by 1.5 km. As there are about 10,000 building in the area, the 3D viewer need to handle more than 50,000 polygons. Buildings are displayed as transparent objects in 3D viewer.

As the GIS data is extremely large, we supply it to a 3D viewer client directly, not through the kernel; static information such as the shapes of buildings and roads are extracted from GIS and converted into 3D objects beforehand. On the other hand, dynamic information such as location of agents and the status of roads and buildings, are supplied by the kernel.

Figure 2 is a snapshot of a 3D data image of part of Nagata-ward, Kobe-area³.

3.2 2D viewer

The 2D viewer, we designed for development support, can be used as a viewer which connects to the kernel via a TCP/IP socket. It is also a log viewer that can be used to read the kernel execution log file and redisplay the simulation. This is useful in analyzing the performance of simulators and agents. As the 2D viewer is written in Java, it can be executed on any machine which supports Java virtual machine(JVM).

A snapshot of 2D viewer is shown as figure 3. The main window of the 2D viewer is a map with objects such as buildings, roads, and agents. It has zoom in/out, and scroll functions. The status of buildings and roads are displayed as different colors; i.e. yellow for heating houses, red for burning houses, and so on. Agents are represented as small icons. The objects are mouse clickable; when users click objects of the map, the detailed information associated with the objects is displayed in the information area.

³ Kobe is selected for the first trial mainly because we have a lot of real data from the great earthquake, which can be used to compare with the results of the simulation

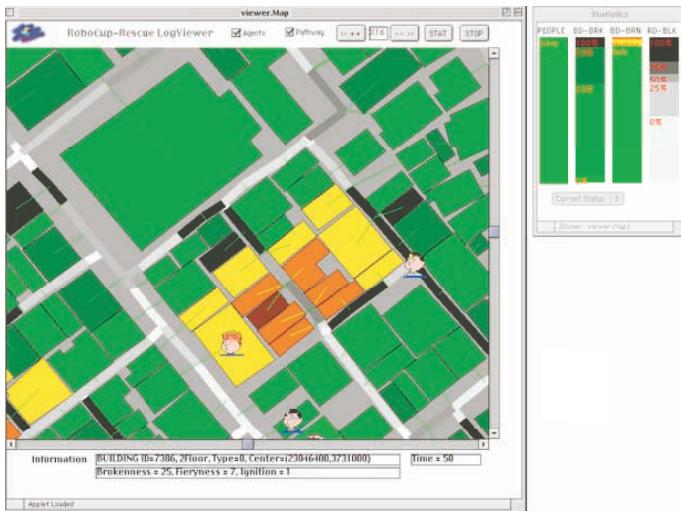


Fig. 3. Snapshot of the 2D Viewer

A small window on the top right corner is used to display statistic information. The percentage of life/death, building corruption, road blockade, and the status of fire spread are presented by small bar graphs with color.

4 Model of Workflow in the Mitigation Process

To apply the RoboCup-Rescue framework to the real world rescue operations, we started building a workflow model⁴ of ideal mitigation process for the next-generation disaster response system. We based this model on the results obtained by several rescue teams and supervisors. The model is shown in Figure 4. The workflow model is divided into four steps: (1) Data Acquisition, (2) Data Analysis, (3) Decision Making, and (4) Command-and-Control. The process starts with the collection of disaster information. Perspectives views from cameras on helicopters and from surveillance cameras on the tops of tall buildings are used for collecting data.

To understand the situation quickly, we can use image analysis in step 2. In step 3, human supervisors make decisions based on the extracted information which includes the type of the disaster, the location and scale of the affected area, and the severity of the damage. In step 4, the rescue teams in the affected areas are informed of these decisions by way of the command-and-control support system. The rescue teams also collect information in the affected areas and send reports to the emergency office. Thus these steps makes a workflow process loop.

⁴ By “workflow”, we mean a series of operations which required in mitigation process.

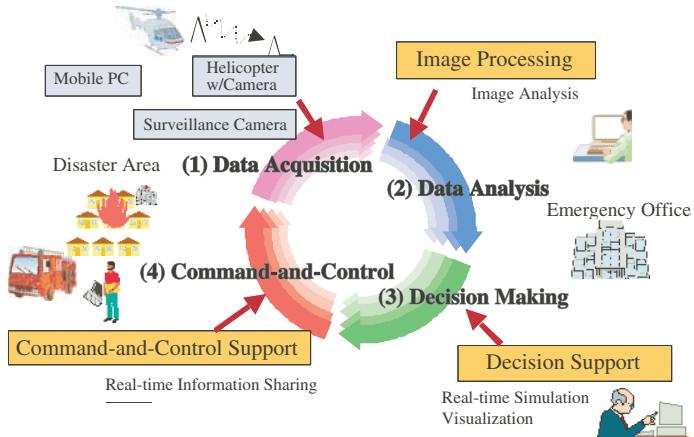


Fig. 4. Workflow model of the mitigation process.

The techniques and strategies developed in RoboCup-Rescue simulator will be used in both the decision-making process and the command-and-control process. At the emergency office, visualization is used to help supervisors make decisions. It is also used by rescue teams for sharing data within the teams and with emergency office.

The prototype systems consists of four components that corresponds to the steps in the workflow process. Figure 5 represents the decision-support system and command-and-control system which support step (3) and step (4).

The decision-support system is used by supervisors at an emergency office, and the command-and-control systems are for rescue teams. These two kinds of systems are designed to work together; all of the information is managed with location and time. A change of a piece of information in one system is transmitted to the other systems. If a member of a rescue team with a command-and-control terminal changes his location, for instance, other members of rescue team as well as the supervisors at the emergency office know it by using these systems.

Because the roles of the emergency office and rescue teams are different, their visualization requirements are different. In the following sections, we discuss two kinds of requirements.

5 Visualization Requirements

5.1 Visualization for Resource Management

The management of rescue resources is the most important of the emergency office's responsibilities. Supervisors in emergency offices need to assign rescue resources to each affected area. In the daily routine work, for example, they dispatch fire engines, ambulances, and rescue teams to areas in response to phone

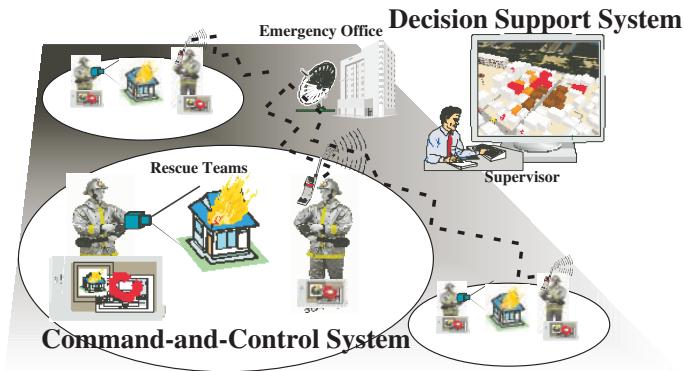


Fig. 5. Prototype system for decision support and for command-and-control: For decision support at an emergency office, 3D visualization is used to gain a quick understanding of the situation, for command-and-control support, wearable systems with touch-panel displays, CCD video cameras, wireless communication channels, and map-based shared white boards are integrated for communication support.

reports. The process becomes more difficult in the case of a big disaster, for which not enough rescue resources are available and the priority of each case need to be determined.

The following requirement list is obtained from the interview of resource managers.

- Intuitive Understanding of Global Information

Because supervisors need to manage a big area, such as a whole city, their decisions must be based on a global view of resources and damage.

- Estimation and Prediction of Damage

In the early stages of mitigation efforts, almost no information is available. Emergency offices thus need to estimate the damage for the decision on their first actions. The prediction of damage is also important for the middle term (around 72 hours) planning in big disasters.

- Trial-and-error simulation

There are two phases of decision support, and the first is to make mitigation plans beforehand. Supervisors can base a decision on these mitigation plans. Building mitigation plans require trial-and-error simulation. Because the visualization of disaster response systems will be used in both phases, it also needs to support trial-and-error.

- Real-time Visualization

In the execution of mitigation plans, live information is collected and should be displayed. Intuitive and real-time visualization is required.

Based on the list, we built a prototype decision-support system with the following functions.

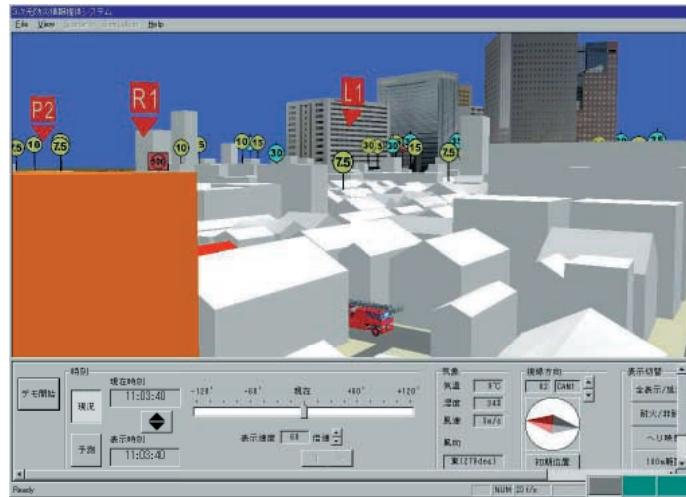


Fig. 6. A 3D perspective view in the prototype decision-support system

1. Real-time 3D visualization

Real-time 3D graphics are used. Users can select any place and any viewpoint in the 3D space. The location, scale, and degree of affected area are shown, as well as rescue resource information such as the locations of rescue teams, fire engines, hydrants.

2. Fire-spreading simulation

A real-time fire spreading simulator with physical model [6] is integrated in this system. The prediction of fire-spreading can also be displayed in the same 3D graphics. By adjusting the time-dial, the system can show the status of fire spreading in any time.

3. Integrated interface with GIS

The system is based on a GIS: The 3D objects are made from geographic data and managed with structured GIS database. We can show the detailed information of a 3D object that is pointed out on a screen.

Figure 6 represents a snapshot of a 3D visualization in our decision-support system.

5.2 Visualization for Communication Support

We assume the rescue teams use communication systems, and we propose to supply the following functions for rescue teams.

1. Real-time map-sharing

Because almost all disaster information is related to the locations, we can use a GIS for managing the data. For example, a photograph taken by a rescue

team member is stored with the location. A real-time map-sharing mechanism provides a shared electric map with dynamic object on it. In shared map, a picture icon appears at the location so that all of the other members can know the availability of the picture. Hand writings and locations of members are also dynamic objects.

2. Visual data acquisition

The field commanders of rescue teams need to know the conditions of the disaster area in detail when they plan their actions. Moving pictures are useful in helping them quickly understand the status of the disaster area. Even in the same disaster area, it is impossible to know all of the status which is essential for decision making. For example, commanders need to know the situations on the other sides of burning houses.

3. Support of multi-modal communication

Walkie-talkies are widely used by rescue teams for communication both within the disaster area and between the area and emergency office. In general, it is very hard to report the conditions of the disaster area and status of a rescue teams' actions via voice links. Using a combination of handwriting-gestures and voice helps. For example, a member of a rescue team can point to one area in the shared map and report vocally that there are burning houses in that area.

4. Safety and security monitoring

The safety and security of rescue teams is also of great importance to field commanders. The commanders always need to pay close attention to the status of rescue team members. It is impossible, however, for most commanders to monitor all of rescue team members while the commanders are planning the next action. Life sensors enable the commanders to monitor the rescue team members on-line.

Because viewers are used as communication tools in disaster areas, they should be easy to use. Figure 7 is a snapshot of a prototype command-and-control system for rescue teams. The system has all of the function except safety and security monitoring.

Another design for a wearable system is described in Reference [8].

6 Conclusions

This paper has proposed a process workflow model for disaster mitigation and has based a discussion the requirements for the visualization of disaster information. A prototype system with visualization is built for the evaluation. The design and implementation of RoboCup-Rescue simulator viewers is also described.

We would like to apply knowledge from RoboCup-Rescue simulation system in practice, and we are interested in the evaluation of the visualization techniques we proposed. This evaluation will require the use of field trials.

We are planning to integrate the prototype system and the RoboCup-Rescue simulator.



Fig. 7. An outfit of the command-and-control system (left), and a screen image of the system (right): A map-based shared white board is shown with city map of Kobe. Dynamic objects such as fire engines, ambulances, and rescue team members are represented as small icons on the map. A rough circle and a arrow in the center of screen are handwritten. The small window in the top right corner is part of a remote conference system used to share live images.

References

1. Paul R. Cohen, Michael L. Greenberg, David M. Hart and Adele E. Howe, *Trial by Fire: Understanding the Design Requirements for Agents in Complex Environments*, AI Magazine Vol. 10, No.3 Pp.32-48, Fall, 1989.
2. Edited by Administrative Inspection Bureau, at Management and Coordination Agency(MCA), *Towards the perfect anti-earthquake procedure – Based on a study of the Hanshin-Awaji earthquake – (in Japanese)*, Printing Bureau, Ministry of Finance, April 6, 1998.
3. Hiroaki Kitano et al., *RoboCup-Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research*, Proc. IEEE SMC, 1999.
4. Hiroaki Kitano et al., *RoboCup-Rescue – Grand challenge for disaster mitigation (in Japanese)*, Kyoritsu-shuppan, May, 2000
5. The RoboCup-Rescue Technical Committee, *RoboCup-Rescue Simulator Manual (Version 0 Revision 2)*, The RoboCup Federation, 2000.
6. Yuji Ishikawa et al., *RoboCup-Rescue Project (7th report) Fire Spreading Simulation based on Physical Models (in Japanese)*, Proc. 60th IPSJ Conference, Mar., 2000.
7. Yoshitaka Kuwata et al., *RoboCup-Rescue Project (8th report) Collection and Visualization of Disaster Information (in Japanese)*, Proc. 60th IPSJ Conference, Mar., 2000.
8. Atsushi Shinjoh et al., *Wearable system for the Disaster Mitigation Problem: Mission Critical Man-Machine Interface for RoboCup-Rescue simulator*, Proc. 9th Int. Conf. on Artificial Reality and Telexistence (ICAT'99), Dec., 1999.

From Multiple Images to a Consistent View

R. Hanek, T. Schmitt, M. Klupsch, and S. Buck

AGILO RoboCuppers, <http://www9.in.tum.de/agilo>
Technische Universität München, Germany

Abstract

The approach presented in this paper allows a team of mobile robots to estimate cooperatively their poses, i.e. positions and orientations, and the poses of other observed objects from images. The images are obtained by calibrated color cameras mounted on the robots. Model knowledge of the robots' environment, the geometry of observed objects, and the characteristics of the cameras are represented in curve functions which describe the relation between model curves in the image and the sought pose parameters. The pose parameters are estimated by minimizing the distance between model curves and actual image curves. Observations from possibly different view points obtained at different times are fused by a method similar to the extended Kalman filter. In contrast to the extended Kalman filter, which is based on a linear approximation of the measurement equations, we use an iterative optimization technique which takes non-linearities into account. The approach has been successfully used in robot soccer, where it reliably maintained a joint pose estimate for the players and the ball.

1 Introduction

1.1 Motivation and Goal of the Work

To successfully perform their tasks, most autonomous mobile robots must be able to estimate their own poses, consisting of position and orientation. Furthermore, the interaction with other robots and the manipulation of objects require them to localize other, possibly moving, objects. A strong restriction is that the localization problem has to be solved in real-time.

Often the required localization accuracy varies with the distance of the object to be localized. A robot that wants to grasp an object requires an accurate position whereas less accurate estimates are sufficient to approach the object. Due to the lower price and weight, visual sensors are often preferred against laser range finders. In this paper, we propose a novel approach for estimating the poses of cooperating, mobile robots and the poses of other objects observed by the robots.

1.2 Previous Work

The problem of pose-estimation from images is frequently addressed by the robotics, the computer vision, and the photogrammetry communities. Due to the huge number of publications, a comprehensive review would be beyond the scope of this paper.

Recently sample-based versions of Markov localization became very popular [2, 5, 3, 6]. Closely related to sample-based Markov localization is the CONDENSATION algorithm [1] which is often used for object tracking. Both approaches

represent the posterior distribution of the sought parameters (e.g. the pose) by samples, which allows to approximate virtually any distribution. However, in order to achieve high accuracy, usually a high number of samples is necessary. The conditional probability density of the environment observation has to be computed for each sample pose. A good match between the observation and a sample pose leads to new randomly generated sample poses in the vicinity of the good sample. However, Markov localization does not try to directly improve the already good pose. Hence, Markov localization leads to limited accuracy and/or relatively high computational cost.

Usually vision-based pose estimation methods consists of three steps. First, features are extracted from the image. Second, correspondences are established between image features and model features. In the third step, the pose is estimated using relations between image features and model features. Image features are for example points [10, 8, 7], lines [7, 10, 9], or ellipses [12]. Most methods are only applicable to specific feature types (e.g. lines, circles, ellipses), whereas our method can be applied to all types of smooth curves. We use model knowledge of the robots' environment in order to restrict the search for features and correspondences. Thus, only local operations on the image are necessary, which significantly increases processing speed.

2 Self-Localization from a Single View

In this section, we address the problem of estimating the pose of a robot in 3D world coordinates from a single image grabbed by a calibrated camera mounted on the robot. In the following, the robot pose is estimated by minimizing the distances between observed curves in the image and 2D projections of curves from the 3D world model.

2.1 Specification of Model Curves

The geometric properties of the robots' operating environment are specified by a predefined world model which consists of curve features. A 3D curve feature is given by a curve function $C_i : D_i \rightarrow \mathcal{R}^3$ defining the set $\mathcal{G}(C_i)$ of curve points in 3D world coordinates by

$$\mathcal{G}(C_i) = \{C_i(s) | s \in D_i\} \quad (1)$$

where $D_i = [s_{i,\min}, \dots, s_{i,\max}]$ is the domain for s . In practice, all relevant curves, e.g. circles or B-splines, can be specified or at least approximated by curve functions. Especially for a man-made environment a polyhedral model is often used. For such a model the curves are line segments which are given by

$$C_i(s) = s \cdot B_{i1} + (1 - s) \cdot B_{i2} \quad (2)$$

where the points B_{i1} and B_{i2} are the endpoints of the line segments and D is equivalent to $[0, \dots, 1]$. The observation of a 3D curve $\mathcal{G}(C_i)$ in the image is a 2D image curve

$$\mathcal{G}(c_i) = \{c_i(s, \Phi) | s \in D_i\} \quad (3)$$

where Φ is the robot pose and c_i is the image curve function given by

$$c_i(s, \Phi) = \text{proj}(C_i(s), \Phi) \quad (4)$$

The function proj projects a 3D point, given in world coordinates, into the image and returns the pixel coordinates of the resulting 2D point. First, the function proj converts the 3D world coordinates into 3D robot coordinates. This conversion depends on the pose Φ of the robot. The first two elements of the robot pose Φ are the x - and the y -coordinate of the robot position in world coordinates and the third element specifies the angle of the robot's orientation. Since we use a calibrated camera mounted on the robot, the 3D robot coordinates can be transformed into 3D camera coordinates and finally, the pixel coordinates of the 2D projection can be computed. The resulting image curve function c_i describes the relation between the sought robot pose Φ and the position of the model curve points in the image.

In order to obtain curve functions of occluding edges, e.g. the silhouette of a sphere or a cylinder, we first compute the tangent points in 3D world coordinates and project them into the image.

We distinguish two types of image curves: 1.) edge curves. An edge curve separates two image regions which differ in the distribution of color vectors. An edge is specified by a single curve function. 2.) line curves. For a line curve two curve functions are used which not only describe the position of the line but also its width.

Usually, not only knowledge of the geometric properties of a curve feature is given but also knowledge of possible color distributions of the adjacent regions. This knowledge is used in 2.2 in order reduce the search space of possible correspondences between observed image curves and model curves. We assign a set of possible color vectors to each side of an edge. Similarly, for a line curve two color sets are used, one for the background color and one for the line itself.

2.2 Iterative Optimization of the Pose

From the context of the application or from previous observations usually uncertain a-priori knowledge of the robot pose Φ is given. We model the a-priori probability density $p(\Phi)$ by a multi-variate Gaussian density with mean vector $\bar{\Phi}$ and covariance matrix \mathbf{C}_Φ . In the following the mean vector $\bar{\Phi}$ is used in order to establish correspondences between image curves and model curves.

Search for points on the corresponding image curve For each projected model curve c_i , the set $\mathcal{H}(c_i)$ of initial points is computed by

$$\mathcal{H}(c_i) = \{c_i(s_{i,j}, \bar{\Phi}) | s_{i,j} \in S_i\} \quad (5)$$

where the set $S_i \subset D_i$ contains the values $s_{i,j}$ for which the curve functions are evaluated. The set S_i has to be chosen such that for all $s_{i,j} \in S_i$, the resulting image points are valid, i.e. the 3D points are in front of the camera and the projections are within the image area.

For each projected model point $P_{i,j} = c_i(s_j, \bar{\Phi}) \in \mathcal{H}(c_i)$, the tangent vector $\mathbf{v}_{i,j}$ is computed by

$$\mathbf{v}_{i,j} = \frac{\partial}{\partial a} c_i(a, \bar{\Phi})|_{a=s_{i,j}} \quad (6)$$

Image points $\tilde{P}_{i,j}$ of the corresponding image curve are sought along the perpendiculars of $\mathbf{v}_{i,j}$, see Fig. 1. In order to avoid wrong correspondences, an image

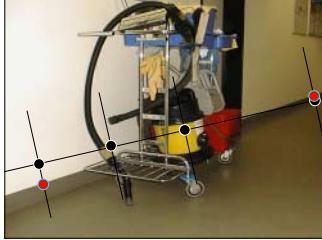


Fig. 1. Search for image points along the perpendiculars: for the two perpendiculars in the center of the image no image points are found which lie between regions of the required colors.

point $\tilde{P}_{i,j}$ on an edge is only accepted if both sides of the edge have consistent colors according to the curve model. In Fig. 1, the region below an image point has to be 'grey' and the region above has to be 'white'. Analogously, for line curves the color distributions of the line and the background are used in order to eliminate wrong image points. Furthermore, the two curve functions defining a line curve are used in order to estimate the line width in the image. A line point is rejected if the line width in the image is not similar to the line width of the projected model line. For each image point $\tilde{P}_{i,j}$, the standard deviation $\sigma_{i,j}$ is estimated which describes the probable precision of the observation $\tilde{P}_{i,j}$ along the perpendicular.

MAP-Estimation In the following, we describe how the robot pose is estimated by minimizing the distances between observed image points $\tilde{P}_{i,j}$ and projected model curves $\mathcal{G}(c_i)$. The approach presented here is a modification and generalization of [7].

In order to express the displacements between image points $\tilde{P}_{i,j}$ and projected model curves $\mathcal{G}(c_i)$ as a function of the robot pose Φ , the projected model curves $\mathcal{G}(c_i)$ are approximated in the vicinities of the projected model points $P_{i,j} = c_i(s_{i,j}, \Phi)$ by straight lines, see Fig. 2. We denote the normal vector perpendicular to $\mathbf{v}_{i,j}(\Phi)$ by $\mathbf{n}_{i,j}(\Phi)$. The displacement $d_{i,j}(\Phi)$ between an observed image point $\tilde{P}_{i,j}$ and the corresponding tangent of the projected model curve c_i is

$$d_{i,j}(\Phi) = (\mathbf{n}_{i,j}(\Phi))^T \cdot [c_i(s_{i,j}, \Phi) - \tilde{P}_{i,j}] \quad (7)$$

where $(\cdot)^T$ indicates vector transposition. If the curvature of the projected model curve is not zero and the observation $\tilde{P}_{i,j}$ is not exactly on the perpendicular then $d_{i,j}(\Phi)$ is just an approximation of the real displacement between $\tilde{P}_{i,j}$ and the projected model curve $\mathcal{G}(c_i)$.

The observations $\tilde{P}_{i,j}$ are noise-affected. We assume that for all observations $\tilde{P}_{i,j}$, the components of the noise in the direction perpendicular to the projected model curves are mutually independent and Gaussian distributed with mean value zero and standard deviation $\sigma_{i,j}$. With this assumption, the maximum

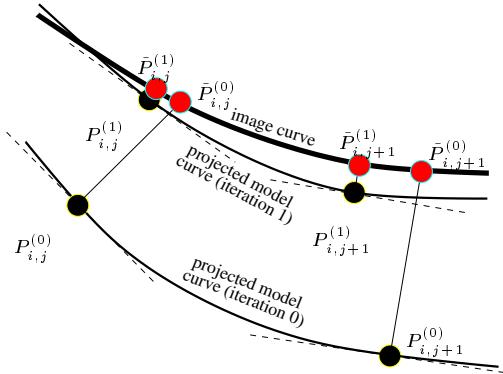


Fig. 2. One iteration step: the new projected model curve is closer to the image curve.

a-posteriori (MAP) estimation $\hat{\Phi}$ of the robot pose is given by

$$\hat{\Phi} = \underset{\Phi}{\operatorname{argmax}} p(\Phi) \cdot \prod_{(i,j)} \frac{1}{\sqrt{2\pi\sigma_{i,j}}} \exp\left(\frac{-d_{i,j}^2(\Phi)}{2\sigma_{i,j}^2}\right) \quad (8)$$

The maximization of this product can be transformed into a more convenient minimization of a sum by taking the negative logarithm:

$$\hat{\Phi} = \underset{\Phi}{\operatorname{argmin}} \chi_1^2 + \chi_2^2 \quad \text{with} \quad (9)$$

$$\chi_1^2 = C_1 (\bar{\Phi} - \Phi)^T \mathbf{C}_\Phi^{-1} (\bar{\Phi} - \Phi) \quad (10)$$

$$\chi_2^2 = C_2 \sum_{(i,j)} \frac{d_{i,j}^2(\Phi)}{2\sigma_{i,j}^2} \quad (11)$$

where $\bar{\Phi}$ is the mean vector and \mathbf{C}_Φ is the covariance matrix of the a-priori density $p(\Phi)$. The variables C_1 and C_2 do not depend on the pose Φ . In order to minimize $\chi_1^2 + \chi_2^2$, we use Newton iteration [11]. Fig. 2 illustrates the result of one iteration step. After the iteration step, the projected model curves are closer to the observed image points. However, the projected points $P_{i,j}$ on the model curves are shifted along the model curves. Due to this shift, the correct displacements between the image points $\tilde{P}_{i,j}$ and the projected model curves may differ from the estimations $d_{i,j}(\Phi)$ which are obtained by linear approximation. In order to yield precise measurements between the model curves and the image curves, new image points are sought along the perpendiculars passing through the new projected model points. Since the deviation between image curves and projected model curves is already reduced, the new search can be performed at low computational cost.

In experiments where the initialization was obtained from previous images after two or three iterations the changes of the projected model curves were less than one pixel. Since the pose consists of three variables, three independent displacements $d_{i,j}$ corresponding to three image point $\tilde{P}_{i,j}$ are sufficient to estimate the pose. However, in order to increase robustness and accuracy, for each visible feature three image points $\tilde{P}_{i,j}$ are sought.

For most applications, a rough quantification of the accuracy of $\hat{\Phi}$ is needed which enables the robot to adapt its behavior to the uncertainty of its presumed pose. In order to estimate the uncertainty of the pose, we apply a method based on Taylor approximation of the minimized term $\chi_1^2 + \chi_2^2$, see e.g. [11]. This method can be applied when the measurement errors are normally distributed and the number of image points is large enough, so that the uncertainties in the estimated parameters Φ do not extend outside a region in which the functions c_i could be replaced by suitable linearization.

3 Combined Self-Localization and Object-Localization

3.1 Localization from a Single View

On the one hand, the estimated world coordinates of the observed objects depend on the world coordinates of the observing robot. On the other hand, a priori knowledge of the poses of the observed objects can be used for the self-localization of the observing robot. In order to exploit these interdependencies, the self-localization and the localization of other objects are performed simultaneously in a single optimization.

Analogously to the self-localization we use curve functions c_i which describe the position of curve points in the image. In order to describe the appearance of an observed object, the object's features are first transformed into world coordinates and thereafter, the observations of these features are computed.

The optimization is done (as for the self-localization) simultaneously over all curves, no matter whether a curve feature belongs to the static world model or to a mobile object. For the combined minimization, the sought parameter vector Φ contains the pose of the observing robot and the poses of the observed objects.

3.2 Data Fusion Over Time

In this section, we discuss the fusion of observations made by several autonomous mobile robots into a single dynamic view of the world. This dynamic world model contains the poses of the robots and the poses of all other observed objects. Since the variety of the robots should not be restricted, we assume that the cameras of the robots are not synchronized and the frame rate may vary from camera to camera. Furthermore, the presented approach allows to use an arbitrary number of cameras mounted on a single robot. The communication between the robots takes place via a wireless LAN. Due to the wireless communication some data packages may be lost.

In order to fuse observations over time, we use a modification of the extended Kalman filter, see e.g. [4] for a description of the extended Kalman filter. For an image obtained at time t , from previous observations an estimation of the a-priori distribution of the sought parameter vector Φ is computed. As described in section 3.1, the new image is used in order to update the estimate of the parameter vector.

Our approach differs from the extended Kalman filter [4] as follows: The Kalman filter theory assumes a linear relation between the measurements in the image and the sought parameter vector, in our case the poses Φ . However, due to rotation(s), perspective projection, and radial distortions of the lens, the curve functions c_i are non-linear in Φ . The extended Kalman filter uses a first-order Taylor approximation in the vicinity of the predicted parameter vector.

The quality of this approximation depends on the distance between the predicted parameter vector and the correct vector, and on the curvatures of the displacements $d_{i,j}(\Phi)$. In contrast to the extended Kalman filter, we use Newton iteration which approximates the displacement functions in the solution of the previous iteration step. This yields a more accurate solution if the relation between the observations and the parameter vector is not linear. The computational cost for a single image is about n times larger, where n is the number of iterations. However, due to the resulting higher accuracy of the predictions, only few iterations (usually two or three) are necessary.

After a robot R_1 has updated its estimate $\hat{\Phi}$, it could broadcast the values of $\hat{\Phi}$ to other robots which could adapt it. However, this communication strategy might be error-prone. If robot R_1 has not received some data packages via the wireless LAN then it would spread out inaccurate data.

Therefore, a robot broadcasts the time t of its latest observation and a second order Taylor approximation $\tilde{\chi}_2^2(\Phi)$ of $\chi_2^2(\Phi)$ obtained for the vicinity of $\hat{\Phi}$. The function $\chi_2^2(\Phi)$, see Eq. (11), summarizes all observations obtained from the latest image. The receiving robots estimate their own predictions for time t , substitute in Eq. (9) the term $\chi_2^2(\Phi)$ by the received Taylor approximation $\tilde{\chi}_2^2(\Phi)$ and minimize $\chi_1 + \tilde{\chi}_2$ according to Eq. (9). It is noteworthy that for the minimization of $\chi_1 + \tilde{\chi}_2$ an efficiently computable closed-form solution exists.

In situations where not enough features are given, pure vision based navigation is impossible. In order to cope with these temporal cases, we combine the visual information with odometric data. The odometric data is used in order to improve the prediction of the robot's pose.

4 Experimental Results

The presented method is applied in our middle-size RoboCup team, the AGILO RoboCuppers. At present, the RoboCup scenario defines a fixed world model with field-boundaries, lines and circles (see Fig. 4). Our approach was successfully applied in 1999 during the RoboCup World Soccer Championship and the German Vision Cup. During a RoboCup match, every robot is able to process 15 to 18 frames per second with its on-board Pentium 200 MHz computer. The localization runs with a mean processing time of 18 msec for a 16-Bit RGB ($384 * 288$) image. Only for 4% of the images the processing time exceeds 25 msec.

In the first experiment, the accuracy of the self-localization is investigated. One robot is set up at six different positions (see Fig. 3) and the self-localization algorithm is run for about 30 seconds. From about 750 pose estimates, the mean error of the x -/ y -coordinates and the rotation angle are computed and displayed in Tab. 1. In general, the accuracy of the estimated pose depends on the visible features and their distance to the camera. The poses 1), 4) and 5) allow for quite accurate pose estimation. In pose 2) the visible features are far away and therefore, the resulting estimates are less accurate. The problem of pose 3) is that an error in the y -coordinate can be compensated by the rotation angle, and vice versa. From pose 6) only one edge is visible which is not sufficient in order to determine all three pose parameters. In this case data-fusion with e.g. odometric data is a necessity to overcome this problem. In areas where precise robot movements are required, e.g. near the goals, enough features for robust vision-based pose estimation are present (see also Experiment 3). An analysis of the errors shows that the variance of the estimates can be neglected. The

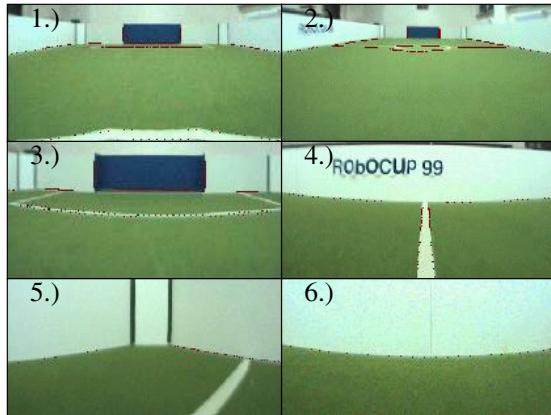


Fig. 3. Views from 6 different poses on the field

Mean pose errors	1)	2)	3)	4)	5)	6)
Φ_x (cm)	0.4	26.0	1.9	0.3	4.3	>99
Φ_y (cm)	1.0	20.3	15.5	8.0	3.2	2.3
Φ_φ (degree)	1.1	4.1	8.1	2.4	7.5	2.8

Table 1. Mean error of the self-localization for the x -, y -coordinates, and the rotation angle φ

bias of the estimation is caused by inaccuracies in the camera calibration and unevenness of the floor.

In the second experiment, the robots sensor data fusion is tested. Three robots are put at predefined positions on the field. After the robots have estimated their pose, a ball is put on the field, such that all robots can see the ball. Fig. 4 outlines the experimental setup. Tab. 2 gives a brief summary of the results. Three rows are given for every robot. The first row exhibits the robot's exact pose and the exact ball position. In the second row, the robot's pose estimate and the corresponding uncertainties are displayed. The third row shows the estimated robot and ball poses after the ball was put on the field. All

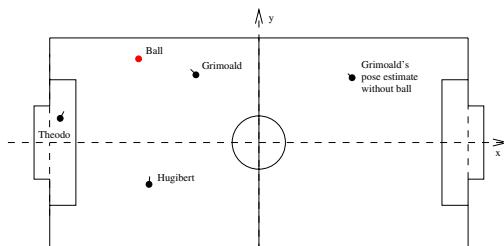


Fig. 4. Cooperative localization: Theodo and Hugibert tell Grimoald where they see the ball. Grimoald used the received ball coordinates to estimate his shift along the wall.

Robot		x	y	φ	ball x	ball y	σ_x	σ_y	σ_φ	σ_{ballx}	σ_{bally}
Theodo	ground truth	-3.80	0.45	40	-2.30	1.60	—	—	—	—	—
	estim. without ball	-3.80	0.43	39	—	—	0.0433	0.0563	1.439	∞	∞
	estim. with ball	-3.77	0.47	40	-2.26	1.63	0.0396	0.0472	1.106	.0767	.0731
Hugibert	ground truth	-2.10	-0.80	90	-2.30	1.60	—	—	—	—	—
	estim. without ball	-2.13	-0.94	92	—	—	.1130	.0924	8.738	∞	∞
	estim. with ball	-2.12	-0.78	90	-2.31	1.63	.0941	.0872	5.072	.1057	.0937
Grimoald	ground truth	-1.15	1.30	140	-2.30	1.60	—	—	—	—	—
	estim. without ball	1.77	1.25	138	—	—	∞	.0149	2.753	∞	∞
	estim. with ball	-1.14	1.27	140	-2.31	1.58	.1053	.0143	2.544	.0863	.0146

Table 2. Results of the cooperative localization experiment (See Figure 4 for the experimental setup.)

positions are given in cartesian world coordinates and are measured in meters. Orientations are measured in degrees. The robots Theodo and Hugibert estimate their poses with high accuracies. Grimoald can only detect the edge between the field and the top boundary. Thus, he estimates y and φ correctly but sees itself too far to the right. After the ball was put on the field, Grimoald receives from the other two robots the ball's position estimates and applies them to correct its own pose. This experiment was also performed several times with moving robots and equally good results.

In the third experiment, we examine the capability to fuse visual and odometric data over time. A robot starts at the top left corner of the playing field and moves along an 8-shaped trajectory across the field, see Fig. 5. Starting- and ending-point of the trajectory are the same. The four major tuning-points are at the corners of the penalty-area ($x = +/ - 3$ m, $y = +/ - 1.25$ m). The first trajectory was derived exclusively from the vision based pose estimates. The second trajectory was obtained by dead-reckoning. The third trajectory is the result of the fusion process combining data from both sensors. The weaknesses of the pure vision-based localization and dead-reckoning approach are clearly visible. For the vision-based approach, comparatively fast changes of the estimated pose may happen if the observed features change. In general this happens only when too few or too distant features are visible. This can be observed when the robot crosses the field diagonally and does not see the middle-line and center-circle anymore (see Fig. 4). The dead-reckoning trajectory exhibits error accumulation over time: starting- and ending-point of the trajectory are about 80 centimeters apart. The approach based on both data types is able to cope with both weaknesses. Vision based pose inaccuracies are compensated by the relative movements derived from the odometric data. The accumulation of the dead-reckoning error is corrected by the vision based pose estimates.

5 Summary and Conclusion

The approach presented in this paper allows mobile robots to estimate cooperatively their own poses and the poses of other observed objects from images. The approach is able to exploit the interdependencies between self-localization and the localization of other observed objects.

The method runs faster than frame-rate. This high speed is achieved by an exclusively local search for image points in the vicinity of predicted model

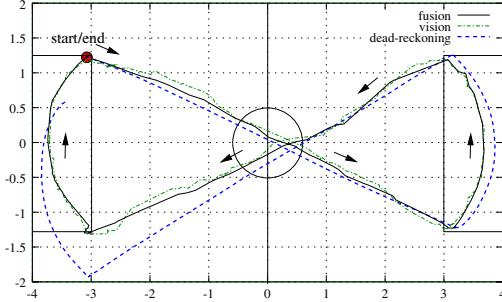


Fig. 5. Data-fusion of vision and dead-reckoning position estimates

points. The fusion over time is also performed efficiently with a technique similar to the extended Kalman filter. In contrast to the extended Kalman filter, we explicitly take the nonlinearity of the measurement equations into account. This leads to high accuracies and good predictions. We have shown that cooperative localization leads to a higher localization accuracy. Furthermore, our method allows for solving certain localization problems that are unsolvable for a single robot.

References

1. BLAKE, A., AND ISARD, M. *Active Contours*. Springer-Verlag, Berlin Heidelberg New York, 1998.
2. DELLAERT, F., BURGARD, W., FOX, D., AND THRUN, S. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition* (1999), pp. II:588–594.
3. ENDERLE, S., RITTER, M., FOX, D., S., S., AND G., K. Soccer-robot localization using sporadic visual features. In *6th Int. Conf. on Intelligent Autonomous Systems* (2000).
4. FAUGERAS, O. Three-dimensional computer vision: A geometric viewpoint. *MIT Press* (1993), 302.
5. FOX, D., BURGARD, W., DELLAERT, F., AND THRUN, S. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence* (1999).
6. FOX, D., BURGARD, W., KRUPPA, H., AND THRUN, S. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots* (2000). To appear.
7. HANEK, R., NAVAB, N., AND APPEL, M. Yet another method for pose estimation: A probabilistic approach using points, lines and cylinders. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition* (1999), pp. II:544–550.
8. HARALICK, R., JOO, H., LEE, C., ZHUANG, X., VAIDYA, V., AND KIM, M. Pose estimation from corresponding point data. *IEEE Trans. SMC 19*, 6 (November/December 1989), 1426–1446.
9. IOCCHI, L., AND D., N. Self-localization in the robocup environment. In *3rd RoboCup Workshop*, Springer-Verlag, *in press*. (1999).
10. PHONG, T., HORAUD, R., YASSINE, A., AND TAO, P. Object pose from 2-d to 3-d point and line correspondences. *International Journal of Computer Vision 15* (1995), 225–243.
11. PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 1996.
12. SHIU, Y., AND ZHUANG, H. Pose determination of cylinders, cones, and spheres. In *Applications of Artificial Intelligence X: Machine Vision and Robotics* (Orlando, Florida, Apr. 1992).

Omni-directional vision with a multi-part mirror

Fabio M. Marchese, Domenico G. Sorrenti

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano - Bicocca
via Bicocca degli Arcimboldi 8, I-20126, Milano, Italy
email {marchese, sorrenti}@disco.unimib.it

Abstract. This paper presents an omni-directional sensor based on a camera and a mirror generated with a surface of revolution. The requirements the device must fulfill result from its use as the main perception system for the autonomous mobile robots used in F2000 RoboCup competitions. The more relevant requirements which have been pursued are: 1) range sensing in a quite wide region centered around the robot, with good accuracy; 2) sensing around the robot in a given vertical sector, in order to recognize team-mates and adversaries (all robots have a colored marker above a given height); 3) range sensing in a region very close around the robot, with the highest accuracy, to locate and kick the ball. Such requirements have been fulfilled by the design of a mirror built up of three different parts. Each part is devoted to the fulfillment of one requirement. Concerning the first requirement the approach developed is based on the design of a mirror's profile capable to optically compensate the image distortion provided by the mirror profiles commonly used in previous literature. This approach resulted to be similar to a previous work by Hicks and Bajcsy, although independently developed by the authors.

Introduction

This work has been accomplished in the framework of the Italian participation to RoboCup [3]. For a more detailed introduction to RoboCup see [4]. A robot capable to compete in a F2000 RoboCup match (F2000 is the so called "middle-size" league, i.e. robots with a dimension of about 0.5 m per side) should be able to observe what happens on the playground in order to recognize and localize objects of interest for the game; e.g. robots, ball and goals. For the above-mentioned aims, an omni-directional vision sensor [6] seems appropriate (Fig. 1); actually this kind of sensing has been chosen by many teams participating to previous RoboCup competitions (e.g. [1][5]). An omni-directional vision sensor should satisfy the following constraints:

1. it must be able to observe around the robot, in the horizontal plane;
2. it must be able to observe the markers, which allow to distinguish team-mates from adversaries, independently from the robots position in the playground; this results in a constraint on the observed angular sector, in the vertical plane;
3. it should be able to perceive colors, because the objects can be distinguished by

their colors (with the current rules all the relevant objects have quite different colors);

4. it must be able to locate the ball (direction and distance) with enough accuracy as follows:
 - 4.1 when the ball is in contact or very near to the robot: very good accuracy, in order to properly control the kicking;
 - 4.2 when the ball is within few meters from the robot: good accuracy and constancy of the accuracy in the range in order to control the motion to properly approach the ball itself;
 - 4.3 when the ball is quite far: good accuracy for the direction, in order to be able to head toward the ball, some inaccuracy may be allowed for the distance;
5. it must allow localization of the relevant objects (players, goals, and other relevant features of the playground);
6. it must allow the self-localization of the robot with respect to the playground.

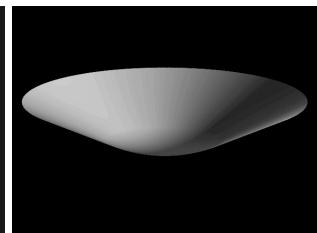
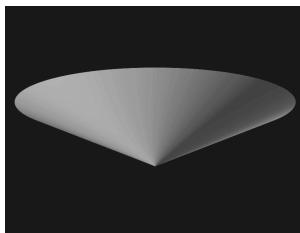
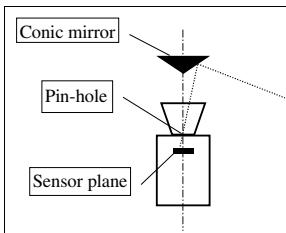


Fig. 1. Sketch of the COPIS sensor (camera plus mirror configuration)

Fig. 2. Conic mirror

Fig. 3. Conic-spherical mirror

In literature different mirror geometries have been proposed and even in RoboCup some teams already used mirrors other than the original conic one (Fig. 2). For instance in [1], a conic mirror with a "spherical vertex" (Fig. 3) has been motivated by the need of getting a higher resolution in the area close to the robot (the spherical part magnifies the scene). Such mirrors introduce large distortions on image distances of objects at the playground level. It should be noted that such distortion grows with the distance of the object from the observer (Fig. 4b). On one hand, it is quite obvious that the "nominal" value of the estimate can be easily corrected exploiting the known profile of the mirror; on the other hand, the accuracy of the measure is corrupted without the possibility to compensate for such degradation.

Therefore, one of the objectives of this work was to develop an optical compensation of the above-described distortion, working directly on the mirror profile in such a way that the absolute localization error remains constant with the object distance. Both the idea of an optical compensation and the analytical set up for the determination of the compensating mirror profile turned out to be indistinguishable from a previous work, by Hicks and Bajcsy [2]. Such work does not detail some implementation aspects, which are presented here instead. Moreover, it should be mentioned that the approach here taken, which aims at the definition of a complete

solution to a set of real requirements, involves more than the optical compensation just mentioned, which provides a solution to only one requirement. Such global approach implies the integration of the different proposed solutions to each requirement. Due to the fact that each solution is the design of a part of a mirror, the overall outcome of this work is one single mirror, built up of different parts.

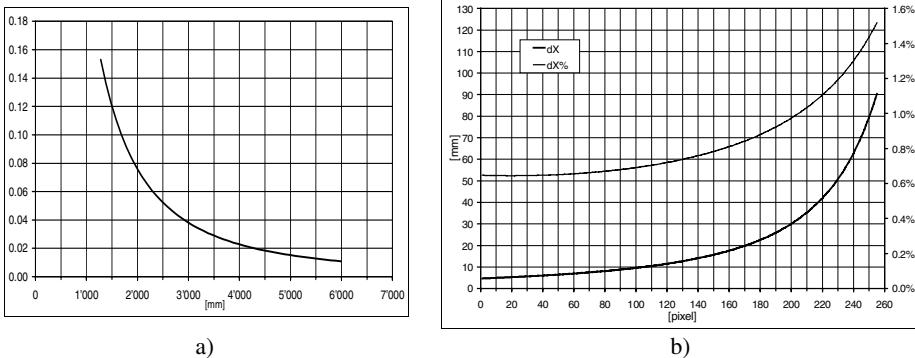


Fig. 4. Conic mirror: a) the image dimension of a given object vs. its distance from the observer; b) absolute (dx) and relative ($dx\%$) error affecting the localization due to the spatial sampling of the radial distance from the center of the image

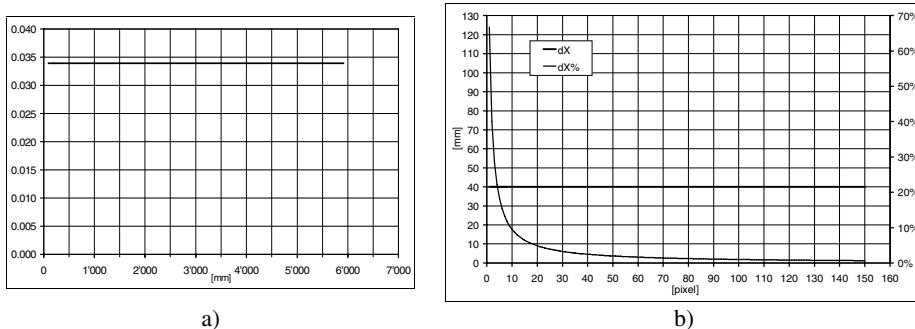


Fig. 5. Isometric mirror: a) the image dimension of a given object vs. its distance from the observer; b) absolute (dx) and relative ($dx\%$) error affecting the localization due to the spatial sampling of the radial distance from the center of the image

Concerning the optical compensation, it should be observed that a constant absolute error, with respect to the distance, can be attained as well as a decreasing relative error. This allows for a better accuracy in locating objects, with respect to what can be attained with commonly used mirrors (compare Fig. 4b and 5b). We called “isometric” this kind of mirror because of its capabilities to map scene distances, in any direction, in proportional image distances within the whole range covered by the mirror (Fig. 4a and 5a). With a conventional (e.g. conic) mirror, the image size of an object is maximum when the object is in contact, is minimum when the object is at the greatest distance. It should now be clear that the effect of the distortion due to

conventional mirrors represent a degradation. This is true not only under the point of view of the localization accuracy, but also under the point of view of the detection of relevant features (e.g. the ball). The smaller the ideal image size, the higher the probability of a detection failure of the feature. A failure can happen when the image formation will not take place under ideal conditions. Non-ideal conditions are due to shadows, non-uniform lightening, electronic noise in the hardware, etc.

Mirror Design

In this section the procedures followed to design the proposed mirror are introduced. To completely satisfy the requirements, the mirror design has been split in three parts: isometric, constant curvature and planar.

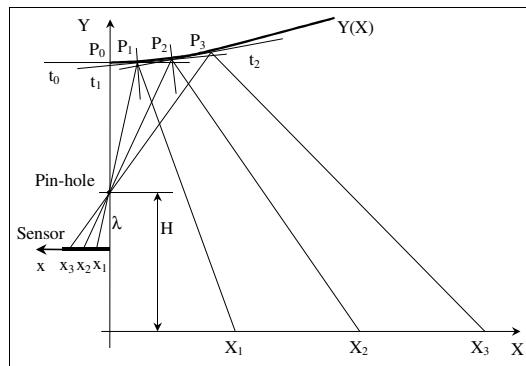


Fig. 6. Sketch for inferring the differential equation generating the isometric part of the mirror

Isometric Mirror Part

The first requirement implies the design of a mirror capable to compensate the distortion, introduced by the linear profile of a conic mirror, by means of a non-constant curvature of the profile. This is the idea that turned out to be the same as the one proposed in [2]. This design problem has been modeled by the following differential equation (1), which can be inferred by applying the laws of the Linear Optics (Fig. 6).

$$\frac{\frac{X}{Y} + \frac{2Y'}{1-Y'^2}}{1 - \frac{X}{Y} \frac{2Y'}{1-Y'^2}} = \frac{\eta Y - X^2}{X(Y+H)}; \quad \begin{cases} Y(0) = Y_0 \\ Y'(0) = 0 \end{cases} \quad (1)$$

where: $Y' = dY / dX$, $\eta = k \lambda$, λ = focal length, k = proportionality constant from X to x , H = pin-hole height from the playground.

This equation has been written in a reference frame XY centered in the pin-hole of the camera. Refer to [2] to find a different formulation obtained under similar conditions. Differently from [2], we developed a "geometrically" based integration of equation (1). Our approach bases on a local first order approximation of the profile: at each point the profile has been approximate by its tangent. The higher the considered number of points on the profile, the better the approximation. The mirror profile should compensate for the radial distortion of the lengths, so that to a given length on the image plane should correspond a proportional length on the scene plane, independently from the position in the scene/image. Referring to Fig. 6, this means that we want to obtain:

$$\frac{X_1}{x_1} = \frac{X_2}{x_2} = \dots = \frac{X_n}{x_n} = k \quad (2)$$

The resulting profile looks quite similar to the one obtained in [2]. It is convex into its first half, i.e. the part that goes from axis of symmetry toward the outside of the mirror. It then has an inflection point and then gets slightly concave. Fig. 7 shows a scene, used in the rest of the paper; the observer robot is the one at the center of the playground. The image observed through the mirror in such a configuration is presented in Fig. 8. The transformation between the two 2D Euclidean spaces, i.e. from the playground to the sensor plane, keeps angles unchanged and changes lengths by the multiplication with a constant. This transformation, being linear, changes its metric Euclidean tensor to the same metric Euclidean tensor, neglecting the constant.

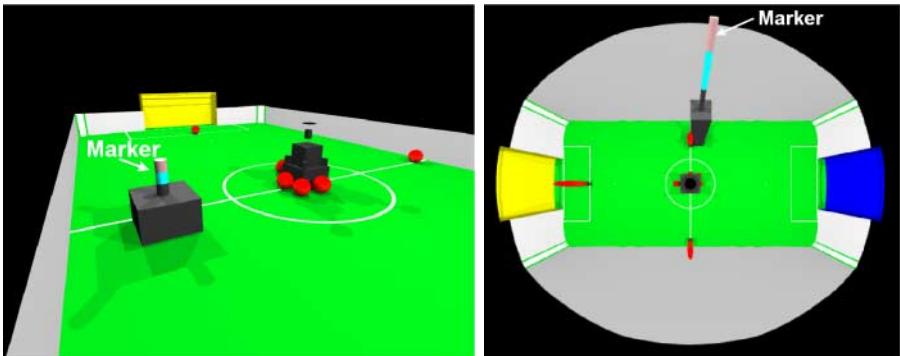


Fig. 7. Perspective view of the scene

Fig. 8. The same scene as in Fig. 7, as seen by the robot through an isometric mirror

Hence lengths remain unchanged, apart the constant, angles remain unchanged, and parallel lines remain parallel. As an example, a chessboard on the playground is transformed in a scaled chessboard in the image. This is true disregarding the aspect ratio, i.e. the different sampling frequencies on the sensor plane.

Constant Curvature Mirror Part

The mirror design described above does not satisfy all the requirements because it observes an angle which heads too low, in such a way that it cannot observe the higher part of the scene. The information needed to distinguish the robots of the two teams can be found instead at a quite high height. Therefore we decided to devote part of the image and of the mirror to this aim. This part of the mirror satisfies a different design criterion with respect to the isometric one. As already mentioned, the process for identifying each robot as a teammate is based on the marker color and not on its shape. Therefore, it has been possible to release the no-distortion requirement, which was convenient for the isometric part. On the other hand, the continuity between the two portions of the image should be preserved, to be able to associate the marker to the body of the robot. The robot body is underneath the marker, i.e. nearer to the image center, along the same radius. Moreover, when the bottom of the robot is observed through the isometric part of the mirror, which is a very likely case, it will be possible to measure its distance with a quite good accuracy. The image continuity can be granted by imposing the continuity of the tangent on the junction between the isometric and the new part of the mirror (point A in Fig. 9). Another condition comes from fixing the point $B = (X_B, Y_B)$ and setting height H_{\max} so that it can be observed at distance d_{\max} . This constraint gives the tangent to the profile in point B.

$$\begin{aligned}\tan(\tau) &= (H_{\max} - Y_B) / (d_{\max} - X_B) \\ \tan(\beta) &= x_b / \lambda \\ \tan(\gamma) &= \tan((\beta + \tau - \pi) / 2)\end{aligned}\quad (3)$$

where: λ = focal length.

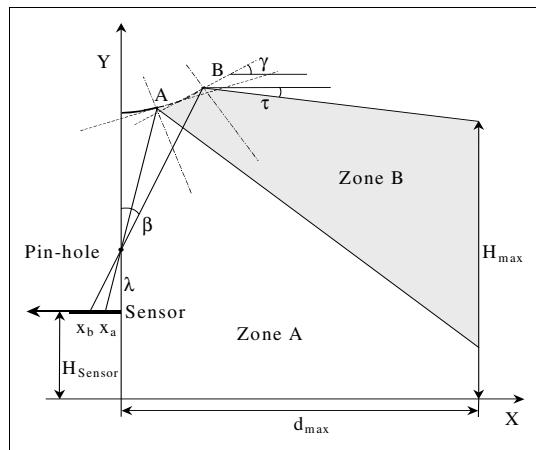


Fig. 9. Sketch for the design of the "constant curvature" part of the mirror

Because there is no other constraint, this portion of the mirror can be designed, e.g., by imposing a constant variation of the tangent between the two endpoints. Hence the

name "constant curvature" given to this part of the design. In this condition the mirror will cover completely the highest part of the scene (Zone B). On the other hand, when the robots are quite near, they will be observed by the first part of the mirror (Zone A).

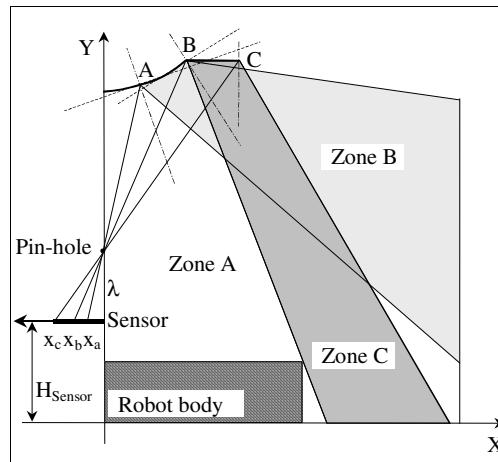


Fig. 10. Sketch for the design of the "planar" part of the mirror

Planar Mirror Part

The overall mirror still does not satisfy the requirements: due to the robot body occlusion (Fig. 10), it is not possible to observe the scene in its neighborhood. More specifically, when the ball is in contact or very near to the robot, it is not observable. Moreover, the isometric part of the mirror produces a too small image of the ball when it is in the kicking range, i.e. where the highest accuracy should be needed in order to control accurately the kicking contact. To solve this problem a third part of mirror has been introduced, aiming at observing an area very close to the robot body. This part of the mirror should be the outmost part of it, although it looks nearer than the other parts do. The reason for this is the need to have the least occlusion from the robot body. The simplest solution to this problem is a planar mirror, specifically a circular crown, lying on a plane perpendicular to the rotational axis. The height of this part has to be as low as possible, with respect to the camera, in order to give out the largest images of the ball. On the other hand this part of mirror should not be on the line sight of other parts of mirror. Hence the choice is to have the circular crown at the same height of the last point of the constant curvature part of the mirror (point B in Fig. 10). The radial dimension (point C) is set as follows:

$$\frac{X_c}{(Y_c - \lambda - H_{\text{Sensor}})} = \frac{x_c}{\lambda} \quad (4)$$

$$Y_c = Y_B$$

where H_{Sensor} = height of sensor plane, λ = focal length.

The area observed by this part of the mirror at the playground level (zone C) is observed by the isometric part of the mirror too. The image produced by the circular crown, on the other hand, is much larger (Fig. 13), hence allowing a more reliable detection and a more accurate localization of the ball when it is near the robot. This configuration produces a discontinuity in the image, which is straightforward to be taken into account.

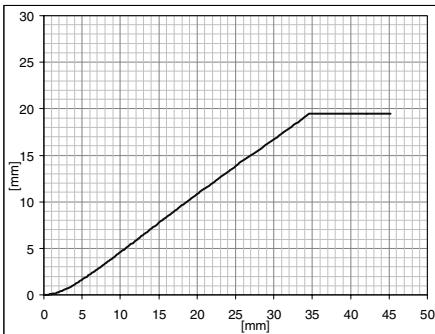


Fig. 11. Profile of the overall mirror

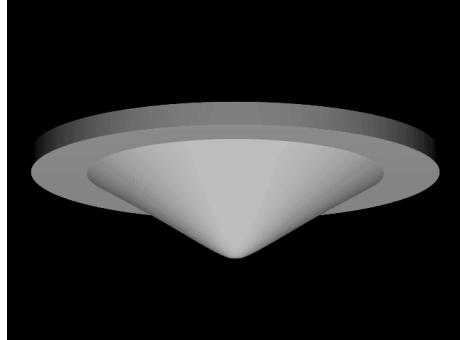


Fig. 12. The proposed mirror

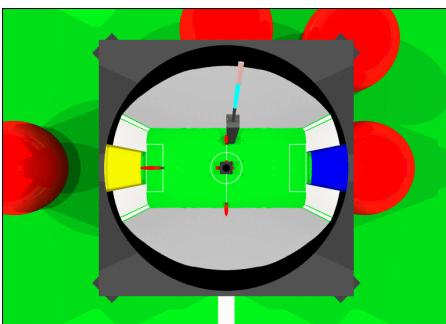


Fig. 13. Image taken when the robot is in the center of the playground

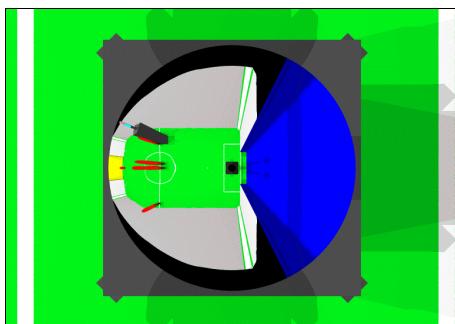


Fig. 14. Image taken when the robot is in front of the blue goal

The Resulting Mirror

The mirror profile resulting from the design described before is shown in Fig. 11. The mirror (Fig. 12) is capable to observe up to 6 m far away without image distortion at the playground level; thanks to its constant curvature part it can observe up to the maximum height, 600 mm, at the maximum distance allowed in the playground (11.2 m). Its outer part allows the observation of objects from 0.39 m to 0.51 m. Fig. 13 shows the image that can be obtained using such mirror on the scene of Fig. 7. By comparing the third part of the mirror with the first one, it can be seen that the third part allows an easier detection and localization of the ball. In Fig. 14 the robot

has been moved in front of the blue goal. The effect of the optical compensation lasts up to 6 m, but, thanks to the continuity with the second part, it is still possible to detect the yellow goal and the other robot marker, although they are distorted by the constant curvature part of the mirror. The distortion is due to the fact that the body of the objects, i.e. the balls and the robot, are over the playground whilst the isometric compensation holds only at the playground level. However, their distances from the observer, measured at the contact point with the playground, can still be recovered with the limited error provided by the isometric design.

Conclusions

The paper stems from the definition of some requirements for the sensor system of a robot for F2000 RoboCup competitions. The proposed solution is based on the design of a mirror of an omni-directional system, differently from approaches trying to compensate in software the shortcomings of conventional mirror design. The proposed design resulted in a three-part mirror, each part being devoted to fulfill one of the requisites. The first part aims at an un-warped image of the playground; this development resulted nearly identical to a precedent work [2]. Details on the integration of the differential equation governing this part of the design are presented. The second part allows to recognize teammates and adversaries observing colored markers over them. The third part allows a precise localization of the ball when very close to the robot.

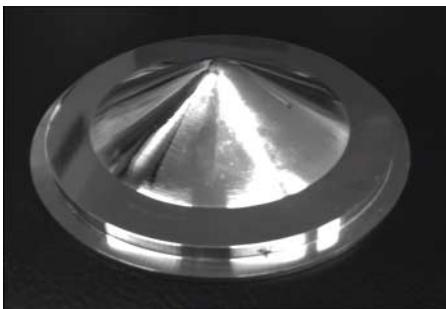


Fig. 15. First prototype of the proposed mirror

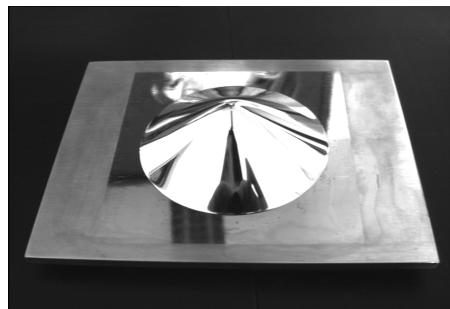


Fig. 16. The latest prototype of mirror obtained with a new design

A preliminary prototype of the proposed mirror, quite heavy and affected by some machining errors, is shown in Fig. 15. An improved design has been applied to machine the mirror of Fig. 16. Examples of images captured with the later are shown in Fig. 17 and 18.

Acknowledgements

This work exists thanks to the contribution of Prof. Andrea Bonarini, Dipartimento di Elettronica e Informazione, Politecnico di Milano. His contribution has been fundamental for our effective introduction to the field of RoboCup competitions. Without his experience, the setting up of the requirements would have not been possible. Moreover, we are grateful for the very fruitful conversations, which took place during the development of the work. We are also pleased to thank Parravicini SrL for the machining of the latest prototype of mirror and Boitor SaS for its polishing.

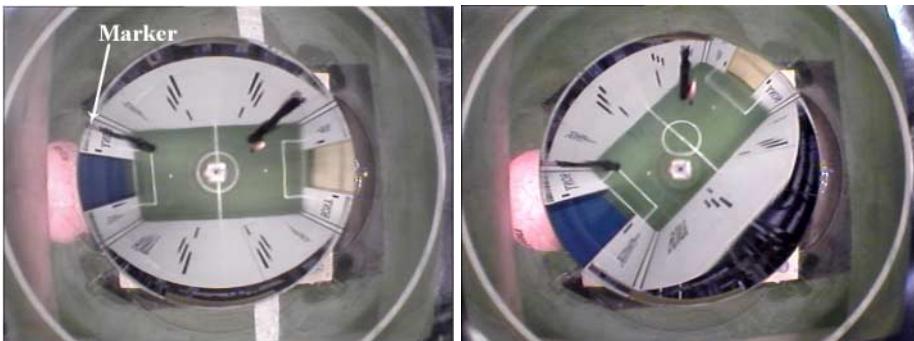


Fig. 17. Image captured from the middle of the **Fig. 18.** Image captured after a robot playground taken in field B at RoboCup2000 movement (translation + clockwise rotation in Melbourne of 45 degrees)

References

- [1] A. Bonarini, P. Aliverti, M. Lucioni, "An omni-directional sensor for fast tracking for mobile robots", Proc. of the 1999 IEEE Instrumentation and Measurement Technology Conference (IMTC99), IEEE Computer Press, Piscataway, NJ, pp. 151-157
- [2] R. A. Hicks, R. Bajcsy, "Reflective Surfaces as Computational Sensors", IEEE Workshop on Perception for Mobile Agents, Proc. of the 1999 IEEE Conference on Computer Vision and Pattern Recognition (CVPR99), IEEE Computer Press, Piscataway, NJ
- [3] D. Nardi, G. Clemente, E. Pagello, "Art Azzurra Robot Team", in Robocup98: Robot Soccer World Cup II, M. Asada ed., Berlin, 1998, pp. 467-474, Springer Verlag
- [4] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, H. Matsubara, "RoboCup: a challenging AI problem", AI Magazine, Vol. 18, N. 1, 1997
- [5] S. Suzuki, T. Katoh, M. Asada, "An application of vision-based learning for a real robot in RoboCup learning of goal keeping behavior for a mobile robot with omni-directional vision and embedded servoing", in Robocup98: Robot Soccer World Cup II, M. Asada ed., Berlin, 1998, pp. 467-474, Springer-Verlag
- [6] Y. Yagi, S. Kawato, S. Tsuji, "Real-time omni-directional image sensor (COPIS) for vision-guided navigation", IEEE Trans. on Robotics and Automation, Vol. 10, N. 1, pp. 11-22, 1994

Observation strategy for decision making based on information criterion

Noriaki Mitsunaga and Minoru Asada

Dept. of Adaptive Machine Systems, Osaka University, Suita, Osaka, 565-0871, Japan

Abstract. Self localization seems necessary for mobile robot navigation. The conventional method such as geometric reconstruction from landmark observations is generally time-consuming and prone to errors. This paper proposes a method which constructs a decision tree and prediction trees of the landmark appearance that enable a mobile robot with a limited visual angle to observe efficiently and make decisions without global positioning in the environment. By constructing these trees based on information criterion, the robot can accomplish the given task efficiently. The validity of the method is shown with a four legged robot.

1 Introduction

When it makes a decision, a mobile robot highly depends on its own location. For self-localization, methods such as dead reckoning and global positioning by geometric reconstruction from image data are commonly used [1][2][3]. However, it seems difficult for a legged robot to use dead reckoning without special treatment since the moving distance on the walking plane is less accurate than that of a wheeled robot. For the latter, geometric reconstruction often highly depends on accurate motion information. Such information may include drastic errors in the case of legged robots due to causes such as slipping. To handle this, Furthermore, it takes a lot of time not just for geometric reconstruction but also for observation to capture the image surrounding the robot in order to make geometric reconstruction stable and accurate. Human beings do not seem to take such a strategy to localize themselves. Rather, they use the minimum necessary information for their decision making when they do not have enough resources (time, etc.).

Moon et al. [4][5] have proposed a view point planning method to move efficiently by reducing the frequent observation for self-localization mostly based on dead reckoning. To apply their method, the map and the route to the goal should be given in advance.

Burgard et al. [6] have proposed an active localization method by Markov localization using occupancy grids as a world model representation. To apply their method, the map of the environment and the dead reckoning model need to be prepared. Also calculation and memorization costs for occupancy grids are very high. These methods depend on geometric reconstruction.

Tani et al. [7] have experimented with the task of watching two visual targets with a limited view angle camera. The robot switched the visual attention depending on prediction accuracy. Since the robot action (wall following) is fixed, the issue can be regarded as a view prediction problem on a route which is almost fixed.

In this paper, we propose a method for a robot, which has a limited view angle camera with panning facility, to make a decision by efficient observation without explicitly localizing itself. With a limited view camera, a robot can widen the angle by panning, but it takes time. Efficient observation is done by a decision tree and prediction trees constructed based on the information criterion. The basic idea of our observation strategy is not for self localization but for decision making, that is, to minimize observation as long as decision making is possible. By constructing a decision tree, one can know which landmark to observe first. Similarly by making and using prediction trees with information criterion, one can reduce the time for observation through decision making.

2 The method

2.1 Assumptions

We assume that the robot can make a decision for the given task at any position by panning its camera head. Before making the decision and prediction trees, sufficient example data are necessary. We used a teaching method to collect such data.

2.2 Making decision and prediction trees

Suppose we have m landmarks and q kinds of actions. Each appearance of the landmark is quantized into r kinds of viewing categories including a non-visible situation. A training datum consists of a set of the appearance of the landmarks at the current position and the action to accomplish the task, and we have n training data. During the training period, the robot pans its camera head from the left-most angle to the right most one, and observes as many landmarks as possible.

First, calculate the occurrence probabilities of actions p_k ($k = 1, \dots, q$) as $p_k = n_k/n$, where n_k denotes the number of taken action k . Therefore, the information I_0 for the action probability is given by

$$I_0 = - \sum_k p_k \log_2 p_k. \quad (1)$$

Next, calculate the occurrence probabilities of actions after each appearance of the landmark was known. We denote the number of times action k was taken as n_{ijk} when the landmark i is observed at the quantized direction j . Then, the occurrence probability becomes,

$$p_{ijk} = \frac{n_{ijk}}{\sum_k n_{ijk}}. \quad (2)$$

Next, calculate the expected information after one of these landmarks is found, as follows:

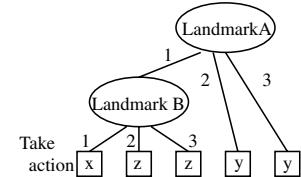
$$I_i = - \sum_j \left\{ \frac{\sum_k n_{ijk}}{\sum_j \sum_k n_{ijk}} \sum_k (p_{ijk} \log_2 p_{ijk}) \right\}. \quad (3)$$

The smaller I_i is, the smaller the uncertainty after the observation is. We put the landmarks into the tree in decreasing order of uncertainty after its observation. This information criterion is same as ID3 [8]. For the training data which take different actions for the same situation, we add a leaf for each action and record the probability that it was taken.

For example, suppose we have training data as shown in Fig.1(a). Since $p_x = 0.2$, $p_y = 0.4$, and $p_z = 0.4$, I_0 is 1.52. By calculating each p_{ijk} , we have expected informations, $I_A = 0.551$, $I_B = 0.8$, and $I_C = 1.2$. So the ordering in the decision tree is landmark A, B, and C, and the tree is shown in Fig.1(b). We calculate the landmark appearance prediction trees in the same manner.

Landmark A	Landmark B	Landmark C	Action
1	1	1	1
2	1		2
3	2	1	2
1	3	1	3
1	2	2	3

(a) trainig data



(b) action decision tree

Fig. 1. Examples of training data and an action decision tree.

2.3 Calculation of the probability distribution

Here, we denote the probability that the viewing direction of the landmark i is j at time t as $p_{ij}^L(t)$ ($i = 1, \dots, m, j = 1, \dots, r$), the probability that the action k was taken at time t as $p_k^a(t)$ ($k = 1, \dots, q$), and the probability that the action k should be taken by the training data at t as $\hat{p}_k^a(t)$ ($k = 1, \dots, q$).

Calculation of the probability distributions are as follows. If currently the landmark i is observed in the quantized viewing direction J we assign the probability 1.0 to $p_{ij}^L(t)$ and 0s to others ($p_{ij}^L(t) = 0$ ($j \neq J$)). When the previous taken action was K , set $p_K^a(t-1) = 1$ and $p_k^a(t-1) = 0$ ($k \neq K$). The probabilities of the landmarks, which are not currently observed, are predicted by the prediction trees using the probability distributions $p_{ij}^L(t-1)$ and $p_k^a(t-1)$. We assign the probability 1.0 to the quantized invisible direction and 0s to others if the landmark cannot be observed while the robot looks around.

Following a landmark prediction tree from the root to one of the leaves, one can obtain 1) the condition of the appearances of the landmarks and the action at time $(t-1)$ which is given by logical product, and 2) the consequence appearance of the landmark at time t . In order to calculate the probability of reaching each leaf, we change the logical product to an arithmetic one and conditions to probabilities at time $(t-1)$. We consider the summation of the probabilities of the leaves of the same appearance as the probability of the appearance at time t .

To calculate the action decision probability $\hat{p}_k^a(t)$, we use these probability distributions $p_{ij}^L(t)$, and follow the action decision tree in the same manner.

2.4 Decision making

In order to make a decision on which action to take, the robot calculates the $\hat{p}_k^a(t)$ as described above. If one of the action probabilities is very high, it takes that action. Otherwise, until one of them becomes high enough, it continues to try to observe the landmark whose probability distributions is flat in the order in which they were placed in the action decision tree (information criterion). When the robot checks the landmark, one may find a peak of the profile, and observation of that direction may help.

3 Experiments

3.1 Task and Assumptions

The task is to push a ball into a goal based on the visual information. We used a legged robot with a limited view angle for the RoboCup 99 SONY legged robot league (Fig.2). In the field, there are 8 landmarks, that is, target goal (TG), own goal (OG), north west pole (NW), north east pole (NE), center west pole (CW), center east pole (CE), south west pole (SW), and south east pole (SE). All the landmarks and the ball are distinguished by their colors. The view angle / number of image pixels of the robot's camera are about 53 degrees / 88 pixels in width, and about 41 degrees / 59 pixels in height. Each leg and the neck have three degrees of freedom. We fixed the joint angles of the legs and the neck except for the pan joint when it observes the landmarks and the ball to make its decision. The robot can rotate the pan joint from -90 degrees to 90 degrees.

The ball can be treated as a special landmark, which is static if the robot does not push it while it can change its position if the robot pushes it. Therefore, the appearance of the ball can be predicted by its previous location in the image and the action of the robot. Note that for the ball prediction tree we cannot use other landmarks because it may move in the field.

Each landmark's appearance is quantized into eight categories, that is, seven directions and one invisible situation (Fig.4). The ball appearance was quantized into eleven categories, that is, the product of five directions and two kinds of distances (near or far), and one invisible situation.



Fig. 2. The SONY legged robot for RoboCup 99 SONY legged robot league.

Since the training data may lack some situations, the sum of the probability distribution $\sum_{i=1}^N p_i$ might be less than one. To avoid this, we added $(1 - \sum_{i=1}^N p_i)/N$ to each p_i . N indicates the size of the distribution.

When it plays back the taught action based on the trees, the robot looks all around to search landmarks and the ball (if they are invisible) if the peak action probability was below 0.6 .

3.2 Experiment 1

In the field (Fig.3), we experimented with the task of guiding a ball into the goal. The ball was put in front of the goal (the circle before the target goal in the figure) and the robot was placed at middle of the field (one of three cross marks in the middle). As actions, we prepared three movements, forward, left forward and right forward for 4.8 seconds. The walking was programmed by us and a motion for 4.8 seconds corresponds to four walking periods. In 4.8 seconds, the robot walks about 0.45[m] in forward movement and at most places the appearance of the landmarks changes due to one of the motions.

We trained the robot starting from one of three positions in the middle of the field. For each starting position, we trained five times and obtained eighty data points to construct trees. We show the sizes (the number of leaves, minimum, mean, and maximum depth) of decision and prediction trees in Table 1. The orders in the trees by the information criterion are shown in Table 2.

Next, we show the examples of action sequences using these trees. From the starting position in the center of the field, the robot took the forward motions four times. In this experiment, the ball and the target goal were observed at every moment for decision making. The probability distributions of landmark observation predictions and the action decision are shown in Fig.5.

The robot took different actions even if the starting position was the same. This is due to the quantization of the observation and variance of the walking. In this example, the robot started from the center of the field as in the first example, but then took other actions instead (1) forward, 2) forward, 3) landmark

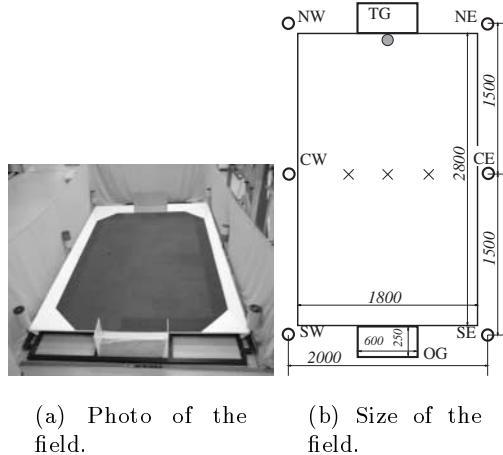


Fig. 3. Experimental field (same as the one for RoboCup SONY legged league). Cross marks indicate the initial positions of the robot and the ball for the first experiment.

observation, 4) forward, 5) landmark observation, 6) left forward, 7) forward, and 8) forward). In this experiment, the ball and the target goal were observed at every moment for decision making. The probability distributions of landmark observation predictions and the action decision are shown in Fig.6.

Here is another example starting from the center right cross mark. In this example, the robot took the actions, 1) left forward, 2) landmark observation, 3) forward, 4) forward, and 5) left forward. The ball and the target goal were observed at every moment for decision making. The probability distributions of landmark observation predictions and the action decision are shown in Fig.7.

We show the number of re-observations in Table 3.2. Each number indicates the number of trials, total steps, re-observations, and the rate of re-observations.

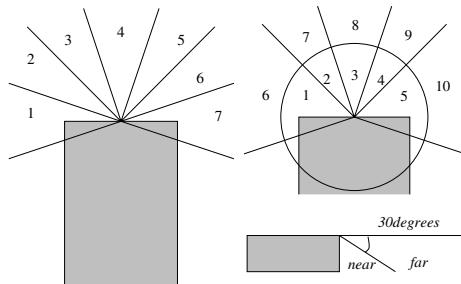


Fig. 4. Quantization for landmarks (left) and the ball (right).

Table 1. Depth of the action and prediction trees (Experiment 1).

	# of leaves	min depth	mean depth	max depth
action	43	1	4.91	8
ball	52	2	2	2
OG	13	1	4.23	8
TG	44	1	5.39	8
SE	6	1	2	3
SW	1	0	0	0
CE	28	2	4.69	8
CW	11	1	3.91	8
NE	51	1	5.96	8
NW	54	2	5.91	8

Table 2. The order of information in trees (Experiment 1, ‘act’ means action, ‘1’ means root node).

	1	2	3	4	5	6	7	8
action	ball	TG	NE	NW	CW	CE	OG	SE
ball	ball	act						
OG	act	NE	TG	NW	CW	CE	OG	SE
TG	TG	act	NE	NW	CE	OG	CW	SE
SE	act	CE	NE	OG	NW	TG	CW	SE
SW	-							
CE	act	NE	TG	CE	NW	CW	OG	SE
CW	TG	act	NE	NW	CE	CW	OG	SE
NE	NE	act	NW	TG	CE	CW	OG	SE
NW	act	NE	TG	NW	CE	OG	SE	CW

We see that the number of re-observation is reduced to about half of the total steps.

3.3 Experiment 2

In the same field we trained the robot with regard to the games of the RoboCup 99. In this experiment, we placed the robot and the ball at many more locations than in Experiment 1. To reduce the load of teaching, we prepared six actions, forward, left forward, right forward, left rotation, right rotation, and track the ball. Each action is performed for 4.8 seconds.

By this training, we obtained 1364 data points. Deleting inappropriate data, we used 856 data points for the action decision tree and 1364 data points for the ball and landmarks predictions. We show the sizes and the order of the trees based on the information criterion in Tables 4 and 5. We used these data for the RoboCup 99 and the robot showed the movement that was expected though the robot did look for the landmarks more frequently than anticipated.

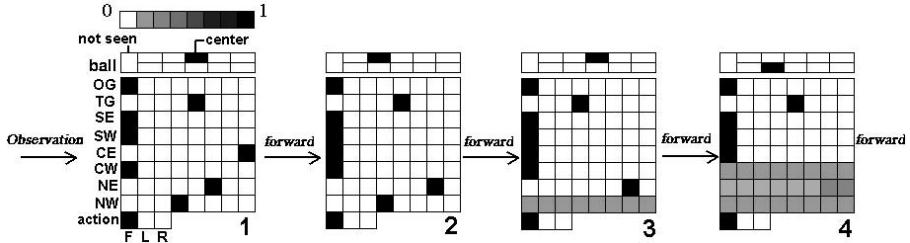


Fig. 5. Probability distribution in Experiment 1-1 (The gray scale of each box indicates the probability. Darkest 1 and lightest 0.).

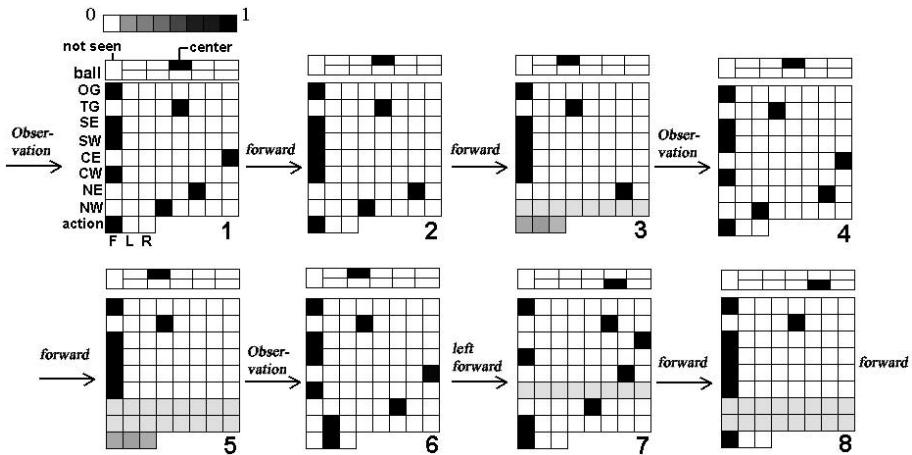


Fig. 6. Probability distribution in Experiment 1-2.

4 Discussions and conclusions

From Experiment 1, we see that the action probability distributions have either a sharp peak (nearly equal to 1.0) or a very flat profile. When the action probability profile is flat, the landmark probability profile, which is important for decision making, is also flat due to poor training data for making prediction trees. Then, the robot looked all around to search the landmarks and the ball.

Comparing the order of action and landmarks in prediction trees between Experiment 1 and Experiment 2, we notice that the action is higher priority in Experiment 1 than in Experiment 2. It seems that in Experiment 1 the training data was too few to extract the fact that prediction of landmarks highly depends on its location, which indicates prediction can be done mostly by landmark observation. Note that although the order is different in both cases, they are extracted from the training data by information criterion and are therefore

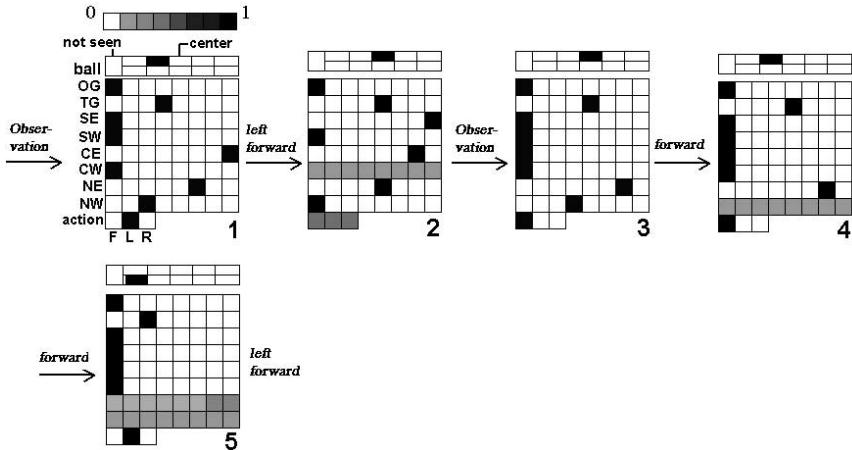


Fig. 7. Probability distribution in Experiment 1-3.

Table 3. The number of needed observation (Experiment 1).

begin from	# of trials	# of total steps	# of re-observation rate	of re-observation
center	12	35	18	.51
left	12	31	15	.48
right	16	64	38	.59

optimal in that sense. Although the proposed method does not depend on it, the training data should be sufficient to accomplish the task.

In this paper, we quantized the appearance space by hand. But, by using a method like C4.5 [9] for making action decision trees, quantization may be self-organized. The order in the tree, which is fixed in this method for memory consumption and simplicity, might be changed as ID3 or C4.5 for further abstraction and observation efficiency.

References

1. J. Wendler, S. Brüggert, H.-D. Burkhard, and H. Myritz. Fault-tolerant self localization by case-based reasoning. In *Proc. of The Fourth International Workshop on RoboCup*, pages 82–89, 2000.
2. S. Enderle, M. Ritter, D. Fox, S. Sablatnög, G. Kraetzschmar, and G. Palm. Vision-base localization in robocup environments. In *Proc. of The Fourth International Workshop on RoboCup*, pages 232–237, 2000.
3. S. Lenser and M. Veloso. Sensor resetting localization for poorly modeled mobile robots. In *Proceedings of ICRA-2000*, 2000.
4. I. H. Moon, J. Miura, Y. Yanagi, and Y. Shirai. Planning of vision-based navigation for mobile robot under uncertainty. In *Proceedings of the 1997 IEEE/RSJ Interna-*

Table 4. Depth and size of the trees (Experiment 2).

	# of leaves	min depth	mean depth	max depth
action	557	2	5.87	9
ball	403	2	2	2
OG	958	2	7.58	9
TG	1050	2	7.67	9
SE	845	2	7.35	9
SW	901	2	7.41	9
CE	901	2	7.13	9
CW	873	2	7.37	9
NE	1031	2	7.60	9
NW	980	2	7.55	9

Table 5. The order of information in trees (Experiment 2).

	1	2	3	4	5	6	7	8	9
action	ball	TG	OG	SW	SE	NW	NE	CE	CW
ball	ball	act							
OG	OG	SE	SW	TG	NW	CW	NE	CE	act
TG	TG	OG	SE	SW	NW	NE	CW	CE	act
SE	SE	OG	TG	SW	CE	NE	NW	CW	act
SW	SW	OG	CW	SE	TG	NW	NE	CE	act
CE	CE	SE	OG	TG	NE	SW	NW	act	CW
CW	CW	SW	OG	TG	NW	SE	NE	CE	act
NE	NE	TG	NE	OG	SE	CE	NW	SW	CW
NW	NW	TG	OG	SW	CW	SE	NE	CE	act

tional Conference on Intelligent Robots and Systems, volume 1, pages 1202–1207, 1997.

5. I. Moon, J. Miura, and Y. Shirai. Dynamic motion planning for efficient visual navigation under uncertainty. In Y. Kakazu, M. Wada, and T. Sato, editors, *In Proc. of the Intelligent Autonomous Systems 5*, pages 172–179, 1998.
6. W. Burgard, D. Fox, and S. Thrun. Active mobile robot localization. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, San Mateo, CA, 1997.
7. J. Tani, J. Yamamoto, and H. Nishi. Dynamical interactions between learning, visual attention, and behavior: An experiment with a vision-based mobile robot. In P. Husbands and I. Harvey, editors, *Fourth European Conference on Artificial Life*, pages 309–317. The MIT Press, 1997.
8. J. R. Quinlan. Discovering rules from large collections of examples: a case study. In D. Michie, editor, *Expert Systems in the Microelectronic Age*. University Press, Edinburgh, Scotland, 1979.
9. J. R. Quinlan. *C4.5: PROGRAMS FOR MACHINE LEARNING*. Morgan Kaufmann Publishers, 1993.

Towards a Logical Approach for Soccer Agents Engineering

Jan Murray, Oliver Obst, Frieder Stolzenburg

Universität Koblenz-Landau, Fachbereich Informatik
Rheinau 1, D-56075 Koblenz, GERMANY
`{murray,fruit,stolzen}@uni-koblenz.de`

Abstract. Building agents for a scenario such as the RoboCup simulation league requires not only methodologies for implementing high-level complex behavior, but also the careful and efficient programming of low-level facilities like ball interception. With this hypothesis in mind, the development of RoboLog Koblenz has been continued. As before, the focus is laid on the declarativity of the approach. This means, agents are implemented in a logic- and rule-based manner in the high-level and flexible logic programming language Prolog. Logic is used as a control language for deciding how an agent should behave in a situation where there possibly is more than one choice.

In order to describe the more procedural aspects of the agent's behavior, we employ state machines, which are represented by statecharts. Because of this, the script language for modeling multi-agent behavior in [8] has been revised, such that we are now able to specify plans with iterative parts and also reactive behavior, which is triggered by external events. In summary, multi-agent behavior can be described in a script language, where procedural aspects are specified by statecharts and declarative aspects by logical rules (in decision trees). Multi-agent scripts are implemented in Prolog. The RoboLog kernel is written in C++ and makes now use of the low-level skills of the CMUnited-99 simulator team.

1 Introduction and Overview

The RoboLog Koblenz system is based on a multi-layered architecture, as introduced in [8]. In the following, we will consider each layer in some detail and explain its (revised) functionality. Several aspects of some layers will be further elaborated in the subsequent sections. A picture of the overall architecture is given in Figure 1. The team that participated in the RoboCup-99 and its theoretical background are described in [5, 8]. See also the team description *RoboLog Koblenz 2000* in this volume.

1.1 A Layered Architecture

The RoboLog system allows us to program *complex actions* and *cooperation*. This is settled in the higher layers of our system architecture (look at Figure 1). For this, a script language has been developed, which is expressible enough to specify plans explicitly, where several agents may work in parallel, trying to achieve one goal together in collaboration. However, one problem with this approach was, that agents may have

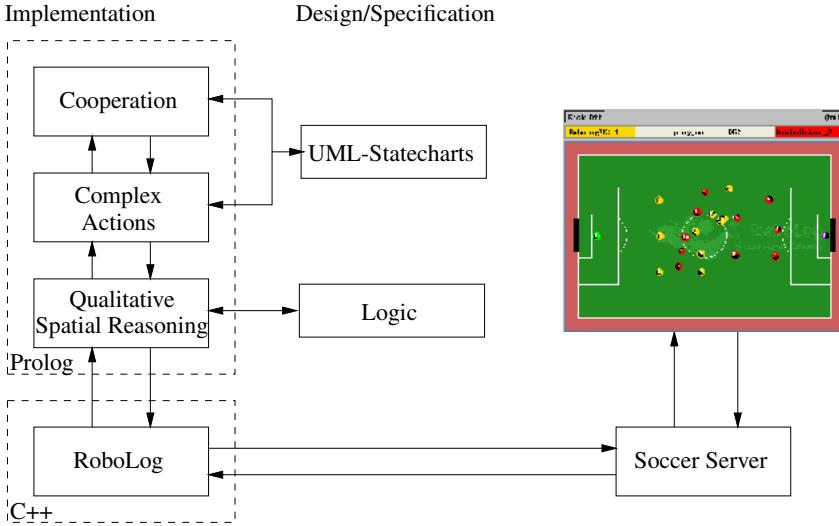


Fig. 1. Overall Architecture of the RoboLog system.

difficulties in reacting to external events or when parts of the intended plan fail. Therefore, we revised the script language and the execution model, such that we can now deal with such events by employing state machines. See Sections 2 and 3.

Logic programming enables us to program specific behaviors and strategies for RoboCup agents. Some of the currently implemented strategies will be explained in the beginning of Section 4. It turns out that besides explicit programming of teamwork, emergent cooperation is also quite helpful. Certainly, this high-level programming of agents requires the careful implementation of low-level skills as well as the possibility of abstracting quantitative sensor data onto a qualitative level, such that more general *qualitative reasoning* becomes feasible.

Last but not least, the basic layer —the *RoboLog kernel*—, implemented in C++, processes the sensor information of an agent. We incorporated a subset of the low-level skills of the CMUnited-99 simulator team [9] into it. Therefore, a variety of skills is available, that can be used to program higher abilities in Prolog. Essentially, this layer provides an interface between the Soccer Server [2] and (now) SWI-Prolog [10]. We will describe our experiences with the new RoboLog kernel in Section 4.3.

1.2 Logical Control and Procedural Behavior

As said earlier, we lay the focus on the declarativity of the approach for specifying multi-agent behavior. We believe that not all aspects of an agent can be implemented by providing some primitive actions plus possibly hierarchically structured rules (for reactive behavior) and learning techniques for automatically improving the multi-agent system. It should always be possible for a programmer of multi-agent systems to manipulate the intended behavior in an explicit manner. Therefore, we provide logical rules for control purposes and statecharts for the procedural aspects of multi-agent systems in

our approach. Of course, this can be complemented with other more implicit techniques for agent control, e.g. from machine learning or artificial neural networks.

2 Execution Model for Flexible Multi-Agent Scripts

2.1 Predefined Multi-Agent Plans

In [8], a script language for programming multi-agent systems has been introduced. It formalizes some aspects present in belief-desire-intention architectures [7] and combines them with logic programming by means of rules of the following form: If an agent has a certain desire, i.e. aims at a certain goal, and the agent believes that the preconditions for achieving it are satisfied, then the intended plan for the given desire is executed. In general, intended (multi-agent) plans are arbitrary acyclic graphs, where each path corresponds to a sequence of actions for one of the agents that is involved in the whole multi-agent plan. For more details, the reader is referred to [8].

One of the advantages of this approach was, that it enabled programmers to specify multi-agent behavior in an explicit manner by means of simple (first-order) logic programming. But the rigidity of this model proved to be one of its major disadvantages. Once a plan had been selected for execution, it was difficult to interrupt or stop it, when external events or other reasons required this. Only time-limits were applied for the cancellation of scheduled actions. In order to overcome this problem, we changed our script language and also the execution model.

2.2 Interruptible Multi-Agent Behavior

Figure 2 shows the revised execution model. The agent now has internal states which represent its role in the script currently executed, its position within this script, and (possibly) the state of the skill the agent is currently executing. The agent knows e.g. that it is dribbling and has kicked the ball in the last cycle. Therefore it will probably send a *dash* command to the server.

At the beginning of an execution cycle, the RoboLog kernel transfers control to Prolog. First the agent's internal states are tested. If the current world state allows us to continue the running script, the agent updates the internal states and selects the next action for execution. It then returns control to the RoboLog kernel, which in turn sends the command to the Soccer Server and updates the agent's world model. If, however, the continuation of the current script is not appropriate in the present world state, the agent selects a new script, which may be a default script, for execution. This choice is based on external events (e.g. change of play mode) and the agent's knowledge of the world. The internal state is updated to record the fact that the agent starts the execution of a new script. Afterwards, the next action is selected, and the control is returned to the RoboLog kernel.

Multi-agent scripts can still be specified similar to the way described in [8], allowing several agents with different roles to interact in one cooperative action (see also Section 3.2). A clear advantage of the new execution model is its increased flexibility together with the persistence of the behaviors. The (multi-agent) scripts grant a certain

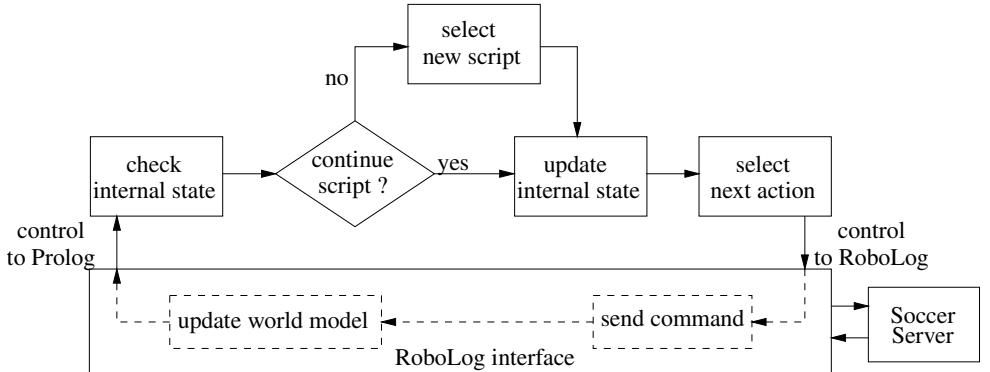


Fig. 2. The execution model.

continuity in the agent's behavior, while the repeated evaluation of the situation at the beginning of each cycle makes quick reactions to changes in the environment possible. But how can the overall behavior of an agent be described, including the transitions from one state to another? The answer will be given in the next section.

3 The Specification of Multi-Agent Scripts by Statecharts

The specification and programming of multi-agent behavior should be as concise as possible. It is desirable that the description language for multi-agent behavior is very flexible, so that the understanding of an agent's behavior and the modification and maintenance of existing code is easily feasible. The best would be, if a specification of multi-agent system can be read as an executable agent program, or at least the translation into running code can be done automatically. Then, no additional step for code verification is necessary.

The specification of multi-agent behavior requires the modeling of activities. In the case of complex actions, i.e. a sequence of actions, it must be possible to model different states an agent can be in. Therefore, it seems to be a good idea to adopt the state machine formalism for our purpose. A well-known means in this context are statecharts, which have a clean semantics—namely via the theory of finite automata—, and they are widely accepted as a means for specifying software. Statecharts are part of the unified modeling language (UML) [6].

3.1 Making Use of Statecharts

The main ingredients of statecharts are states and transitions. In order to make the understanding of statecharts for specifying multi-agent behavior in the RoboCup as easy as possible, we make the assumption, that each transition (including self-transitions) corresponds to exactly one simulator cycle step. What happens along one transition, is described in our execution model (see Section 2). In statecharts, transitions are annotated with events, further conditions and actions.

The most simple agent is just a reactive agent with only one state, but many transitions, that are self-loops. The basic structure of a reactive agent is sketched in Figure 3. There, we identify *events* with the play modes in the RoboCup. The (guard) *conditions*, written in square brackets, may request the actual world model, which is represented in logic, it is the belief of the agent. Only one *action* per transition is expressible by statecharts. But this corresponds quite well to the fact that the Soccer Server permits only one action per simulation step (with few exceptions, namely *change_view*, *say* and *turn_neck*).

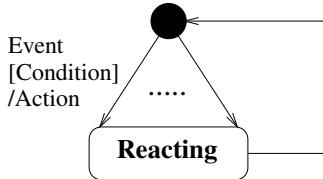


Fig. 3. A purely reactive agent.

More flexible agents, of course, should not only be reactive. They have internal states. Then it becomes possible to express complex actions that require the performance of a sequence of primitive actions, which are provided by the RoboLog kernel. Thus, e.g. before the kick-off, a player should move to its start position only once. Afterwards, the player could scan the field, until the game is actually started. Depending on the situation the agents are in, different more complex scripts can be initiated.

3.2 Multi-Agent Scripts with Iteration

Complex decision processes can be expressed by choice-points, that are pseudo-states, drawn as bullets. This is shown on the right in Figure 4, where one possible action is to perform double passing. This could also be viewed by means of decision trees, which are employed in [1] (see also Section 5.1). Double passing is itself a complex behavior, represented by a composite state. There are two actors in a double passing situation. At first, actor 1 passes the ball to actor 2, then actor 1 runs towards the opponent goal, and expects a pass from actor 2. Look at Figure 5, where this is sketched. For the sake of readability, some labels are omitted.

In principle, double passing is realized by two sequences of actions for the two roles in the script. The sub-script of actor 1 can be seen on the left, and the one for actor 2 on the right of Figure 5. Note that there are two links from the initial state. The whole script corresponds to the acyclic graph script in [8]. As one can see, scripts are now interruptible because of the outgoing edges, where time-limits or other breaking conditions may be annotated. This means, both complex and reactive behavior can be combined easily.

We can go even one step further. One disadvantage of the script in Figure 5 is that the double passing cannot be iterated. Since in [8], scripts correspond to Prolog rules,

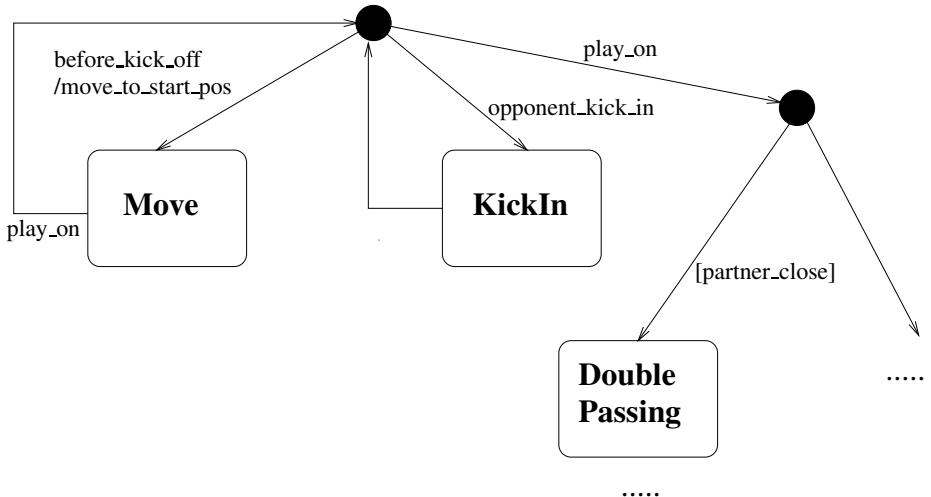


Fig. 4. Overall behavior of a soccer agent.

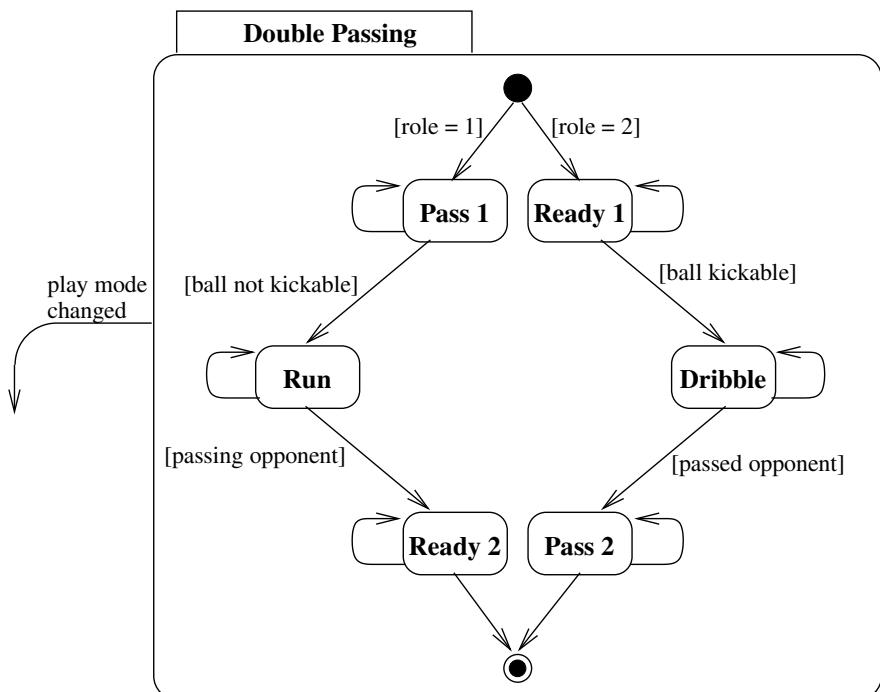


Fig. 5. Double passing script.

namely one non-recursive rule for each role, iteration could not be expressed. But now this is possible, and this is shown in Figure 6. This means, that we again have two actors in the script. On the left-hand side, the two states correspond to the role of actor 1. On the right-hand side, there are two states corresponding to the role of actor 2. But as one can see, they swap their roles at the end, so that continuous passing is possible.

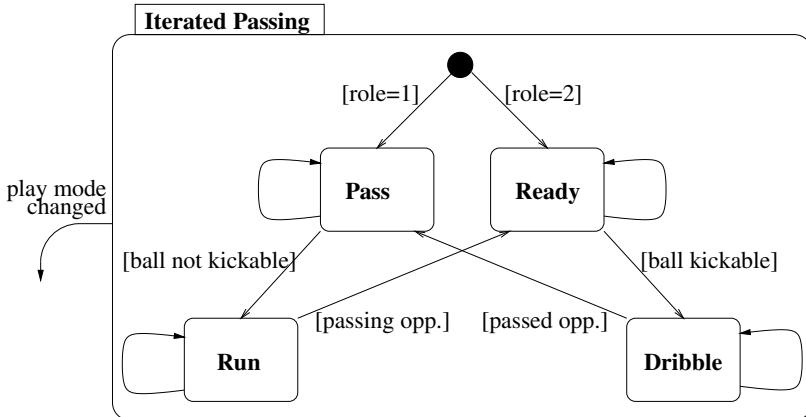


Fig. 6. Iterated passing.

4 Implementing Agents with RoboLog

4.1 Basic Strategies for Soccer Agents

As stated in the previous sections, behaviors and strategies of agents can be specified declaratively. We started implementing soccer agents which behave as follows: If the match is in *play-on* mode, then the agents kick the ball towards the opponents' goal if possible. If the ball is too far away, they try to intercept the ball or just look where it is. Formally, the latter procedure can easily be realized by the following program rules:

```

go_to_ball :-  
    my_side(S),  
    my_number(N),  
    player_interceptionable(S,N),  
    player_interception_point(S,N,X,Y),  
    go_to_point(X,Y,0,100).  
  
go_to_ball :-  
    face_ball.

```

The RoboLog Koblenz team plays according to the following system: There are three defenders, and two attackers, i.e. we play with a double-forward, who go to their

respective home positions, whenever the opponent team has a free kick or a kick-in. The rest are midfielders who try to mark one of the opponent players, if they cannot get to the ball. This basic strategy of our team can be observed in the snapshot of one test game, shown in Figure 7.

Interestingly, with this simple strategy, our team shows some (emergent) cooperative behavior and appears to be competitive with many other simulation league teams. For instance, even without performing a pass explicitly, many situations arise in test games where apparently team-mates are passing the ball to each other, because they try to intercept the ball. In addition, the defenders implicitly build an offside-trap, since they follow the ball if it is played in the opponent half. Nevertheless, the overall performance has been improved by implementing these cooperative behaviors explicitly.

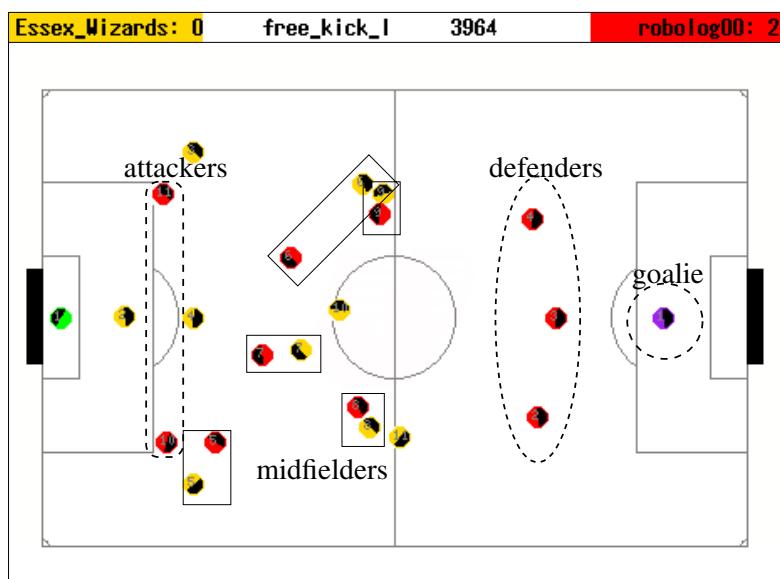


Fig. 7. Basic strategy and positioning.

4.2 Implementation in Prolog

The implementation of state machines with Prolog as specified in Section 3 is straightforward and has been done (manually up to now). The current state is stored in the Prolog database and is updated in each step. Each state transition can then be implemented by the following scheme:

```
behave :-
    current_state(S),
```

```

event(E),
test_conditions(S,E,C),
schedule_action(S,E,C,A),
update_state(S,E,C,A).

```

4.3 The New RoboLog Kernel

The kernel for RoboLog-2000 is built upon the low-level skills of the CMUnited-99 simulator team [9]. The main difference between the RoboLog-99 kernel and the RoboLog-2000 kernel lies in the overall control flow. In RoboLog-99, the Prolog predicates were proven in (quasi-)parallel to the execution of the low level kernel code. This led to uncertain timings and sometimes even to inconsistencies. Now, in RoboLog-2000, Prolog is called just after the internal world model of an agent is updated. After selecting an action, Prolog returns control to the RoboLog kernel. This is indicated in Figure 2 by the dashed arrows and boxes.

5 Related and Future Work

5.1 Related Work

There is (at least) one other related approach that employs logic and also procedural aspects, such as iteration, sequencing and non-deterministic choice between actions, namely *(Con)Golog* [4, 3]. It can be seen as an extension of logic programming, that allows us to reason about actions, their preconditions and effects in the situation calculus. Additional constructs for concurrent execution and handling exogenous action triggered by external events are present in this framework.

The advantage of this approach is that explicit reasoning about actions is possible, which allows agents to plan and extrapolate future behavior. However, it is a more or less single-agent centered system. Our approach provides a clean means to specify multi-agent behavior. Teamwork can be modeled explicitly and adapted easily. Nevertheless, external events and conflicting goals can be resolved by means of logic.

The approach in [1] is also related. It is based on *decision trees*, providing priorities in behaviors. Interruption and termination of behaviors and sequential and concurrent behaviors are also considered in this framework. Explicit communication among agents is sparsely used; coordination and collaboration is a more or less implied property. Although there are several similarities with the approach presented here, our approach is more formal, by making use of state machines and logic, such that we have a clean syntax and semantics for specifying multi-agent systems. Furthermore, in our approach the notion of (internal) states is represented more explicitly.

5.2 Conclusions and Future Work

In this paper, we have shown how multi-agent systems can be specified declaratively by means of logic programming, allowing to program cooperative and reactive behavior. Playing with early versions of our team, consisting of simple *kick-and-run* players, we beat some of the top ten teams of RoboCup-99. This shows that the performance of RoboCup agents depend highly on well implemented low-level skills. On the

other hand, the high-level parts contribute as yet an insignificant part to the success of most of the teams. However, looking at the top ten teams of this year's competition, it seems that they have similar low-level skills—in fact, several teams also made use of the CMUnited-99 library—but different high-level capabilities such as attacking with more than one player or sophisticated defense strategies.

Our further work will concentrate on incorporating (even) more explicit teamwork and more sophisticated strategies into the system. We will follow the goal that explicit agent programming is possible, by combining logical and procedural techniques for the specification and implementation of multi-agent systems. Furthermore, formalizing the development process for multi-agent systems also enables us to apply techniques for the formal analysis and verification of the systems such as model checking.

References

1. S. Coradeschi and L. Karlsson. A role-based decision-mechanism for teams of reactive and coordinating agents. In H. Kitano, editor, *RoboCup-97: Robot Soccer WorldCup I*, LNAI 1395, pages 112–122, Berlin, Heidelberg, New York, 1998. Springer.
2. E. Corten, K. Dorer, F. Heintz, K. Kostiadis, J. Kummeneje, H. Myritz, I. Noda, J. Riekki, P. Riley, P. Stone, and T. Yeap. *Soccerserver Manual*, 5th edition, May 1999. For Soccerserver Version 5.00 and later.
3. G. De Giacomo, Y. Lespérance, and H. J. Levesque. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In M. E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 1221–1226, Nagoya, Japan, 1997. IJCAI Inc., San Mateo, CA, Morgan Kaufmann, Los Altos, CA, Volume 2.
4. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.
5. J. Murray, O. Obst, and F. Stolzenburg. RoboLog Koblenz. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer WorldCup III*, LNAI 1856, pages 628–631. Springer, Berlin, Heidelberg, New York, 2000. Team description.
6. Object Management Group, Inc. *OMG Unified Modeling Language Specification*, 1999. Version 1.3, June 1999.
7. M. P. Singh, A. S. Rao, and M. P. Georgeff. Formal methods in DAI: Logic-based representation and reasoning. In G. Weiss, editor, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*, chapter 8, pages 331–376. The MIT Press, Cambridge, MA, London, 1999.
8. F. Stolzenburg, O. Obst, J. Murray, and B. Bremer. Spatial agents implemented in a logical expressible language. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer WorldCup III*, LNAI 1856, pages 481–494. Springer, Berlin, Heidelberg, New York, 2000.
9. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer WorldCup III*, LNAI 1856, pages 35–48. Springer, Berlin, Heidelberg, New York, 2000.
10. J. Wielemaker. *SWI-Prolog 3.3 Reference Manual*. University of Amsterdam, The Netherlands, 2000.

Bridging Gap between the Simulation and Robotics with a Global Vision System

Yukiko Nakagawa¹, Hiroshi G. Okuno^{1,2}, Hiroaki Kitano^{1,3}

¹ Kitano Symbiotic Systems Project, ERATO, JST
yuki@symbio.jst.go.jp

² Science University of Tokyo okuno@nue.org

³ Sony Computer Science Laboratories, Inc. kitano@csl.sony.co.jp

Abstract. A wide gap lies between the simulation league and real robot leagues of RoboCup, in particular in applications. In the simulation league, the soccer server maintains all data on the pitch and feeds such data to applications via networks. Therefore, several applications such as 2D- and 3D-viewers, commentary systems have been developed. In this paper, we propose that using a global vision system can bridge the gap of applications. A global vision system, which is usually used in the small size league, is extended to serve as a data feeder. Its main issues include missing object identification (due to occlusion and feature extraction failure), errors in player identification (due to mis-assignment of objects among continuous frames), detection of stoppage, and elimination of unnecessary scenes. To cope with these issues, the proposed system uses estimation and scene analysis. We also present an extension of the soccer server protocols to feed state specific to the small size league. The resulting system succeeded in feeding data of the final game of RoboCup'99 to a 2D-viewer and a commentary system.

1 Introduction

A wide gap lies between the simulation league and the real robot league in many aspects. For example, location and orientation data of each player is completely represented in the former, while it often contains errors in the latter. The speed of the ball may sometimes travel much faster in the real robot league than in the simulation league. Their regulations differ, too.

In the simulation league, the soccer server[1][2] maintains all data on the pitch and feeds such data to applications. Therefore, several applications such as a simple 2D-viewer[2], a 3D-viewer[3][4], commentary systems[5][6], and analyzer have been developed in addition to soccer teams based on the soccer server.

However, neither the soccer server (data feeder) nor applications has been developed for the real robot league, although they are expected to play an important role in visualizing, analyzing and evaluating a game played by robot players. One way to bridge this gap is to develop a data feeder that is equivalent to the soccer server in the simulation league. When real world information is carefully designed, we have a chance to develop various applications by integrating of the

simulation and robotics. For example, a simple broadcasting system using viewers and commentary systems via the network, strategy evaluation using existing analyzers[7] for the simulation league.

In this paper, we demonstrate bridging the gap using a global vision system for the small size league. Before discussing the global vision system, we present what problems lie on the gap in Section 2. Then details of the global vision system to solve the problems are described in Section 3. To evaluate our approach, a simple broadcasting system has been realized using applications of the simulation league via network. A video of the final game of the small size league in RoboCup'99 has been used as a benchmark in Section 4. Based on the results, we discuss our approach and future work in Section 5, and conclude in Section 6.

2 The Gap between Simulation and Real Robot

2.1 The Gap of Data and Method to Obtain Information

To understand the gap between the simulation league and the real robot league using vision, a comparison of data and methods are given in Table 1. The simulation league has different input information formats for the player applications and for the viewer applications. Corresponding to the input information, there are overhead camera(s) of the global vision systems and on-board camera(s) of the local vision systems in the real world. The data part in Table 1 shows that

Table 1. Comparison about Data and Method to Obtain Information

	Simulation league		Real robot leagues	
	for player client	for monitor client	global vision ¹	local vision ²
Data				
Data Source	simulated data	simulated data	image sequence	image sequence
Data error	none	none	exist	exist
Sampling rate	10times/sec.	10times/sec.	30frames/sec.	30frames/sec.
Perspective	side view	bird eye view	bird eye view	side view
Orientation	relative	absolute	absolute	relative
Range of view	local	global	global	local
Method to Obtain Information				
Object position	calculation	calculation	object extraction	object extraction
Object missing	none	none	exist	exist
Object consistency	calculation	calculation	tracking	tracking
Object confusion	none	none	exist	exist
Player status	none	command translation	scene analysis	scene analysis
Quality of player status	-	certain	ambiguous	ambiguous
Game status	from referee	from referee	scene analysis	scene analysis
Quality of game status	certain	certain	ambiguous	ambiguous

1: Its use is restricted only to the small size league.

2: It can be used by all robot leagues.

the player applications can use information of the local vision, and the viewer application can handle information of the global vision, respectively. It is noted that the referee gives game state to both applications in the simulation. In addition, the referee uses global information to know the game state transition. Since,

we think the global vision is superior to the local vision to obtain the global information. We make clear a gap of game state transition between simulation and real world using global vision in the next section.

2.2 The Gap of Game State Transition

Fig.1 shows the game state transition of the small size league.

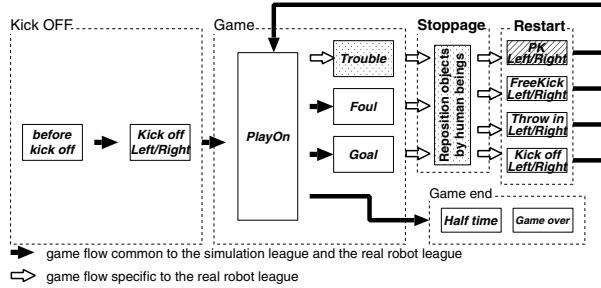


Fig. 1. Game State Transition of The Small Size League

Dash line rectangles represent the sub-state of game state transition, while solid line rectangles show details of the game state. The sub-state of game state is common to all leagues, while each league shows different details. In Fig.1, arrows indicate the game state transition. A black arrow is the common game flow between the simulation league and the real robot league, and a white arrow points out game flow specific to the real robot league.

The different details arise from the regulation (rectangle filled with slash) and the gap between the simulation and real world (rectangles filled with dots).

It is difficult to obtain the game state using object tracking only. The soccer server analyzes the game state transition using position of ball and players. By applying such scene analysis to data of objects from an image sequence, the game state can be obtained in the real world. We think that the soccer server protocol can absorb the difference of regulations using some additional game state. However, to bridge the gap, specifics of the real robot league has to be eliminated to connect the application of the simulation league.

3 The Global Vision System

Fig.2 (1) and (2) are the hardware design and task flow to feed real world information into the applications of the simulation league. For the global vision system of the small size league, we design hardware consisting of an overhead camera (XC-711R, Sony) and a PC with a frame grabber (GV-VCP/PCI, I/O Data). The system can connect to other computers on which the applications of the simulation league are installed.

The task flow generating an image processing sequence is shown in Fig.2 (2). Before kick off, we prepare some color ranges for object extraction in the *prepare*

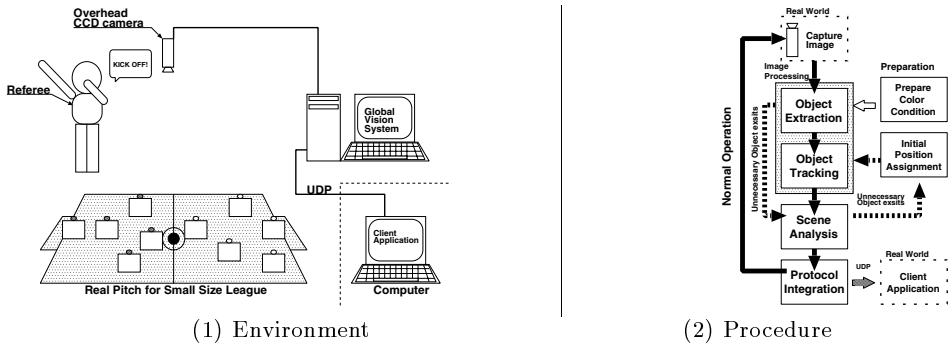


Fig. 2. Global Vision System Environment and Procedure

color condition part, then each initial location of robots is assigned for *object tracking* manually in the *initial position assignment* part in Fig.2 (2). During the game, an image sequence is taken by the hardware in the *capture image* part. The image sequence is captured at 30 frames per second, and the size of the image is 320×240 . A pixel of the image is expressed by 24 bit RGB color model, but it is translated into HSV color model to absorb small differences in lighting conditions around the pitch. The image processing obtains consistent object data about location and orientation from the image sequence using *object extraction* and *object tracking*. Based on the data, the *scene analysis* leads to the game state and player state. The *scene analysis* is a set of if-then rules for data sequence and location of objects. If the game state is specific to the small size league, then the loop is back to the *initial position assignment* recovering the object assignment manually in our system. It means that the *object tracking* process is skipped during a game stoppage. In the case of normal operation, the last procedure forms the same protocol as the soccer server from the obtained information. We shall discuss the details of image processing, scene analysis and interface adjustment in following sections.

3.1 Object Extraction

We use conventional color extraction and labeling using given a range of hue, saturation and value to extract blobs of the same color in the current frame. The extracted blobs are sorted by given a range of number of pixels of object such as target object candidates and others (noise and unnecessary objects). The noise is small blob and pixels in narrow line arising from random reflection. It is removed during this procedure. The unnecessary objects are mainly parts of human's body. Both of them can be distinguished by the given number of pixels (area) and ratio of width and height. This task flow jumps to the scene analysis when unnecessary objects appear in the current frame. In the normal operation, the target objects are counted to know if there is a missing or extra object. It is impossible to know which is the missing object or extra one by object counting. Thus we should identify each object candidate to find out the missing or extra objects tracking.

3.2 Object Tracking

To keep continuity, the *object tracking* procedure assigns each object in the current frame to an object in the previous frame. The method for object tracking is to search the objects in the previous frame using a given range and location of the object in the current frame. If the procedure finds a unique object in the previous frame, then the object in the current frame is assigned to the previous one. When multiple objects are found in the range, the estimated location and orientation of each candidate is applied to identify each object. The estimation uses location and orientation of each object in three previous frames to obtain speed, acceleration and direction. Using speed and direction, procedure of the estimation obtains path, location and orientation of object by the current frame. Ambiguous objects in the current frame are distinguished if it appears on the path.

After object assignment, orientation of robot is calculated. There are various teams, such as an omni directional mobile robot team, and a team putting several markers on the top of each robot. It is hard to assume for the general purpose global vision system that the robot moves in a restricted direction and each marker shows front direction.

Therefore, front direction is calculated using the object location in the current and the previous frames, and rotation is calculated using color patch positions, if robots have them , when the object is staying in the same place. It should be noted that this direction calculation differs significantly to the global vision system to control the robots.

Missing object can be found when all objects in the current frame are assigned. If there are missing objects, they are compensated with the estimated location and orientation. In other case, extra objects are removed in this procedure. At last, location and orientation of identified objects are prepared for all players and the ball.

3.3 Soccer Scene Analysis

After consistent objects are extracted or unnecessary objects are detected, various scene analysis techniques are applied to these objects to find state of the game and players. We define objects not concerned with the game is unnecessary object. For scene analysis, we summarize the game state of the small size league in Table 2. Some aspects of game state are the same as the simulation league, but the others are different. The name of the game state refers to the play mode of the soccer server. Therefore each game state has “PM_” to indicate the game state. The scene analysis uses a sequence of location of objects to obtain the game state using if-then rules. For example, we describe the procedure of the scene analysis of “PM_PKLeft” as follows:

1. If the previous play mode is one of following: **before kick off**, **free kick**, **kick in**, **foul**, **pause**, and **human**, then
2. initial positions of objects are given manually to track object (See Fig.2 (2)),

Table 2. Game Status of The Small Size League

Play mode for the simulation league	
PM_BeforeKickOff	before kick off
PM_TimeOver	game over
PM_PlayOn	normal situation of game
PM_KickOff_Left/Right	kick off from left/right team
PM_KickIn_Left/Right	throw in from left/right team
PM_FreeKick_Left/Right	free kick from left/right team
PM_AfterGoal_Left/Right	after goal restart from left/right team
Additional play mode for the small size league	
PM_PKLeft and PM_PKRigh	PK (kicker is left/right team)
PM_FirstHalfOver	Half time
PM_Pause	Pause (Stoppage for replacing robots or ball)
PM_Human	Human being coming into pitch
PM_FOUL_ChargeLeft/Right	charge by left/right team
PM_FOUL_PushLeft/Right	left/right team push left team goalie
PM_FOUL_MultipleAttackerLeft / Right	multiple left/right team players are in the penalty area for left team
PM_FOUL_BallOutLeft/Right	ball is kicked out of pitch by left/right team

3. check the position of objects in PK position and kicker team,
4. if all objects satisfy condition of position, then
5. the play mode is PK.

The **kick state** and the **stand state** are obtained by the scene analysis. For example, the scene analysis to obtain the **kick state** is described as follows:

1. When a robot and the ball exist within a given range, the location of the robot in the three previous frames are kept for evaluation.
2. The procedure understands movement of the robot as “kick” when speed and direction of the ball changes in the next frame.

It should be noted that the scene analysis has to be made simple to save processing time. All of scene analyses have to finish before the next image is captured by the overhead camera.

3.4 Interface Adjustment

This procedure is started by a request from an application via the network. The protocol for the viewer application is filled with the required information from the real world. At the same time, the procedure eliminates unnecessary scenes using the obtained game state to identify scenes specific to the real robot league. Moreover, the procedure adjusts the sampling rate of the image sequence to the application of the soccer server.

4 Experiments

The proposed system has been implemented on a PC. It connects to a SPARC station to feed the viewer applications the observed information over the network using UDP. The installed viewer application is a 2D viewer. The 2D viewer can show location and orientation of objects in the real pitch. We use the video of the final game of the small size league in RoboCup 1999 to evaluate the system (See Fig.3).

4.1 Image Processing

The object extraction is evaluated using a video tool and the log player with the 2D viewer. The state of the game and players for the small size league are added to the log player and the 2D viewer. The video tool shows frame number and an image frame. The image sequence includes well conditioned image and ill conditioned image. The well conditioned image means that the image shows all objects such as ball and player robots without occlusion. The ill-conditioned image defined as an image including occlusion or missing object. We show a well-conditioned image and the same scene shown by the 2D viewer in Fig.3 (a), (b), ill-conditioned scene in (c) and unnecessary scene which humans enter into the pitch to reposition some robots and the ball after goal in (d).



Fig. 3. Visual Scene and 2D viewer

Substitution by the estimation is applied in the case of occlusion and missing feature. It failed when noise having similar feature is detected instead of missing information. This failure arises from wrong assignment of objects. Besides of this

failure, substitution of location is completely succeeded in the game scene such as (c) in Fig.3. The orientation of objects is sometimes missed near wall. The object assignment between continuous frames has a success rate of 82%. This rate is given by (1).

$$r = s/F, \quad (1)$$

where r is the rate, s is number of succeeded frames which all objects are assigned between two frames, and F is number of couple of frames to assign objects.

4.2 Scene Analysis

The scene analysis has been implemented for each state of the game and players in Section 3.3. We can check the result of the scene analysis using log player of the simulator and message on the top and color of players in the 2D viewer. The game state is shown by text in the 2D viewer. The 2D viewer also shows black circle (the **kick state**) and colored circle (the **stand state**) for the player state.

The result of the scene analysis is shown in Table.3. The player states are checked about ten in a lot of performances. The game state is checked for the first half of the game. The system misunderstands the **pause state** when

Table 3. Result of The Scene Analysis for Game and Player Status

	Status	Result	Human Judgement
Player Status	Kick	10	10
	Stand	10	10
Game Status	Play on	19	19
	Human	19	19
	Pause	17	19
	PK	3	3
	Score	20	8
	Multiple Defender	2	2
	Charge	20	3

Before kick off, kick in, free kick are given by manual.

humans re-enter into the pitch after set up. The score is higher compared with real judgment, because the goal area has a strong shadow. The system fails when noise appears in the center circle after goal. The number of charge is larger than what were called. Although, there are a lot of potential charges, but some of them are handled as a foul. This may be improved by applying more restricted conditions. The other game state doesn't appear in this game.

5 Discussion, Related Work and Future Work

To date, there is no report to bridge the simulation leagues and the real robot leagues in RoboCup.

Our demonstration indicates various possibilities to improve integration of AI (simulation) and robotics (real robots). Introduction of the applications of

the simulation league improves robotics in many aspects. For example, the log player facilitates the improvement of robot movement and strategy using existing analyzer for the simulation league.

Our approach also improves applications of the simulation league including the soccer server. To handle real world tasks, the play modes and the player state of small size league will be added to next revision of the soccer server (version 5.28 or later). This improvement is the first step to bridge the gap between the simulation league and the real robot league. Moreover, this improvement shows possibilities of entertainment application of robot soccer competition such as broadcasting system via network.

In research on global vision, teams try to control robots in the small size league. Our work is significantly different from the global vision systems to control robots, but it can be compared with them with respect to object extraction and tracking. Some teams have special hardware to extract objects [8][9][10] to reduce processing time. Others extract objects using a software approach[11][12]. As for the accuracy of the extraction and the tracking, our approach is almost the same as the global vision systems to control robots. However, introduction of image processing hardware should be examined to decrease the processing time. Our goal is to acquire the same information about all players, while the global vision system to control robots obtains more detailed information on their own team than opponents. In addition, our approach of image processing can also be applied to multiple overhead cameras system.

To bridge the gap, we mentioned a method to handle errors in visual information and scene analysis work on the image sequence using global vision system. The same principle can be applied to the middle size league vision system. In this case, information of the middle size league can be fed to the player applications of the simulation. Current robot in RoboCup is just a player. When global and local information can be used in a system of autonomous mobile robot, the robot will be an audience, a referee, a coach besides of a player in the future. It is important to obtain state transition using scene analysis for such a robot or system working in the dynamic environment. The state transition can give trigger to change processing for expecting objects in the data depending on situation. If such a mechanism can be realized, it can help smart operation. We found that the visual scene analysis has a limitation of obtaining the game state in the experiment. We should handle other perceptive inputs to obtain certain state transition. In the small size league, the referee's voice, the whistle sounds and the visual scenery of the game can affect the change of the game state. We should utilize sounds to know more precise state in this case. In the future, we should integrate other perceptive inputs and vision to improve our system.

6 Conclusion

The major contribution of this work is bridging the gap between the applications of the simulation league and the global vision system for the small size league by generating log of the game.

Much research on global vision systems have been performed in the small size league, while this is the first study to clearly demonstrate that the simulation

league can handle information from the real world. We found that the information for the simulation league can be obtained by image processing to cope with errors of visual information and the scene analysis of the real world. We have clear evidence that the information of the real world, when designed carefully, can be handled by the application software of the simulation league.

Acknowledgment

This paper owes much to the thoughtful and helpful comments of Dr.K.Tanaka-Ishii, Dr.I.Frank and Dr.I.Noda.

References

1. I.Noda, H.Matsubara, K.Hiraki, I.Frank, "Soccer Server: A Tool for Research on Multi-Agent Systems", Applied Artificial Intelligence, Vol.12, No.2-3, pp.233-251, 1998
2. Soccer Server Manual: <http://www.dsv.su.se/~johank/RoboCup/manual/ver5.1release/browsable/main.html>
3. A.Shinjoh, S.Yoshida, "Autonomous Information Indication System", IJCAI-99 Workshop on RoboCup, pp.199-204. 1999
4. A.Shinjoh, S.Yoshida, "A Development of Autonomous Information Indication System for RoboCup Simulation League", Proc. of Systems Man and Cybernetics, Vol.6, pp.756-761, 1999
5. E.Andre, J.Binsted, K.Tanaka-Ishii, S.Luke, G.Herzog, T.Rist, "Three RoboCup Simulation League Commentator Systems", AI Magazine, Spring, pp.57-66, 2000
6. <http://coach.isi.edu>
7. K.Tanaka-Ishii, I.Frank, I.Noda, H.Matsubara, "A Statistical Perspective on the RoboCup Simulator League : Progress and Prospects", Proc. of RoboCup Workshop IJCAI99, pp.217-222, 1999
8. J.Akita, "Linked99: Simple Soccer Robots with High-Speed Vision system Based on Color Detection Hardware", Team Descriptions small and middle leagues of RoboCup-99, pp.53-58, 1999
9. R.D'Andrea, J.W.Lee, "Description of Cornell BigRed Small League RoboCup Team", Team Descriptions small and middle leagues of RoboCup-99, pp.14-23, 1999
10. F.Yong, B.Ng, S.Quek, K.Chuo, K.Chong, D.Chou, J.Y.Toh, J.Tan, C.Tan, T.Teo, "LuckyStar", Team Descriptions small and middle leagues of RoboCup-99, pp.9-13, 1999
11. T.Naruse, T.Takahashi, M.Nagami, Y.Nagasaki, K.Murakami, Y.Mori, "OWARI-BITO Team Description", Team Descriptions small and middle leagues of RoboCup-99, pp.59-63, 1999
12. M.Veloso, M.Bowling, S.Archim, K.Han, P.Stone, "The CMUnited98 champion small robot team", RoboCup-98: Robot Soccer World Cup II, H.Kitano Eds., Springer, pp.77-92, 1998

Real-time Estimating Spatial Configuration between Multiple Robots by Triangle and Enumeration Constraints

T. Nakamura*

M. Imai

M. Oohara

T. Ogasawara

A. Ebina

H. Ishiguro**

Graduate School of Information Science, Nara Institute of Science and Technology
Takayama-cho 8916-5, Ikoma, Nara Japan 630-0101

*E-mail:takayuki@is.aist-nara.ac.jp

WWW:<http://robotics.aist-nara.ac.jp>

** Dept. of Computer and Communication System, Wakayama University
Sakaetani 940, Wakayama, Wakayama Japan 640-8510

Abstract. In the multi-agent environment, it is important to identify position and orientation of multiple robots for accomplishing a given task in cooperative manner. This paper proposes a method for estimating position and orientation of multiple robots using multiple omnidirectional images based on geometrical constraints. Our method reconstruct not only relative configuration between robots but also absolute one using the knowledge of landmarks in the environment. Even if there are some obstacles in the environment, our method can estimate absolute configuration between robots based on the results of self-localization of each robot.

1 Introduction

In order that multiple robots operate successfully in cooperative way, such robots must be able to localize themselves and to know where other robots are in a dynamic environment. Furthermore, such estimation should be done in real-time. An accurate localization method is a key technology for successful accomplishment of tasks in cooperative way. Most of conventional localization algorithms extract a small set of features from a single robot's sensor measurements. For example, landmark-based approach[1], which is very popular, scan sensor readings for the presence of landmarks to estimate a single robot's position and orientation.

On the other hand, in case that there are multiple robots in the environment, the localization problem can be easily resolved. Since robots in the multiple robot system can share the observations by communicating each other, each robot in such multiple robots can utilize redundant information for localizing itself. Therefore, such robots can solve the localization problem more easily than a single robot does. However, in multiple robot system, it is difficult to identify other robots by using only visual information.

In order to realize such identification and localization capability, Kato et. al [3] developed a relative localization algorithm for multiple robots equipped with ODVS. This method is based on identifying potential triangles among any three robots using the simple triangle constraint. After potential triangles are identified, they are sequentially verified using information from neighboring triangles.

However, their system is not really autonomous decentralized one, because their method strictly assumes that multiple robots communicate each other in order to reconstruct a relative configuration. Furthermore, the processing time for estimation is apt to be long, because their method uses only one geometrical constraint.

In this paper, in order to enhance functions of autonomous decentralized system, our method gives a self-localization capability to a single robot among multiple robots, in addition to capability of estimating a relative configuration from the sharing observation. Due to self-localization capability, our method can use one more constraint which is called "enumeration constraint" to make our algorithm fast and robust, and can reconstruct absolute configuration between multiple robots in the environment. This paper proposes an enhanced version of method for identifying and localizing robots in the environment using omnidirectional vision sensors, on the basis of Kato et. al's method [3]. Even if there is an obstacle in the environment where the multiple robots are located, our method can estimate absolute configuration between robots in the environment with high accuracy.

2 Prerequisites

Before details about our method is described, several prerequisites must be stated.

Omnidirectional image sensor We assume that each robot has an omnidirectional vision sensor [2] which can capture 360° view image at a time. Hereafter, we call this sensor "ODVS." Further, we call an image taken by ODVS as "ODI."

Robot The total number of the robots is n . n robots are located within a field. Each robot has an omnidirectional image sensor. Each robot has an visual identifier such that other robot may easily observe it. However, it can't always view other robots, because the robot can't detect other robots due to failure of image processing or due to an obstacle. This assumption is very reasonable in real situation. Furthermore, each robot can't precisely measure the distance to the robot or the landmarks, because such rough distance is estimated by calculating the distance from the center of the omnidirectional image to the center of the region of the robot or the landmark.

Landmarks The environment has several landmarks which are easily identified. Such landmarks are painted some colors in order that they are easily identified. Each robot among a group of robots has an visual identifier as a marker which is also painted a particular color.

Task A task of our system is to identify all of the robots and to know absolute spatial configuration between them in the environment. Here, absolute spatial configuration means positions and orientation for all robots in the environment.

3 Overview of our method

First, each robot performs self-localization based on observing some landmarks in the environment. There have been many methods [1] for determining its position and orientation in the environment, which is called "self-localization." Since our robots equip with ODVSs, we utilize the ODVS-based method [4] and the landmarks in the environment for self-localization.

Second, our system performs reconstruction of relative configuration based on observed azimuth angles and the results of self-localizations. When reconstructing the relative configuration, geometrical constraints are utilized for determining relative spatial structure between robots. These geometrical constraints are described in the following section.

Finally, our system estimate absolute configuration based on the reconstructed relative configuration. To determine absolute configuration, the origin and orientation of the relative configuration are calculated based on the best result of self-localization. Then, in order to determine the size of the relative configuration in the environment, our system observes two landmarks whose positions are known using ODVSs. Such observation limits to the position of each robot in the environment. As a result, the position and orientation of the robot i in the environment (x_i, y_i, θ_i) can be estimated.

3.1 Triangle constraint

One of the key constraints used in our method is a fact that three relative angles between three robots always add up to 180° . That is, each robot corresponds to a vertex of a triangle. Therefore, the angles between three robots must add up to 180° . We call this rule “triangle constraint.”

3.2 Enumeration constraint

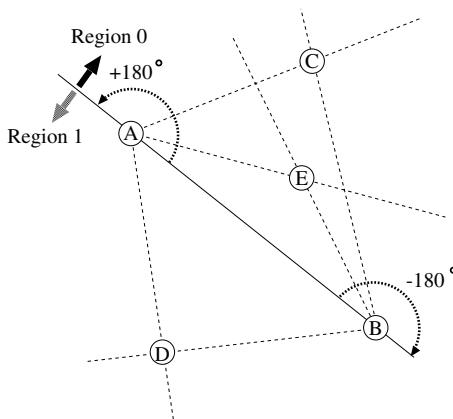


Fig.1. Enumeration constraint

Positions of robots can be qualitatively described by relative positions with respect to a line passing over two robots in the environment as shown in **Fig.1**. This line divides the environment into two regions called *Region0* and *Region1*. Therefore, the position of a robot can be represented by which region the robot is located in. Let five robots be A, B, C, D and E . For example, the position of E is represented in *Region0* defined by line AB . That is, this line limits possible region where E is located. Furthermore, this line limits the maximum number of robots which a robot can observe in a part of its ODI corresponding to *Region0*.

When several robots are located in the environment as shown in **Fig.1**, the number of robots which robot A or B can observe in its ODI is less than or equal to the number of robots which are actually located in *Region0*. For example, both of robot A and B observe *Region0* as the part of its ODI which corresponds to a 180° field of view. We define a counterclockwise/clockwise angle from the base axis as a positive/negative angle. In this paper, this base axis would be regarded as a sight line. A sight line is a hypothetical line from an observing robot to an object which the robot tries to identify. For example, the base axis is a segment AB which corresponds to a sight line from the observing robot A/B to the observed robot B/A (see **Fig.1**). According to this definition, with respect

to robot A , the angle of the sight line ranges from 0° to $+180^\circ$ in *Region0*. On the other hand, with respect to robot B , the angle of the sight line ranges from 0° to -180° in *Region0*. Robot A and B also observe other robots in such their 180° field of view. The sight lines from the observing robots A and B to other robots is described by a dotted line as shown in Fig.1.

Suppose there are two robots in the *Region0*. As mentioned before, the number of other robots observed by robot A or B must be less than or equal to two. Furthermore, both of robot A and B must be located on the boundary between *Region0* and *Region1*. That is, if robot B observes more than three robots in *Region0*, robot B wouldn't be located on such boundary.

In this way, the position of the robot would be limited by enumerating other robots in its 180° field of view. We call this constraint “**enumeration constraint**.” This constraint can be used to eliminate impossible configuration between robots. In this paper, the number of robots which are actually located in *Region0* is estimated based on the result of self-localization.

If there is an obstacle among robots, some of robots might fail to see other robots in its ODI. In practice, there are some limitations in observation. For example, a ODVS cannot observe an object locating in distance, because the capability of image processing has its limitation. Under these situations, the number of other robots observed by a robot is less than the number of robots which are actually located. Since such situations would be judged as inequality in the enumeration constraint, this constraint can work well in anytime.

4 Estimating relative configuration between robots

Our method for estimating relative configuration between robots consists of 5 steps.

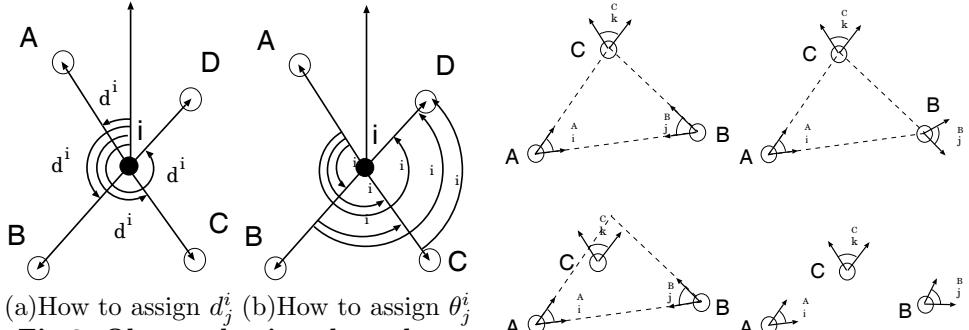


Fig.3. Impossible triangles

1. Each robot i observes the other robots using its ODVS and measures every azimuth angles d_k^i of robots from some base axis. Here, $i (= 1 \sim n : n(\geq 3))$ shows an robot ID and $k (= 1 \sim m : 2 \leq m \leq n - 1)$ shows an index of an azimuth angle to k th robot. In this paper, we select a base axis such that it coincides with either of two axes which are defined to represent an ODI. Fig.2.(a) shows how to assign d_k^i in the case of $n = 5$.
2. Relative angles between two observed robots, θ_j^i are calculated in terms of each robot i as follows:

$$\theta_j^i = |d_k^i - d_{k'}^i|,$$

- where j shows a pair of two observed robots k and k' . This correspondence is stored in a table. **Fig.2.(b)** shows how to assign θ_j^i in the case of $n = 5$.
3. By checking the triangle constraint, all combinations of three different robots (i^p, i^q, i^r) are obtained among all relative angles $\theta_j^i (i = 1 \sim n)$ such that $\theta_{j_\alpha}^{i^p} + \theta_{j_\beta}^{i^q} + \theta_{j_\gamma}^{i^r} = 180 \pm \Delta E^\circ$. Here, ΔE shows an allowable error, that is reflected as an error caused by image processing. The resulting triplets that satisfy the triangle constraint are candidates for computing the relative positions of the robots.
 4. The resulting triplets from the **Step 3** include impossible triangles. These impossible triangles can be classified into four different types, as shown in **Fig.3..** Enumeration constraint is checked with respect to the resulting triplets, in order to eliminate the triplets which form impossible triangles. Such a pruning process is executed as follows:
 - Choose one of the triplets which just satisfy the triangle constraint. Let this triplet be $(\theta_0^A, \theta_j^B, \theta_k^C)$, suppose that A, B and C might be the vertices of a triangle (see **Fig.4**).
 - Based on the result of self-localization, estimate the number of robots which are actually located in half space associated with the line passing over A and B . Let this number be N_R .
 - With respect to each of robot A and B , enumerate the number of the other robots which appear in its 180° field of view. Let this number be N_{OR} . Then, check if $N_{OR} \leq N_R$ is satisfied.
 - With respect to other two pairs of robots (B, C) and (C, A) , the same procedure is applied.
 5. The resulting triplets from the **Step 4** may still include impossible triangles. Therefore, this step evaluates neighboring triangle candidates and eliminates impossible triangles.
 - Choose one of the triplets. Let this triplet be $(\theta_0^A, \theta_j^B, \theta_k^C)$, suppose that A, B and C can be the vertices of the triangle (see **Fig.4**).
 - Examine each segment of the triangle ABC (hereafter, $\triangle ABC$). Let each endpoint of this segment be vertex A and B , respectively. Check whether vertex A has the other relative angle θ_1^A or not, in such a way that the relative angle θ_1^A can be one of angles of a triangle which shares the segment AB but is except $\triangle ABC$. If the vertex A has such relative angle θ_1^A , there must be a new vertex of a triangle except C and B . Let this new vertex be D . Therefore, such a new triangle is $\triangle ABD$ (see **Fig.4**).
 - If there is $\triangle ABD$, check if the new vertex D can be one of angles of neighboring triangles which are adjacent to $\triangle ABC$ and $\triangle ABD$. That is, check if the neighboring triangles $\triangle BCD$ and $\triangle ACD$ can be generated or not.
 - In addition, check whether vertex A has the other relative angle θ_2^A or not, in such a way that this relative angle can be one of angles of the triangle which share the segment AB . Then, check if the neighboring triangles can be generated or not, in such a way that such triangles share vertices and segments with the verified triangles.
 - In the same way, this process is applied to all triangle candidates acquired by the enumeration constraint. Finally, a relative configuration between robots is obtained as a set of the verified triangles.

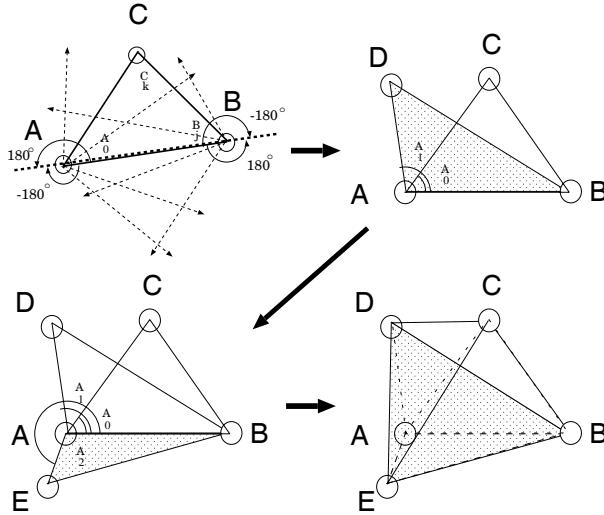


Fig. 4. Iterative verification

5 Estimating absolute configuration between robots

The result of estimating relative configuration between robots does not directly reflect the absolute configuration between robots which shows distances between two robots, positions and orientations of multiple robots in a world coordinate system. This world coordinate system is attached to the environment where such robots are located. It is because relative configuration is reconstructed only by a set of observed azimuth angles which contain no measurement value in the world coordinate system.

The reconstructed relative configuration is similar to the absolute configuration as shown in **Fig.5(a)**. In order to reconstruct the absolute configuration between robots, it is necessary to determine the size, and position and orientation of the relative configuration with respect to the world coordinate system. In this paper, it is assumed that each robot can estimate its position and orientation in the world coordinate system with a certain degree of accuracy. Therefore, in order to determine an origin of relative configuration, we utilize the best result of self-localization which one of such multiple robots get. We call such a robot "base robot." The origin of the relative configuration is defined as the position of the robot which can get the best result of self-localization, that is a base robot. Here, the best result means a result with the highest accuracy. The orientation of the relative configuration is defined as the orientation of the base robot. The position and orientation of the base robot are used for determining those of the relative configuration in the world coordinate system.

If the context of the task given to the multiple robots is relevant to the robotic soccer, the goalie robot would be used as the base robot. It is because the goalie robot provides more precise result of estimating position and orientation than the other robots do. (See **Fig.5(a)**) Since the goalie robot can see landmarks very well, it can obtain its position and orientation with high accuracy.

Next, let's consider how to determine the size of the relative configuration in the environment. By observing two landmarks in the environment whose position are known, the position of the robot i in the environment (x_i, y_i) can be limited

on a curve. In the context of robotic soccer task, such two landmarks can be regarded as own and opponent goals. So, such curve can be defined based on the sine formula as follows:

$$(x_i + \frac{L}{2 \tan \theta_i})^2 + y_i^2 = \left(\frac{L}{2 \sin \theta_i} \right)^2,$$

where L and θ_i shows the distance between two landmarks in the environment and the angle between two sight lines to two landmarks observed by robot i , respectively. The position of each robot except the goalie is limited on an arc as shown in **Fig.5(b)**. In order to determine the positions of the robots except the base robot in the environment, an additional constraint is required. As such a constraint, the sight lines to the other robots can be utilized. The sight line to the other robot is equivalent to a halfline to the other robot whose starting point is the position of the goalie. Therefore, the position of each robot can be obtained as an intersection of the arc and the halfline. Here, instead of calculating the position of the robots directly, the length between the position of the base robot and the i th intersection SL_i is calculated. Furthermore, the length of the segment between the base robot and robot i is computed on the basis of the length of a segment. Let such computed length be SL_i^R . Under this preparation, the ratio $\frac{\sum_{i=1}^n \frac{SL_i}{SL_i^R}}{n}$ corresponds to the scale ratio between the relative and absolute configurations. Using this scale ratio, the size of the absolute configuration is determined.

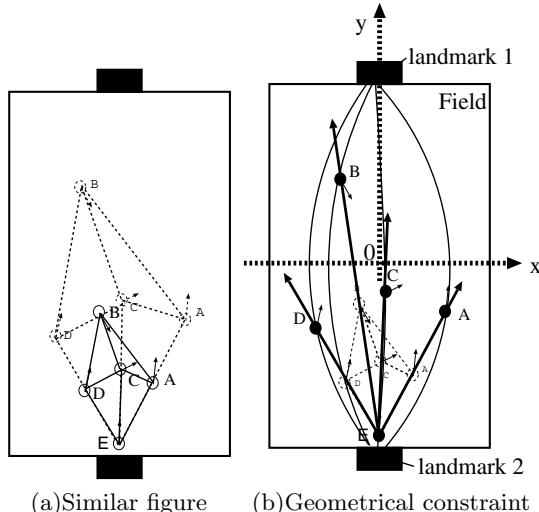


Fig. 5. How to estimate absolute configuration between robots

6 Experimental results

6.1 Simulation

In this paper, a computer simulation is performed according to the context of robotic soccer task.

All procedures for estimating positions and orientations of multiple robots is carried out on the SGI O2 MIPS R5000 180 Mhz. In simulation, an allowable error $\Delta E = 7^\circ$.

Table 1 shows performance of our method in the case that all robots can observe all of other robots in the environment. In this table, *Robots*, *Triangles*, *Resultant triangles1*, *Resultant triangles2*, *Real triangles* and *Computational time* shows the number of robots, the number of possible triangles, the number of possible triangles after **Step 3**, the number of possible triangles after **Step 4**, and the number of estimated triangles after **Step 5**, the actual number of triangles and the processing time for reconstructing absolute configuration, respectively. The computational time is faster than Kato et. al's method [3].

Table 1. Performance of our method (without obstacles)

Robots	Triangles	Resultant triangles 1	Resultant triangles 2	Real triangles	Computational time
4	7	5	4	${}_4C_3 = 4$	0.65ms
5	105	15	10	${}_5C_3 = 10$	10ms
6	955	47	20	${}_6C_3 = 20$	94ms
7	5660	134	35	${}_7C_3 = 35$	1012ms
8	23515	220	56	${}_8C_3 = 56$	3266ms

Fig.6 shows a simulation result for reconstructing absolute configuration between four robots. **Fig.6(a)** and (b) shows the input ODIs of four robots (A,B,C and D)and the reconstructed absolute configuration between them, respectively. In this experiment, the average error of positions and orientations is $(\bar{\Delta}X, \bar{\Delta}Y, \bar{\Delta}\theta) = (2.68(mm), 4.22(mm), 1.27(^{\circ}))$, where $(\bar{\Delta}X, \bar{\Delta}Y)$ and $\bar{\Delta}\theta$ shows the average error of estimated positions and orientation with respect to four robots.

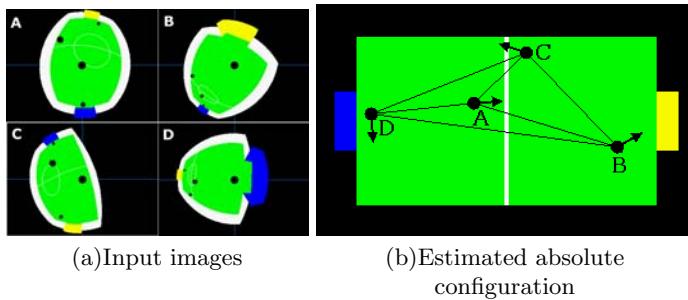


Fig. 6. Experimental result for the environment without obstacles

Fig.7(a) and (b) shows the actual and the reconstructed absolute configuration between four robots in the environment with an obstacle, respectively. The computational time is 0.65ms. In this experiment, the average error of positions and orientations is $(\bar{\Delta}X, \bar{\Delta}Y, \bar{\Delta}\theta) = (2.72(mm), 4.42(mm), 1.38(^{\circ}))$. As shown in **Fig.7**, our method correctly reconstructs triangles even if there is an obstacle. If there is an obstacle between two robots, our method correctly disappear the sight line between the two robots.

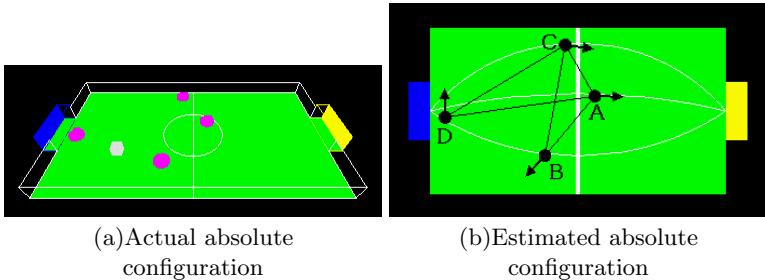


Fig. 7. Experimental result for the environment with an obstacle

6.2 Results for real robots

Fig. 8 shows examples of real ODIs and segmented images based on a simple color-based segmentation algorithm. Currently, to reduce the computational cost of image processing, the size of captured image is 120×90 pixels. As shown in **Fig. 8**, our color segmentation algorithm succeeds in extracting a red ball, blue and yellow goals, green field (ground), and purple markers on the teammates. As a result of this image processing, each robot gets input information to our algorithm such as the result of self-localization and the azimuth directions to the other robots. **Fig. 9** shows an experimental environment and actual configuration between 4 real mobile robots. **Fig. 10** shows the actual and the estimated absolute configuration between four robots in the real environment, respectively. Actual and estimated configuration corresponds to a set of triangles whose vertices are connected by dotted and solid lines, respectively. The computational time is 0.65ms. As shown in **Fig. 10**, our method correctly reconstructs triangles in real environment without obstacles.

7 Discussion

This paper proposed a new method for estimating spatial configuration between multiple robots in the environment using omnidirectional vision sensors. Even if there was an obstacle in the environment where the multiple robots were located, our method could estimate absolute configuration between robots in the environment with high accuracy. In this paper, in order to enhance functions of autonomous decentralized system, our method gives a self-localization capability to a single robot among multiple robots, in addition to capability of estimating a relative configuration from the sharing observation. Due to self-localization capability, our method can use one more constraint which is called "enumeration constraint." This constraint drastically eliminates impossible triangles and makes our algorithm fast and robust. Finally, the self-localization capability enables our method to reconstruct absolute configuration between multiple robots in the environment. The result of simulation shows that our method performs well even under noisy and ill-formed conditions of observation.

In this paper, we just verified that our method works in the environment with one obstacle. If there are plural obstacles, some robots frequently can't observe other robots. In such situation, it might be unable to reconstruct the relative configuration between robots. Even if there is no obstacle in the field, ill-formed conditions happen due to the limitation of the image processing where the marker of the other robot is occluded by the other marker. So, we should clarify the limit of our method in the environment with plural obstacles and ill-formed conditions of observation.

Now, we are going to perform real robot experiment of reconstructing absolute configuration between real robots in the environment with obstacles. We will show the results of such experiments. As future work, we will investigate a method for estimating the positions of obstacles based on the reconstructed absolute configuration.

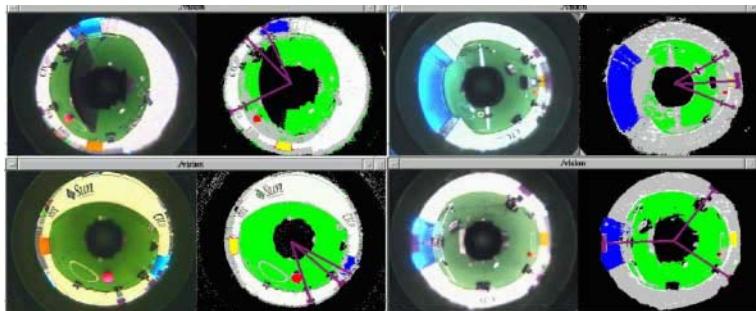


Fig. 8. Examples of proccesed ODIs



Fig.9. Experimental environment

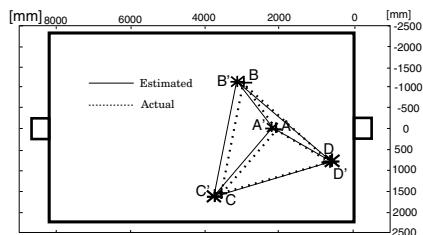


Fig.10. Estimated absolute configuration

References

1. Borenstein, J., Everett, B. and Feng, L: "Navigating Mobile Robots: Systems and Techniques". *A. K. Peters, Ltd., Wellesley, MA*, 1996.
2. Ishiguro, H: "Development of low-cost compact omnidirectional vision sensors and their applications". *Proc. of Int. Conf. Information systems, analysis and synthesis*, pp.433-439, 1998.
3. Kato, K, Ishiguro, H. and Barth, M.: "Identifying and Localizing Robots in a Multi-Robot System Environment". *Proc. of IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems*, pp.966-971, 1999.
4. Yamazawa, K, Yagi, Y. and Yachida, M: "Obstacle Detection with Omnidirectional Image Sensor HyperOmni Vision". *Proc. of Int. Conf. on Robotics and Automation*, pp.1062-1067, 1995.

Framework of Distributed Simulation System for Multi-agent Environment

NODA, Itsuki¹

noda@etl.go.jp

Electrotechnical Laboratory, Tsukuba 305, Japan

Abstract. Simulation systems are important tools for researches on Multi-agent systems. As a research tool, a simulation system should have features of: (1) light-weight and small requirement of computational resources, and (2) scalability to add new functions and features. I investigate experience of development of Soccer Server, the official soccer simulator used in RoboCup Simulation League, and propose a new framework for distributed simulation system for general purpose of multi-agent researches.

In the new framework, a simulation is divided into several modules and executed in parallel. These executions are combined by a kernel module via a computer network. Because of the modularity over networks, users easily maintain the development of simulation system.

I also discuss about the relation to HLA, another framework for distributed military simulation system.

1 Introduction

“Simulation” is an important research tool on multi-agent systems [2]. Most of multi-agent systems, in which each agent behaves intelligently and adaptively, are complex systems. Generally, it is hard to apply simple mathematical analysis to such complex systems, so that computer simulation is an indispensable method to investigate behaviors of multi-agent systems.

I developed Soccer Server[6], a soccer simulation system, as a tool for researches on multi-agent systems. It has been used as an official platform in RoboCup Simulation League. The reasons why Soccer Server is chosen are open system, light weight, and widely supported platforms. These features enable many researchers to use it as a standard tool for their research. And now, we have a large community of simulation league, in which we discuss new rules, share ideas and information, and cooperate with each other to develop libraries and documents. In the same time, it becomes clear that Soccer Server has many design problems. It was originally developed just as a prototype of the simulator, and modified again and again to add new features according to requirements from various viewpoints of researches. Therefore, the system became complicated and difficult to maintain.

Based on these experience, I investigate requirement to a platform for simulation of multi-agent systems, and propose a new framework for it, which will provide:

- capability to execute the simulation in distributed way,
- facility to add new functions easily to the simulation.

In the following sections, I investigate features and problems of the current Soccer Server, and describe the proposed framework.

2 Soccer Server and its Problems

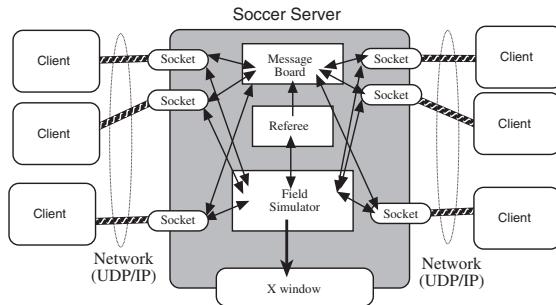


Fig. 1. Architecture of Current Soccer Server

2.1 Soccer Server

Soccer Server [6, 5] enables a soccer match to be played between two teams of player-programs (possibly implemented in different programming systems). The match is controlled using a form of client-server communication. The server (Soccer Server) provides a virtual soccer field and simulates the movements of players and a ball. A client (player program) can provide the ‘brain’ of a player by connecting to the server via a computer network and specifying actions for that player to carry out. In return, the client receives information from the player’s sensors.

A client controls only a player. It receives visual and verbal sensor information (‘see’ and ‘hear’ respectively) from the server and sends control commands (‘turn’, ‘dash’, ‘kick’ and ‘say’) to the server. Sensor information tells only partial situation of the field from the player’s viewpoint, so that the player program should make decisions using these partial and incomplete information. Limited verbal communication is also available, by which the player can communicate with each other to decide team strategy.

2.2 Features of Soccer Server

Soccer Server has been used as an official platform for RoboCup Simulation League. In addition to it, it is used widely for research, education and entertainment. Here, I like list up features of Soccer Server that are reason why it is used widely.

- Soccer Server is light. It can run entry-level PCs and requires small resources. This enables researchers to start their research from small environment. And also, in order to use it for educational purpose, it is necessary to run on PCs students can use in computer labs in schools.
- Soccer Server runs on various platforms. Finally, it supports SunOS 4, Solaris 2.x, Linux, IRIX, OSF/1, and Windows¹. It also requires quite common tools and libraries like Gnu or ANSI C++ compiler, standard C++ libraries, and X window. They are distributed freely and used widely.
- Soccer Server uses ascii string on UDP/IP for protocol between clients and the server. It enables researchers/students to use any kind of program language. Actually, participants in past RoboCup competitions used C, C++, Java, Lisp, Prolog and various research oriented AI programming systems like SOAR [10]. Version control of protocol is also an important feature. It enables us to use old clients to run in newer servers.
- The system has a separated module, `soccermonitor`, for displaying the field status on window systems. Simulation kernel, `soccerserver`, permits to connect additional monitors. While this mechanism was introduced only for displaying field window on multiple monitors, it leads unexpected activities in the different research field. Many researchers have made and have been trying to build 3D monitors to demonstrate scene of matches dynamically [8]. In addition to it, a couple of groups are building commentary systems that describe situations of matches in natural language dynamically [11, 1]. Both kinds of systems are connected with the server as secondary monitors, get information of state of matches, analyze the situations, and generate appropriate scenes and sentences.

2.3 Open Issues of Soccer Server

During past three years, Soccer Server was modified again and again in order to add new functions and to fix bugs. From these experience, it became to be clear that Soccer Server have the following open issues.

- huge communication:
Soccer Server communicates with various clients (player clients, monitor

¹ Windows versions were contributed by Sébastien Doncker and Dominique Duhaud (compatible to version 2), and now by Mario Pac (compatible to version 4) independently. Information about Mario's versions is available from:

<http://users.informatik.fh-hamburg.de/~pac/m/>

clients, offline-/online-coach clients) directly, so that the server often becomes a bottle-neck of network-traffic. Especially, communication with 22 player clients is a big issue. If a term use many communication, it causes network collisions and slow down of the server, so that another team may be influenced. This problem will be solved by distributed processing over the network.

- maintenance problem:

Though Soccer Server was maintained by a small number of developers, the source code became so complicated that it is difficult to figure out bugs and to maintain the code. The reason is that structure of classes of C++ program became not to reflect a hierarchy of required functions. Therefore, it is the time to re-design modules of the server according to required functions.

- version control:

In order to keep upper compatibility as much as possible, Soccer Server uses version control of protocol between clients and the server. Because the current server is a single module, the server must include all version of protocols. In order to solve the problem, the server should have a mechanism that enable to connect with a kind of filter or proxy that convert internal representation and each version of the protocol.

A hint to overcome these problems is “modular structure over network”. In Soccer Server, the monitor module is separated from the simulation module. As mentioned before, this modularity brings the following merits:

- This evolved many activities to develop systems to show plays in 3D, to describe game status in natural language, and to analyze performance of teams from various point of view. This is possible because the modules are connected via networks. Therefore, each developer can develop these systems independently.
- This makes researchers to develop such monitors on various platforms. This is possible because communication between modules use open and standard protocol (character strings via UDP/IP).

I apply the similar technique to other part of the simulator. In the next section, I describe the general framework, called FUSS, for distributed simulation based on the idea, and show the implementation of the soccer simulator as an example.

3 FUSS: An Universal Simulation System for Multi-Agent System

3.1 Overview

FUSS (Framework for Universal Simulation System) is a collection of programs and libraries to develop distributed simulation systems. It consists of the following items:

- **fskernel**: A kernel for a simulation system. This provides services of:

- module database
- shared memory management
- synchronization control
- FUSS library (libfuss): A library to develop modules of the simulation system. The library consists of:
 - **FsModule** library: provides a framework of a simulation module.
 - **ShrdMem** library: provides an interface to access shared memories.
 - **PhaseRef** library: provides a facility to control synchronization of simulation.
- utility library and programs: A collection of utilities.

Generally, a simulation system on FUSS consists of a kernel (**fskernel**) and a couple of simulation modules. The modules are combined into a system by the kernel via the FUSS library. Fig. 2 shows a brief structure of an example of a simulation system built on FUSS.

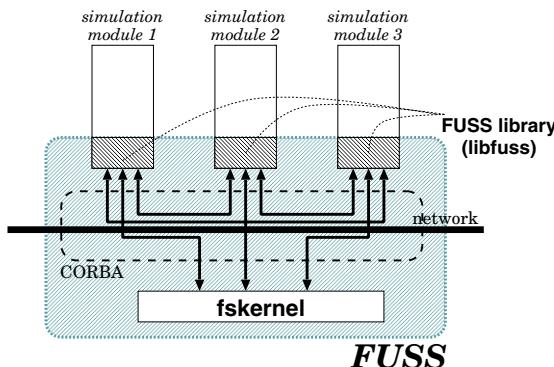


Fig. 2. A Simulation System Built on FUSS

In order to guarantee open-ness in communication among modules and the kernel, FUSS uses CORBA for the communication. This makes users free from selection of platform and programming languages to develop simulation modules. While the current implementation of FUSS uses C++, we can develop libraries on other languages that have CORBA interface.

In addition to it, FUSS uses POSIX multi-thread facility (pthread) to realize flexible interactions between modules and **fskernel**. Using this facility, the user need not care to manage control of execution of simulation and the communication.

Currently, FUSS is implemented by using C++ and OmniORB2[7]. It is available from the FUSS homepage:

<http://ci.etl.go.jp/~noda/soccer/fuss/>

3.2 Shared Memory Management

A shared memory is defined as a sub-class of `ShrdMem` class in each module. The memory is registered to the kernel before the simulation. Then, a module becomes an owner of the memory, who has an original data, and other module have its copy. The ownership can be transferred by calling `takeOwnership` method explicitly.

Modules need not know who is owner of the shared memory. A module calls `download` method before using the shared memory, and calls `upload` method after modifying the memory. Then the FUSS library maintain the consistency of the memory.

Each shared memory must be defined by IDL of CORBA. The definitions are converted into C++ classes and included by all related modules. Because FUSS uses CORBA, we can develop modules or utilities by other CORBA-compliant programming systems.

3.3 Synchronization by Phase Control

In a simulation on FUSS, modules are synchronized by `fskernel` by *phases*.

A *phase* is a kind of an event that have joined modules. When a module is plugged into the simulation system, the module send `joinPhase` messages to `fskernel` to joins a couple of phases in which it executes a part of simulation. When the phase starts, the kernel notifies the beginning of the phase by sending an `achievePhase` message to all joined modules. Then the kernel waits until all joined modules finish operations of the phase. Each module must inform the end of the operation of the phase by sending an `achievePhase` message to the kernel.

The kernel can handle two types of phases, *timer phase* and *adjunct phase*.

A *timer phase* has its own interval. The kernel tries to start the phase for every interval. For example, suppose that we develop a soccer simulator, in which a field simulator module calculate objects' movements every 100ms. In this case, we will define a timer phase (named as **field simulation phase**) that has the field simulator module as a joined module. The interval of the phase is set to 100ms. Then, the field simulator receives an `achievePhase` message for every 100ms, and executes its operation and sends an `achievePhase` message back to the kernel.

An *adjunct phase* is invoked before or after another phase adjunctively. In the example of soccer simulation, suppose that referee's judgments, which is operated by a referee module, should be performed just after the field simulation. In this case, a **referee phase** will be registered as an adjunct phase after a **field simulation phase**. Then the kernel starts the **referee phase** immediately after the **field simulation phase** is achieved. For another example, a **player phase**, in which player simulators/proxies upload players' commands, will be registered as an adjunct phase before a **field simulation phase**. In this case, the kernel starts the **player phase** first, and starts the **field simulation phase** after it is achieved.

A phase may have two or more adjunct phases before/after it. To arrange them in an order explicitly, each adjunct phase has its own tightness factor. The factor is larger, the phase occurs more tightly adjoined to the mother phase. For example, a **field simulation phase** may have two adjunct phases, a **referee phase** and a **publish phase**, after it. Tightness factors of the referee and publish phases will be 100 and 50 respectively. So, the **referee phase** occurs just after the field simulation phase, and the **broadcast phase** occurs last.

Fig. 3 shows phase-control and communication between the kernel and modules in the soccer simulation example. Note that the implementation of the phase control mechanism is general and flexible, so that there is no limitation on the number of phase, the duration of the interval of timer phase, or the depth of nest of adjunct phases.

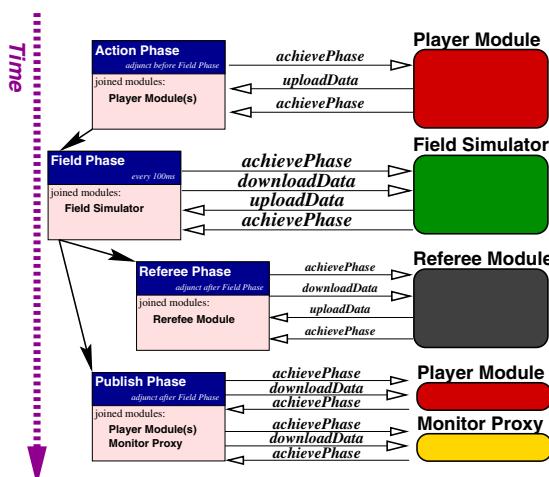


Fig. 3. Phase Control and Communication with Joined Modules

4 Implementation of Soccer Simulator on FUSS

4.1 Overview of the Design

I implemented Soccer Simulator using FUSS. In the implementation, I divided the functions of Soccer Server into the following modules:

- **Field Simulator** is a module to simulate the physical events on the field respectively.

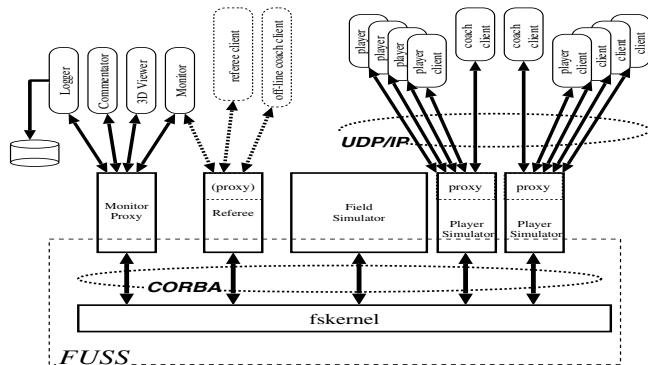


Fig. 4. Plan of Design of New Soccer Server

- **Referee Module** is a privileged module to control a match according to rules. This module may override and modify the result of field simulator.
- **Player Simulators/Proxies** are modules to simulate events inside of player’s body, and communicate with player and on-line coach clients.
- **Monitor Proxy** provides a facility of multiple monitor, commentator, and saving a log.

Execution of these modules are controlled by phases shown in Fig. 3. Note that the execution and communication with clients are processed in parallel using the multi-thread facility.

4.2 Player Simulator/Proxy

As mentioned in Sec. 2, one of major problems of the current Soccer Server is management of protocol. In the Soccer Server, the protocol is implemented in various point of the whole system. Therefore, it is difficult to maintain and version-up the protocol.

In the new design, on the other hand, A player simulator/proxy receives whole information about data from the kernel, and convert it to the suitable protocol. As a result, maintainers may focus only to this module when we change the protocol.

This style brings another merit. The current system communicates with clients directly, so it the server tends to be a bottle neck of network traffic. On the other hand, this module works as a proxy that connects with multiple clients. Therefore, when we run two proxies for both teams on two machines placed in separated sub-networks, we can distribute the traffic. This also equalizes the condition for each team even if one team uses huge communication with the server.

4.3 Referee Module

The implementation of the referee module is the key of the simulator. Compared with other modules, the referee module should have a prerogative, because the referee module needs to affect to behaviors of other modules directly. For example, the referee module restricts movements of players and a ball, which are controlled by the field simulator module, according to the rule.

We may be able to realize the prerogative by it that the referee module only controls flags that specify the restrictions, and simulator modules runs according to the flags. In this implementation, however, it is difficult to maintain the referee module separately from other modules.

In the implementation using FUSS, the referee module is invoked just before and after the simulator module and check the data. In other words, the referee module works as a ‘wrapper’ of other modules. The merit of this implementation is that it is easy to keep simulator modules independent from referee modules. Phase control described in Sec. 3.3 enables this style of implementation in a flexible manner.

5 Related Work and Discussion

Researches on distributed simulation systems are done mainly for military simulation and training [9, 12]. DIS (Distributed Interactive Simulation) [3] and HLA (High-Level Architecture) [4] have been developed for this purpose. These works focuses on connecting stand-alone simulators like flight simulators over networks. In such simulation, accuracy of simulation of behaviors of each planes and vehicles is more important than interaction between them. Therefore, DIS and HLA is designed mainly for synchronizing relatively independent simulators.

In simulation of multi-agent systems like Soccer, on the other hand, interactions among agents occur more frequently. Therefore, execution of each simulation module should be controlled more carefully. Moreover, the simulation may be regulated by complex rules that related the whole simulated world. For example, the soccer simulation have to be carried out according to rules of soccer that is related to total condition of the field. In order to implement such regulation, each simulation module needs to have a function to deal with it in HLA, in which it is difficult to maintain the regulation. Compared with this, it is easy to implement such regulation using FUSS, because FUSS has a strong ordering mechanism by phase control. In other words, FUSS has an advantage in implement tightly coupled simulation with supervised modules.

6 Summary

I investigated problems of current Soccer Server, and figured out issues that should be solved in the new simulator. Two main points are modularity and possibility of distributed simulation. Base on this investigation, I proposed a

framework for general purpose multi-agent simulation called FUSS. FUSS enables for us to develop simulation systems that run in a distributed way over computer networks.

The proposed framework is general and is not restricted to simulation of Soccer. So, it is possible to use this design as a prototype of the kernel of other simulation of complex environment like rescue from huge disasters.

Moreover, FUSS uses CORBA as a base of communication among the kernel and modules. While the current implementation provides only a library for C++, it is possible to develop libraries for other languages.

References

1. E. Andrè, G. Herzog, and T Rist. Generating multimedia presentations for RoboCup soccer games. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 200–215. Lecture Notes in Artificial Intelligence, Springer, 1998.
2. J. L. Casti. *Would-be Worlds: how simulation is changing the frontiers of science*. John Wiley and Sons, Inc., 1997a.
3. Thomas L. Clarke, editor. *Distributed Interactive Simulation Systems for Simulation and Training in the Aero Space Environment*, volume CR58 of *Critical Reviews Series*. SPIE Optical Engineering Press, April 1995.
4. Judith S. Dahmann. High level architecture for simulation: An update. In Azzeidine Bourkerche and Paul Reynolds, editors, *Distributed Interactive Simulation and Real-time Applications*, pages 32–40. IEEE Computer Society Technical Committee on Pattern Analysis and Machine Intelligence, IEEE Computer Society, July 1998.
5. Itsuki Noda and Ian Frank. Investigating the complex with virtual soccer. In Jean-Claude Heudin, editor, *VW'98 Virtual Worlds (Proc. of First International Conference)*, pages 241–253. Springer, July 1998.
6. Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12(2–3):233–250, 1998.
7. omniORB homepage. WWW home page <http://www.uk.research.att.com/omniORB/omniORB.html>.
8. A. Shinjoh and S. Yoshida. The intelligent three-dimensional viewer system for robocup. In *Proceedings of the Second International Workshop on RoboCup*, pages 37–46, July 1998.
9. Stow 97 - documentation and software. WWW home page http://web1.stricom.army.mil/STRICOM/DRSTRICOM/T3FG/SOFTWARE_LIBRARY/ST%_OW97.html.
10. Milind Tambe, W. Lewis Johnson, Randolph M. Jones, Frank Koss, John E. Laird, Paul S. Rosenbloom, and Karl Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16(1), Spring 1995.
11. Kumiko TANAKA-Ishii, Itsuki NODA, Ian FRANK, Hideyuki NAKASHIMA, Koiti HASIDA, and Hitoshi MATSUBARA. MIKE: An automatic commentary system for soccer. In Yves Demazeau, editor, *Proc. of Third International Conference on Multi-Agent Systems*, pages 285–292, July 1998.
12. Warfighters' simulation (warsim) directorate national simulation center. WWW home page <http://www-leav.army.mil/nsc/warsim/index.htm>.

Robust Real Time Color Tracking

Mark Simon, Sven Behnke, and Raúl Rojas

Free University of Berlin
Institute of Computer Science
Takustr. 9, 14195 Berlin, Germany
{simon|behnke|rojas}@inf.fu-berlin.de
<http://www.fu-fighters.de>

Abstract. This paper describes the vision system that was developed for the RoboCup F180 team FU-Fighters.

The system analyzes the video stream captured from a camera mounted above the field. It localizes the robots and the ball predicting their positions in the next video frame and processing only small windows around the predicted positions. Several mechanisms were implemented to make this tracking robust. First, the size of the search windows is adjusted dynamically. Next, the quality of the detected objects is evaluated, and further analysis is carried out until it is satisfying. The system not only tracks the position of the objects, but also adapts their colors and sizes. If tracking fails, e.g. due to occlusions, we start a global search module that localizes the lost objects again. The pixel coordinates of the objects found are mapped to a Cartesian coordinate system using a non-linear transformation that takes into account the distortions of the camera. To make tracking more robust against inhomogeneous lighting, we modeled the appearance of colors in dependence of the location using color grids. Finally, we added a module for automatic identification of our robots.

The system analyzes 30 frames per second on a standard PC, causing only light computational load in almost all situations.

1 Introduction

In the RoboCup Small-Size (F180) league, five robots on each team play soccer on a green field marked with white lines. The ball is orange and the robots, as well as the goals, are marked either yellow or blue. In addition to the yellow or blue team marker (a ping-pong ball centered on top of the robot), further markers are allowed, as long as they have different colors (refer to [6] for more details).

The robots are controlled by an external computer connected to a camera mounted above the field, such that the entire field is visible, as shown in Figure 1. The task of the vision system is to compute the positions and orientations of the robots, as well as the position of the ball. The behavior control software uses this information to operate the robots, relying on visual feedback. Since the robots and the ball move quickly and vision is usually the only input for behavior control, a fast and reliable computer vision system is essential for successful

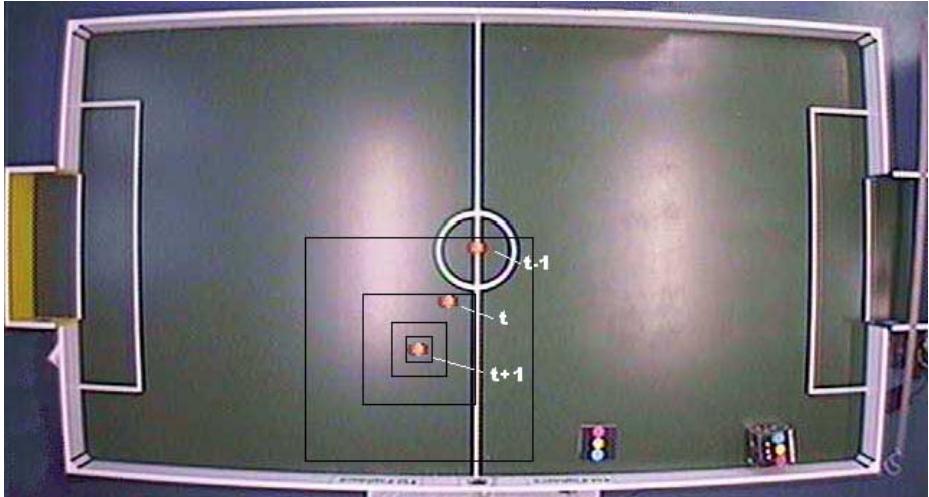


Fig. 1. A typical camera image, showing the field with shadows at the walls and reflections in the center. The linear ball prediction and a variable search window are shown too. The robots are marked with three colored dots.

play. Further information about the overall system and the hierarchical reactive control of the F180 team FU-Fighters can be found in [1] and [2].

Appropriate coloring of the interesting objects partially simplifies the vision problem, but does not make it simple. There are several problems. First, the interesting objects are not always visible. The ball can be occluded, due to the central camera mount and vertical sides of robots. Also, the robots can be occluded by people that move them manually, e.g. to place them at their kickoff positions.

Inhomogeneous lighting is also problematic. There are shadows from the robots, the walls, and the referee, as well as highlights from the spot lighting. These conditions are far from a homogeneous diffuse lighting that would give the objects a constant appearance.

The vision problem is also complicated by the undefined background next to the field and the variability of the robots' markers. All colors are allowed for the markers, as long as they are different from the reserved ones, and also the number and form of the markers changes from team to team. The vision system must be able to adapt quickly to the different markers of the opponents.

The non-linearity of the cameras wide angle optical system has also to be taken into account. The straight walls of the field appear to be convex in the captured image. One has also to correct for the height of the objects when mapping them to a 2D standard coordinate system.

The remainder of the paper is organized as follows. In the next section, we describe the robust tracking method we developed. Then, we explain the non-linear coordinate transformation used. Section 4 presents improvements that we

incorporated into the system prior the Melbourne competition: (a) the color map approach that models the appearance of colors dependent on the position; and (b) automatic identification of our robots.

2 Robust Tracking

The input for our vision system consists of an analog video stream produced by an NTSC S-Video camera mounted above the field. We capture the image using a PCI frame grabber at a resolution of 640×480 pixels in RGB format. The frame rate is 30fps. This produces an enormous data rate of 26MB/s from which we want to extract the few bytes relevant to behavior control. We need to estimate the positions of the ball and the opponent robots, as well as the positions and orientations of our robots.

The vision system analyzes only those parts of the image containing the field. The background is ignored. This reduces the data rate, but it would still not be possible to analyze the entire field in every frame without using special purpose hardware.

We therefore decided to develop a tracking system that predicts the positions of the interesting objects and analyzes only small windows around them. With a high probability, the objects will be within these small windows and we do not have to process most parts of the image.

Many other teams at RoboCup'99 relied on special hardware, like FPGAs or DSPs to process the entire image in real time [3, 4].

2.1 Ball and Robot Models

The orange ball, and the blue and yellow team markers appear as colored dots of about constant size in the image. The form of these dots is not always circular, since moving objects are captured by the camera at different positions for different half images. This causes significant differences between the odd and even image lines. The lighting and the ball shape of the objects produces highlights and shadows. However, most of the pixels belonging to an object have similar colors.

We model the ball and the team marker with its position and size, as well as its color in HSI space. In addition to the team marker, we put two colored dots on top of our robots, such that they form a line from the left to the right wheel (refer to Figure 1 for a top down view). These dots are modeled in the same way as the ball and the team markers.

The robot model contains also information about the distance of the dots and the orientation of the line. For the tracking of other team's robots we have further models with only one or two dots. We initialize the models by clicking with the mouse on the dots and we assign an ID to each of our robots.

2.2 Variable Search Windows

At any given time, most objects move slowly on the field and only some objects might move fast. This allows to predict the positions of dots from the difference of the last two positions. We use linear prediction clipped at the borders of the field (see Figure 1). If an object moves fast, this prediction is not very accurate. Therefore, we need a larger search window for this situation. However, if the object stands still, the prediction will be very good and a small search window suffices.

We developed an algorithm for dynamic adjustment of the search window sizes. If we find the dot within its window, we slowly decrease the window's side length for the next frame. If the search is not successful, we increase the size by a factor of two and search again, as illustrated for the ball in Figure 1. The size is always limited between a minimal (e.g. 16×16) and a maximal value (e.g. 128×128) and the searched region is clipped by the rectangle containing the field. During regular play, the average size of the search windows is close to the minimal value. Large windows are only necessary, when objects move very fast (e.g. the ball has been kicked) or dots cannot be segmented (e.g. due to the lighting or occlusions).

One problem when enlarging the search windows for the robot's dots is that one can find dots of the same color that belong to different robots. To prevent this, we use two heuristics. First, we remove dots of found robots from the image by “painting” them black. Second, we enlarge a search window only after all dots have been searched for in their small window.

2.3 Color Segmentation

To find a colored dot in its search window, we first determine the pixel that has the smallest RGB-distance to the color of the dot's model. We assume that this pixel belongs to the dot and try to segment the remaining pixels from the background by analyzing a quadratic window with a side length of about twice the dot's diameter that is centered at the selected pixel.

We use two methods for color segmentation. The first method that is illustrated in Figure 2 works in HSI color space [5]. Since we are looking for colorful dots, we apply a saturation mask and an intensity mask to the window. Only pixels that are saturated and neither too bright nor too dark are analyzed further. The final segmentation decision is done using the hue distance to the model's color. Working in HSI space has the advantage that the method is quite insensitive against changes in intensity. However, it can only be applied to saturated dots. If the model dot is not saturated or the HSI segmentation fails, we try to segment the dot in RGB color space. This backup method is needed, since some teams might use unsaturated markers, and also since saturated markers may appear as unsaturated dots when viewed in shadow or hot spots. Here, we use only the RGB-distance between the pixels color and the color of the model. All pixels of similar color are segmented as belonging to the dot.

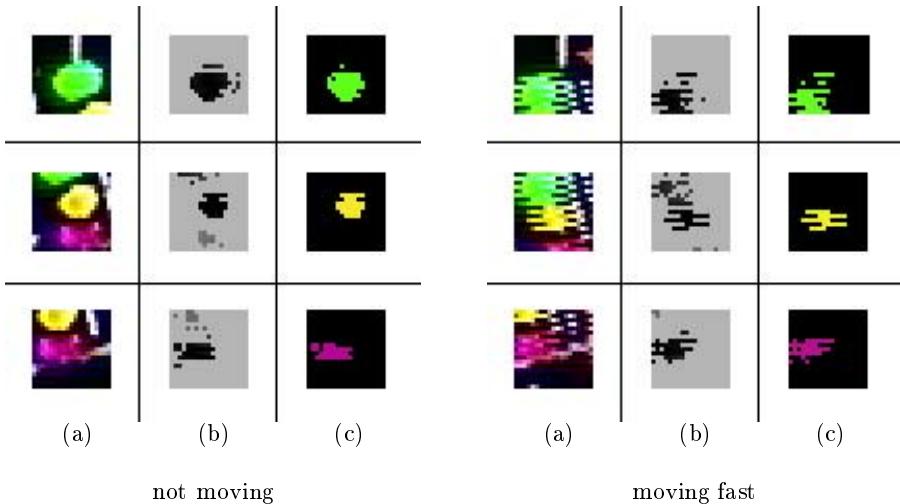


Fig. 2. Segmentation of colored dots in HSI space. Shown are (a) the original images, (b) the saturation/intensity masks, (c) and the segmented pixels, that have been selected using the hue distance to the model.

A quality measure is computed for the dot from the similarity to its model. If the size and the color are similar to the model, then the quality is high. The dot's position is updated if its quality is high enough and the dot is part of a valid robot, or if it is the ball. We estimate the position of the dot with sub-pixel resolution as the average location of the segmented pixels. If the quality is good, the size and average color of the dot are adapted slowly, as long as they do not deviate significantly from the initial model.

The segmentation does not explicitly take the shape of the dot into account. It enforces compact dots by searching many pixels in a small quadratic window.

2.4 Robot Search

To find a robot, all dots that belong to its model are searched for. If all dots can be localized, we compute the robot's quality from the degree of similarity to its model. We take into account the quality of the individual dots and their geometry, e.g. the distance between them.

For our robots, where three colinear equidistant colored markers are present, we check if the dots found fulfill this geometric constraint. If only one of the three dots is not found with sufficient quality, we can exploit the redundancy of our model. From the locations of the two dots found, we know where the third one should be and can check if it is really there.

We update the model of the robot only if the quality of the robot found is high enough. The position is adapted faster and more often than the geometry.

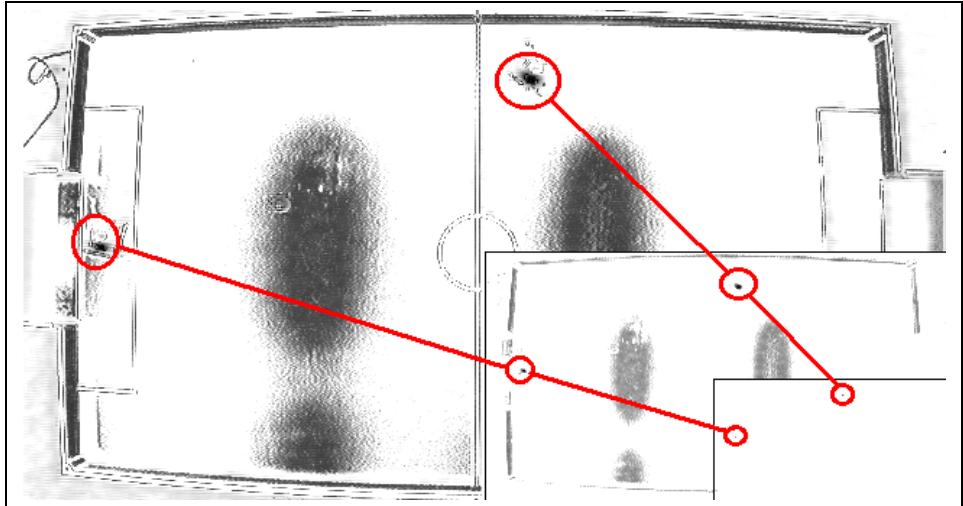


Fig. 3. Global search for a color. Shown are the differences to the desired color at three resolutions. Marked are two dots that have the most similar colors.

To prevent the tracking of non-robots, we count how frequently the quality of the robot found is low. If the robot is found for a certain time with low quality, then it is considered lost and a global search is performed to find it again.

If we lose objects, we assume that they are still at the position where we saw them last and signal this to the behavior control. If we lose the ball next to a robot that is near to the goal line, we assume that this robot occludes the ball and update its position together with the robot. This feature proved to be useful in penalty situations.

2.5 Global Search

The global search method is needed if robots are lost, e.g. due to occlusions. It is not needed for the ball since we enlarge its dynamic search window until it covers the whole field.

Global search analyzes the entire field after the found objects have been removed from the image. We search for the colors of the dots contained in the models of the lost robots and try to combine them to valid robots.

To reduce pixel noise, we spatially aggregate the color information as follows. First, we compute the distance of all pixels to the desired colors in RGB color space. Next, we subsample these distance maps twice, taking each time the average of four pixels, as shown in Figure 3. Now, we find the best positions for each color, taking into account minimal dot distances. Using the lists of dots found, we search for combinations of dots that fulfill the geometrical constraints of the robot models. This search starts with the dots that have the highest quality. If more than one robot from a team is lost then we assign the found robots to the models that have the closest positions.

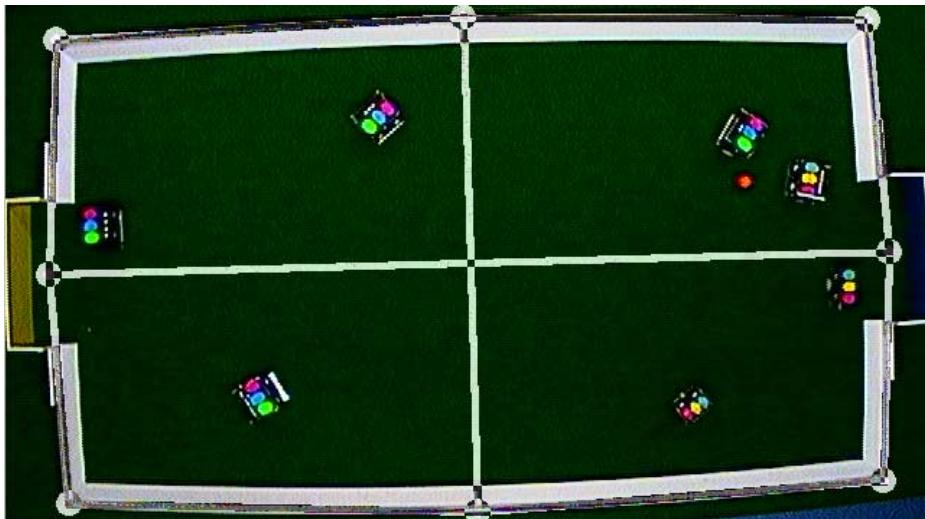


Fig. 4. Eight points parameterize the non-linear coordinate transformation.

We avoid calling the costly global search routine every frame, since this would reduce the frame rate and would therefore increase the risk of loosing further objects. It would also add overhead to the reaction time of the overall system. Fortunately, the global search is needed mostly when the game is interrupted, e.g. for a kickoff. Then, people moving robots by hand cause occlusions that trigger the global search. To avoid this interference, the FU-Fighters robots position themselves for kick off.

During regular play, the processor load caused by the 30Hz computer vision is as low as 30% on a Pentium-II-300, leaving enough time for behavior control and communication that run on the same machine.

3 Coordinate Transformation

All tracking is done in pixel coordinates, but the behavior control needs the positions of the objects in a standard coordinate system. The origin of that system is located in the middle of the field and its length corresponds to the interval $[-1, 1]$. The width of the standard system is chosen such that the aspect ratio of the field is preserved.

A linear coordinate transformation is not sufficient, since the camera produces distortions. We model them for the four quadrants of the field using eight positions located at the fields corners and in the middle of the walls (see Figure 4). The central point is computed from the intersection of the two lines connecting the wall centers that become the axis of the standard system.

In each quadrant, we perform a bilinear interpolation between its corners. We also account for the different heights of the objects by multiplying the trans-

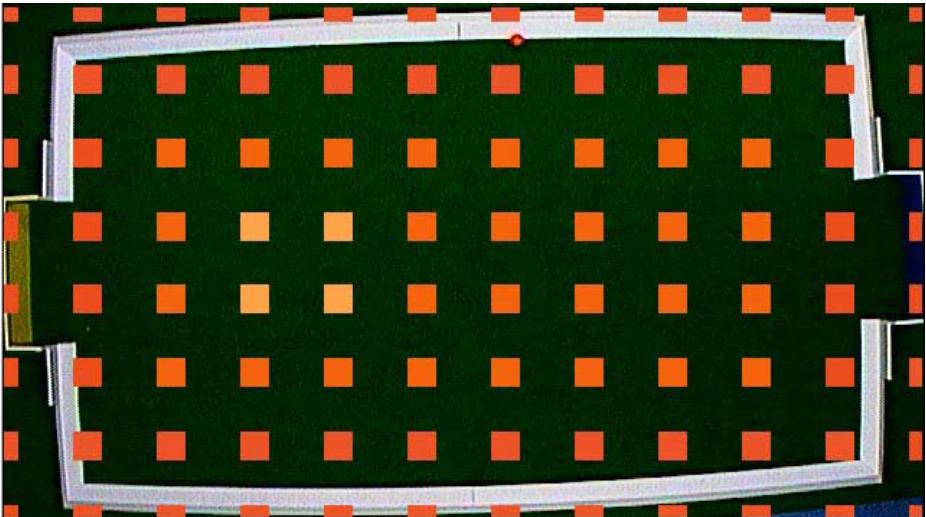


Fig. 5. Color map for the orange ball with a resolution of 12×8 . The color is darker near the walls and brighter in the center. A hot spot is visible in the middle of the left half.

formed coordinates with a suitable factor. The described transformation models the non-linearities of the camera sufficiently.

4 Improvements

4.1 Color Maps

When adapting the color of individual dots, one assumes that the appearance of a color only depends on the dot itself. This is not correct, since its appearance also depends on the possibly inhomogeneous lighting. Therefore, modeling the inhomogeneities should improve the tracking system.

The idea is to maintain for each color a map that models its appearance in dependence of the location on the field. We assume that the lighting changes slowly with position and use a low-resolution grid (e.g. 12×8).

We initialize these maps uniformly and edit it at extreme positions, e.g. by clicking at a colored marker when it is in a hot spot or in a shadowed region. A map for the orange ball might look like the one shown in Figure 5 after initialization.

When the tracking system is running, we adapt the maps automatically every time objects have been found. This approach can cope with slow global changes of the lighting as well as local inhomogeneities that vary slowly in space and time.



Fig. 6. Robot identification. Three different binary codes are shown, represented by the black and white markers located next to the colored markers.

4.2 Robot Identification

When localizing our robots they have to be mapped to the behavior processes that control the individual robots. Our first approach was to use identical robot markers and to maintain the mapping by initializing it manually and tracking the robots all the time. This works fine, as long as tracking loses less than one robot at a time. However, when robots are occluded or leave the field, the danger of exchanging two of them is high.

Such permutations cannot be corrected by the vision system described so far. Both robots get then the wrong visual feedback, which does not lead to useful behavior. To deal with this problem, we developed an automatic identification module.

Since the number of different colors is limited, we decided to use additional black and white markers, arranged in a binary pattern. Three markers are placed next to the three colored dots, as can be seen in Figure 6. To avoid uniform markers, we use only the six words that contain at least one black and at least one white marker. The resulting code is similar to the one described in [4].

Identification is done only after successful localization. We compute the positions of the markers from the positions of the colored dots and estimate their intensity using a Gaussian filter that has approximately the size of the markers.

Then, we order the three filter answers. Since both black and white markers are present, the brightest response must be white and the darkest one must be black. We only have to decide if the remaining marker is black or white. This can be done by comparing it to the average of the extreme responses.

Due to interlace problems, we do identification only if robots don't move too fast. We also check the investigated regions for homogeneity, to make sure that we are looking at the inside of a marker. Identification works reliably even when lighting changes. It significantly reduces the time needed for manual initialization of the system.

5 Summary

This paper described the tracking of robots and the ball for our RoboCup Small-Size team. We implemented several mechanisms that make the system robust and

fast, such as dynamic search windows, global search, color maps and robot identification. One important aspect was the careful design of the robot's markers, such that no combination of the markers from two different robots can be seen as a robot and such that there is some redundancy, for the case that one marker cannot be localized.

The FU-Fighters used the described tracking system in Amsterdam, where we won the European Championship 2000 and the improved system during the RoboCup'2000 competition in Melbourne, where we finished second, next to Big Red from Cornell University. The system delivered reliable information even in situations where other teams failed to localize the ball or the robots, e.g. when the ball was inside the wall's shadow or the referee was projecting a shadow on the field.

In the future, we plan to use a digital camera that can be connected to the PC via an IEEE-1394 link. This camera produces an uncompressed YUV 4:2:2 image with 640×480 pixels at a rate of 30Hz. The digital link, as well as the progressive scan should increase the image quality and therefore simplify interpretation.

We are also investigating the possibility of adding an omni-directional camera to the robots for local vision. The image will initially be transferred via a wireless analog video link to an external PC, where it can be analyzed. We plan to apply the tracking principle to edges, e.g. the ones between the green field and the white walls to localize the robots. If this can be done with a low computational load, it would be possible to implement the local computer vision on a small, low power on-board computer, such as a PDA.

References

1. Peter Ackers, Sven Behnke, Bernhard Frötschl, Wolf Lindstrot, Manuel de Melo, Raul Rojas, Andreas Schebesch, Mark Simon, Martin Sprengel, and Oliver Tenhio. The soul of a new machine. Technical Report B-12/99, Freie Universität Berlin, 1999.
2. Sven Behnke, Bernhard Frötschl, Raul Rojas, Peter Ackers, Wolf Lindstrot, Manuel de Melo, Mark Preier, Andreas Schebesch, Mark Simon, Martin Sprengel, and Oliver Tenhio. Using hierarchical dynamical systems to control reactive behaviors. In *Proceedings IJCAI'99 - International Joint Conference on Artificial Intelligence, The Third International Workshop on RoboCup - Stockholm*, pages 28–33, 1999.
3. S. Cordadeschi, editor. *RoboCup'99 Small-Size Team Descriptions*. <http://www.ida.liu.se/ext/robocup/small/intro>.
4. Paulo Costa, Paulo Marques, Antonio Moreira, Armando Sousa, and Pedro Costa. Tracking and identifying in real time the robots of a F-180 team. In *Proceedings IJCAI'99 - International Joint Conference on Artificial Intelligence, The Third International Workshop on RoboCup - Stockholm*, 1999.
5. A.K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.
6. The RoboCup Federation. *RoboCup Regulations & Rules*. <http://www.robocup.org>.

Reinforcement Learning for 3 vs. 2 Keepaway

Peter Stone, Richard S. Sutton, and Satinder Singh

AT&T Labs — Research

180 Park Ave.

Florham Park, NJ 07932

{pst^on, s^ott^on, b^oveja}@research.att.com

<http://www.research.att.com/{~pst^on, ~s^ott^on, ~b^oveja}>

Abstract. As a sequential decision problem, robotic soccer can benefit from research in reinforcement learning. We introduce the 3 vs. 2 keepaway domain, a subproblem of robotic soccer implemented in the RoboCup soccer server. We then explore reinforcement learning methods for policy evaluation and action selection in this distributed, real-time, partially observable, noisy domain. We present empirical results demonstrating that a learned policy can dramatically outperform hand-coded policies.

1 Introduction

Robotic soccer is a sequential decision problem: agents attempt to achieve a goal by executing a sequence of actions over time. Every time an action can be executed, there are many possible actions from which to choose.

Reinforcement learning (RL) is a useful framework for dealing with sequential decision problems [11]. RL has several strengths, including: it deals with stochasticity naturally; it deals with delayed reward; architectures for dealing with large state spaces and many actions have been explored; there is an established theoretical understanding (in limited cases); and it can deal with random delays between actions.

These are all characteristics that are necessary for learning in the robotic soccer domain. However, this domain also presents several challenges for RL, including: distribution of the team's actions among several teammates; real-time action; lots of hidden state; and noisy state.

In this paper, we apply RL to a subproblem of robotic soccer, namely 3 vs. 2 keepaway. The remainder of the paper is organized as follows. Sections 2 and 3 introduce the 3 vs. 2 keepaway problem and map it to the RL framework. Section 4 introduces tile coding or CMACs, the function approximator we use with RL in all of our experiments. Sections 5 and 6 describe the experimental scenarios we have explored. Section 7 describes related work and Section 8 concludes.

2 3 vs. 2 Keepaway

All of the experiments reported in this paper were conducted in the RoboCup soccer server [6], which has been used as the basis for successful international

competitions and research challenges [4]. As presented in detail in [8], it is a fully *distributed, multiagent* domain with both *teammates* and *adversaries*. There is *hidden state*, meaning that each agent has only a partial world view at any given moment. The agents also have *noisy sensors and actuators*, meaning that they do not perceive the world exactly as it is, nor can they affect the world exactly as intended. In addition, the perception and action cycles are *asynchronous*, prohibiting the traditional AI paradigm of using perceptual input to trigger actions. *Communication* opportunities are limited; and the agents must make their decisions in *real-time*. These italicized domain characteristics combine to make simulated robotic soccer a realistic and challenging domain.

The players in our experiments are built using CMUnited-99 agent skills [9] as a basis. In particular, their skills include the following:

HoldBall(): Remain stationary while keeping possession of the ball in a position that is as far away from the opponents as possible.

PassBall(f): Kick the ball directly towards forward f .

GoToBall(): Intercept a moving ball or move directly towards a stationary ball.

GetOpen(): Move to a position that is free from opponents and open for a pass from the ball's current position (using SPAR [15]).

We consider a “keepaway” task in which one team (forwards) is trying to maintain possession of the ball within a region, while another team (defenders) is trying to gain possession. Whenever the defenders gain possession or the ball leaves the region, the *episode* ends and the players are reset for another episode (with the forwards being given possession of the ball again).

For the purposes of this paper, the region is a 20m x 20m square, and there are always 3 forwards and 2 defenders (see Figure 1). Both defenders use the identical fixed strategy of always calling GoToBall(). That is, they keep trying to intercept the ball (as opposed to marking a player). Upon reaching the ball, they try to maintain possession using HoldBall().

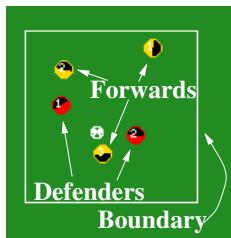


Fig. 1. A screen shot from a 3 vs. 2 keepaway episode.

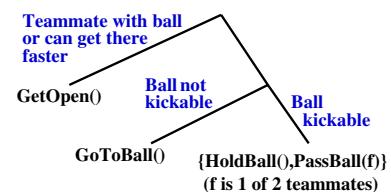


Fig. 2. The forwards' policy space.

An omniscient coach agent manages the play, ending episodes when a defender gains possession of the ball for a set period of time or when the ball goes

outside of the region. At the beginning of each episode, it resets the location of the ball and the players semi-randomly within the region of play. The defenders both start in one corner and each forward is placed randomly in one of the other three corners – one forward per corner.

3 vs. 2 keepaway is a subproblem of robotic soccer. The principal simplifications are that there are fewer players involved; they are playing in a smaller area; and the players are always focussed on the same high-level goal: they don't need to balance offensive and defensive considerations. Nevertheless, the skills needed to play 3 vs. 2 keepaway well would clearly also be very useful in the full problem of robotic soccer.

3 Mapping Keepaway onto Reinforcement Learning

Our keepaway problem maps fairly directly onto the discrete-time, episodic RL framework. The RoboCup Soccer server operates in discrete time steps, $t = 0, 1, 2, \dots$, each representing 100 ms of simulated time. When one episode ends (e.g., the ball is lost to the defenders), another begins, giving rise to a series of episodes. RL views each episode as a sequence of states, actions, and rewards:

$$s_0, a_0, r_1, s_1, \dots, s_t, a_t, r_{t+1}, s_{t+1}, \dots, r_T, s_T$$

where s_t denotes the state of the simulator at step t , a_t denotes the action taken based on some, presumably incomplete, perception of s_t , and $r_{t+1} \in \mathbb{R}$ and s_{t+1} represent the resultant reward and the next state. If we wish to refer to an event within a particular episode, we add an additional index for the episode number. Thus the state at time t in the m th episode is denoted $s_{t,m}$, the corresponding action is $a_{t,m}$, and the length of the m th episode is T_m . The final state of any episode, s_{T_m} is the state where the defenders have possession or the ball has gone out of bounds. In this case we consider, all rewards are zero except for the last one, which is minus one, i.e., $r_{T_m} = -1, \forall m$. We use a standard temporal discounting technique which causes the RL algorithms to try to postpone this final negative reward as long as possible, or ideally avoid it all together.¹

A key consideration is what one takes as the action space from which the RL algorithm chooses. In principle, one could take the full joint decision space of all players on a team as the space of possible actions. To simplify, we consider only top-level decisions by the forward in possession of the ball, that is, $a_t \in \{\text{HoldBall}(), \text{PassBall}(f), \text{GoToBall}(), \text{GetOpen}()\}$.

That is, we considered only policies within the space shown in Figure 2. When a teammate has possession of the ball,² or a teammate could get to the

¹ We also achieved successful results in the undiscounted case in which $r_t = 1, \forall t$.

Again, the RL algorithm aims to maximize total reward by extending the episode as long as possible.

² "Possession" in the soccer server is not well-defined since the ball never occupies the same location as a player. One of our agents considers that it has possession of the ball if the ball is close enough to kick it.

ball more quickly than the forward could get there itself, the forward executes `GetOpen()`. Otherwise, if the forward is not yet in possession of the ball, it executes `GoToBall()`; if in possession, it executes either `HoldBall()` or `PassBall(f)`. In the last case, it also selects to which of its two teammates to bind f . Within this space, policies can vary only in the behavior of the forward in possession of the ball. For example, we used the following three policies as benchmarks:

Random Execute `HoldBall()` or `PassBall(f)` randomly

Hold Always execute `HoldBall()`

Hand-coded

If no defender is within 10m: execute `HoldBall()`

Else If receiver f is in a better location than the forward with the ball and the other teammate; and the pass is likely to succeed (using the CMUnited-99 pass-evaluation function, which is trained off-line using the C4.5 decision tree training algorithm [7]): execute `PassBall(f)`

Else execute `HoldBall()`

4 Function Approximation

In large state spaces, RL relies on function approximation to generalize across the state space. To apply RL to the keepaway task, we used a sparse, coarse-coded kind of function approximation known as tile coding or CMAC [1, 16]. This approach uses multiple overlapping tilings of the state space to produce a feature representation for a final linear mapping where all the learning takes place (see Figure 3). Each tile corresponds to a binary feature that is either 1 (the state is within this tile) or zero (the state is not within the tile). The overall effect is much like a network with fixed radial basis functions, except that it is particularly efficient computationally (in other respects one would expect RBF networks and similar methods (see [12]) to work just as well).

It is important to note that the tilings need not be simple grids. For example, to avoid the “curse of dimensionality,” a common trick is to ignore some dimensions in some tilings, i.e., to use hyperplanar slices instead of boxes. A second major trick is “hashing”—a consistent random collapsing of a large set of tiles into a much smaller set. Through hashing, memory requirements are often reduced by large factors with little loss of performance. This is possible because high resolution is needed in only a small fraction of the state space. Hashing frees us from the curse of dimensionality in the sense that memory requirements need not be exponential in the number of dimensions, but need merely match the real demands of the task. We used both of these tricks in applying tile coding to 3 vs. 2 keepaway.

5 Policy Evaluation

The first experimental scenario we consider is policy evaluation from an omniscient perspective. That is, the players were given fixed, pre-determined policies

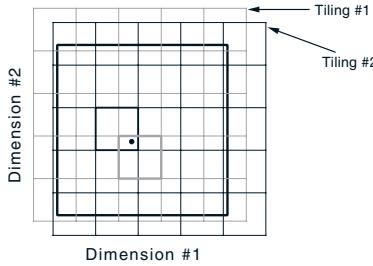


Fig. 3. Our binary feature representation is formed from multiple overlapping tilings of the state variables. Here we show two 5×5 regular tilings offset and overlaid over a continuous, two-dimensional state space. Any state, such as that shown by the dot, is in exactly one tile of each tiling. A state's tiles are used to represent it in the RL algorithm. The tilings need not be regular grids such as shown here. In particular, they are often hyperplanar slices, the number of which grows sub-exponentially with dimensionality of the space. Tile coding has been widely used in conjunction with reinforcement learning systems (e.g., [16]).

(not learned on-line) and an omniscient agent attempts to estimate the value of any given state from the forwards' perspective. Let us denote the overall policy of all the agents as π . The value of a state s given policy π is defined as

$$V^\pi(s) = E\left\{\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, \pi\right\} = E\{-\gamma^{T-1} \mid s_0 = s, \pi\}$$

where $0 \leq \gamma < 1$ is a fixed *discount-rate* parameter (in our case $\gamma = 0.9$), and T is the number of time steps in the episode starting from s .³ Thus, a state from which the forwards could keep control of the ball indefinitely would have the highest value, zero; all other states would have negative value, approaching -1 for states where the ball is lost immediately. In this section we consider an RL method for learning to approximate the value function V^π .

As is the case throughout this paper, both defenders always call `GoToBall()` until they reach the ball. The forwards use the hand-coded policy defined in Sections 2.

Every simulator cycle in which the ball is in kicking range of one of the forwards we call a *decision step*. Only on these steps does a forward actually have to make a decision about whether to `HoldBall` or `PassBall(f)`. On each decision step the omniscient coach records a set of 13 state variables, computed based on F1, the forward with the ball; F2, the other forward that is closest to the ball; F3, the remaining forward; D1, the defender that is closest to the ball; D2, the other defender; and C, the center of the playing region (see Figure 4). Let $\text{dist}(a,b)$ be the distance between a and b and $\text{ang}(a,b,c)$ be the angle between a and c with vertex at b. For example, $\text{ang}(F3,F1,D1)$ is shown in Figure 4. We used the following 13

³ In the undiscounted case, the γ parameter can be eliminated and $V^\pi(s) = E\{T \mid s_0 = s\}$.

state variables: $\text{dist}(F1, C)$, $\text{dist}(F1, F2)$, $\text{dist}(F1, F3)$, $\text{dist}(F1, D1)$, $\text{dist}(F1, D2)$, $\text{dist}(F2, C)$, $\text{dist}(F3, C)$, $\text{dist}(D1, C)$, $\text{dist}(D2, C)$, $\text{Minimum}(\text{dist}(F2, D1), \text{dist}(F2, D2))$, $\text{Minimum}(\text{dist}(F3, D1), \text{dist}(F3, D2))$, $\text{Minimum}(\text{ang}(F2, F1, D1), \text{ang}(F2, F1, D2))$, and $\text{Minimum}(\text{ang}(F3, F1, D1), \text{ang}(F3, F1, D2))$.

By running the simulator with the standard policy π through a series of episodes, we collected a sequence of 1.1 million decision steps. The first 1 million of these we used as a training set and the last 100,000 we reserved as a test set. As a post-processing step, we associated with each decision step t , in episode m , the empirical return, $-\gamma^{T_m-t-1}$. This is a sample of the value, $V^\pi(s_{t,m})$, which we are trying to learn. We used these samples as targets for a supervised learning, or *Monte Carlo*, approach to learning an approximation to V^π . We also applied temporal-difference methods, in particular, the TD(0) algorithm (see Sutton and Barto (1998) for background on both kinds of method).

It is not entirely clear how to assess the performance of our methods. One performance measure is the standard deviation of the learned value function from the empirical returns on the test set. For example, the best constant prediction (ignoring the state variables) on the test set produced a standard deviation of 0.292. Of course it is not possible to reduce the standard deviation to zero even if the value function is learned exactly. Because of the inherent noise in the domain, the standard deviation can be reduced only to some unknown base level. Our best result reduced it to 0.259, that is, by 11.4%. Our tests were limited, but we did not find a statistically significant difference between Monte Carlo and TD methods.

Another way to assess performance is to plot empirical returns vs estimated values from the test set, as in Figure 5. Due to the discrete nature of the simulation, the empirical returns can only take discrete values, corresponding to $-\gamma^k$ for various integers k . The fact that the graph skews to the right indicates a correlation between the estimated values and the sample returns in the test set.

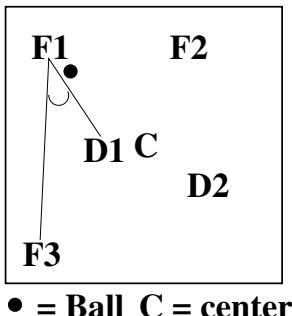


Fig. 4. The state variables input to the value function approximator.

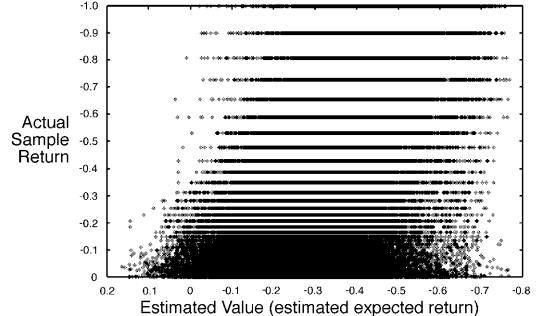


Fig. 5. Correlation of our best approximation of V^π to the sample returns in the training set.

The biggest influence on performance was the kind of function approximation used—the number and kind of tilings. We experimented with a few possibilities and obtained our best results using 14 tilings, 13 that sliced along a single state variable (one for each of the 13 state variables), and 1 that jointly considered $\text{dist}(F1, D1)$ and $\text{dist}(F1, D2)$, the distances from the forward with the ball to the two defenders.

6 Policy Learning

Next we considered reinforcement learning to adapt the forwards' policies to maximize their time of possession. To address this challenge, we used Q-learning [16], with each forward learning independently of the others. Thus, all forwards might learn to behave similarly, or they might learn different, complementary policies. Each forward maintained three function approximators, one for each of its possible actions when in possession of the ball (otherwise it was restricted to the policy space given by Figure 2). Each function approximator used the same set of 14 tilings that we found most effective in our policy evaluation experiments.

Below we outline the learning algorithm used by each forward. Here $\text{ActionValue}(a, x)$ denotes the current output of the function approximator associated with action a given that the state has features x . The function $\text{UpdateRL}(r)$ is defined below.

- Set counter = -1
- If episode over (ball out of bounds or defenders have possession)
 - If counter > 0, $\text{UpdateRL}(-1)$
- Else, If the ball is not kickable for anyone,
 - If counter ≥ 0 , increment counter
 - If in possession of the ball, $\text{GoToBall}()$
 - Else, $\text{GetOpen}()$
- Else, If ball is within kickable range
 - If counter > 0, $\text{UpdateRL}(0)$
 - Set LastAction = $\text{Max}(\text{ActionValue}(a, \text{current state variables}))$ where a ranges over possible actions, LastVariables = current state variables
 - Execute LastAction
 - Set counter = 0
- Else (the ball is kickable for another forward)
 - If counter > 0, $\text{UpdateRL}(0)$ (with action possibilities and state variables from the other forward's perspective)
 - Set counter = -1

$\text{UpdateRL}(r)$:

- $\text{TdError} = r * \gamma^{counter-1} + \gamma^{counter} \text{Max}(\text{ActionValue}(a, \text{current state variables})) - \text{ActionValue}(\text{LastAction}, \text{LastVariables})$
- Update the function approximator for LastAction and LastFeatures, with TdError

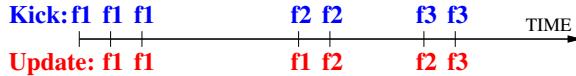


Fig. 6. A time-line indicating sample decision steps (kicks) and updates for the three forwards f_1 , f_2 , and f_3 .

Note that this algorithm maps onto the semi-Markov decision process framework: value updates are only done when the ball is kickable for *someone* as indicated in Figure 6.

To encourage exploration of the space of policies, we used *optimistic initialization*: the untrained function approximators initially output 0, which is optimistic in that all true values are negative. This tends to cause all actions to be tried, because any that are untried tend to look better (because they were optimistically initialized) than those that have been tried many times and are accurately valued. In some of our runs we further encouraged exploration by forcing a random action selection on 1% of the decision steps.

With $\gamma = 0.9$, as in Section 5, the forwards learned to always hold the ball (presumably because the benefits of passing were too far in the future). The results we describe below used $\gamma = 0.99$. To simplify the problem, we gave the agents a full 360 degrees of noise-free vision for this experiment (object movement and actions are still both noisy). We have not yet determined whether this simplification is necessary or helpful.

Our main results to date are summarized in the learning curves shown in Figure 7. The y-axis is the average time that the forwards are able to keep the ball from the defenders (average episode length); the x-axis is training time (simulated time \approx real time). Results are shown for five individual runs over a couple of days of training, varying the learning-rate and exploration parameters. The constant performance levels of the benchmark policies (separately estimated) are also shown as horizontal lines. All learning runs quickly found a much better policy than any of the benchmark policies, including the hand-coded policy. A better policy was even found in the first data point, representing the first 500 episodes of learning. Note that the variation in speed of learning with the value of the learning-rate parameter followed the classical inverted-U pattern, with fastest learning at an intermediate value. With the learning-rate parameter set to $1/8$, a very effective policy was found within about 15 hours of training. Forcing 1% random action to encourage further exploration did not improve performance in our runs. Many more runs would be needed to conclude that this was a reliable effect. However, Figure 8, showing multiple runs under identical conditions, indicates that the successful result is repeatable.

7 Related Work

Distributed reinforcement learning has been explored previously in discrete environments, such as the pursuit domain [13] and elevator control [3]. This task

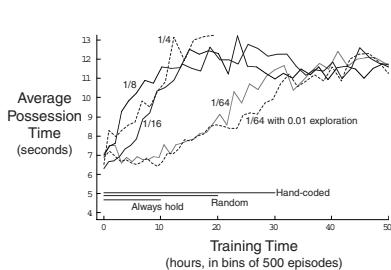


Fig. 7. Learning curves for five individual runs varying the learning-rate parameter as marked. All runs used optimistic initialization to encourage exploration. One run used 1% random actions to further encourage exploration.

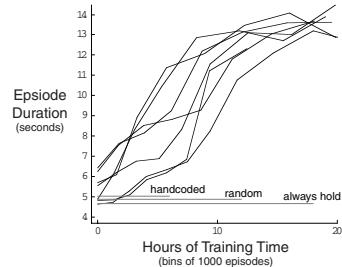


Fig. 8. Multiple successful runs under identical characteristics. In this case, undiscounted reward and 1% random exploration were used. Q-values were initialized to 0.

differs in that the domain is continuous, that it is real-time, and that there is noise both in agent actions and in state-transitions.

Reinforcement learning has also been previously applied to the robotic soccer domain. Using real robots, Uchibe used RL to learn to shoot a ball into a goal while avoiding an opponent [14]. This task differs from keepaway in that there is a well-defined goal state. Also using goal-scoring as the goal state, TPOT-RL was successfully used to allow a full team of agents to learn collaborative passing and shooting policies using a Monte Carlo learning approach, as opposed to the TD learning explored in this paper [10]. Andou's "observational reinforcement learning" was used for learning to update players' positions on the field based on where the ball has previously been located [2].

In conjunction with the research reported in this paper, we have been exploring techniques for keepaway in a full 11 vs. 11 scenario played on a full-size field [5].

8 Conclusion

In this paper, we introduced the 3 vs. 2 keepaway task as a challenging subproblem of the entire robotic soccer task. We demonstrated (in Section 5) that RL prediction methods are able to learn a value function for the 3 vs. 2 keepaway task. We then applied RL control methods to learn a significantly better policy than could easily be hand-crafted. Although still preliminary, our results suggest that reinforcement learning and robotic soccer may have much to offer each other.

Our future work will involve much further experimentation. We would like to eliminate omniscience and centralized elements of all kinds, while expanding the task to include more players in a larger area. Should the players prove capable of learning to keep the ball away from an opponent on a full field with 11 players on each team, the next big challenge will be to allow play to continue when the defenders get possession, so that the defenders become forwards and vice versa.

Then all players will be forced to balance both offensive and defensive goals as is the case in the full robotic soccer task.

References

1. J. S. Albus. *Brains, Behavior, and Robotics*. Byte Books, Peterborough, NH, 1981.
2. T. Andou. Refinement of soccer agents' positions using reinforcement learning. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 373–388. Springer Verlag, Berlin, 1998.
3. R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Processing Systems 8*, Cambridge, MA, 1996. MIT Press.
4. H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The RoboCup synthetic agent challenge 97. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 24–29, San Francisco, CA, 1997. Morgan Kaufmann.
5. D. McAllester and P. Stone. Keeping the ball from cmunited-99. In P. Stone, T. Balch, and G. Kraetschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Berlin, 2001. Springer Verlag. To appear.
6. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
7. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
8. P. Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.
9. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 35–48. Springer Verlag, Berlin, 2000.
10. P. Stone and M. Veloso. Team-partitioned, opaque-transition reinforcement learning. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999. Also in *Proceedings of the Third International Conference on Autonomous Agents*, 1999.
11. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, Massachusetts, 1998.
12. R. S. Sutton and S. D. Whitehead. Online learning with random representations. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 314–321, 1993.
13. M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.
14. E. Uchiibe. *Cooperative Behavior Acquisition by Learning and Evolution in a Multi-Agent Environment for Mobile Robots*. PhD thesis, Osaka University, January 1999.
15. M. Veloso, P. Stone, and M. Bowling. Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, Boston, September 1999.
16. C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

Fault-tolerant self localization by case-based reasoning

Jan Wendler*, Steffen Brüggert, Hans-Dieter Burkhard, and Helmut Myritz

Humboldt University Berlin, Department of Computer Science,
Artificial Intelligence Laboratory, Unter den Linden 6, D-10099 Berlin, Germany
email: wendler@bruegger.hdb.myritz@informatik.hu-berlin.de,
WWW: <http://www.ki.informatik.hu-berlin.de>

Abstract. In this article we present a case-based approach for the self-localization of autonomous robots based on local visual information of landmarks. The goal is to determine the position and the orientation of the robot sufficiently enough, despite some strongly incorrect visual information. Our approach to solve this problem makes use of case-based reasoning methods.

1 Introduction

In order to enable robots to make goal-oriented decisions, their position and direction are very important. If the operational area of the robot is known in advance, e.g. given by geographical maps, the robot should be supplied with this information to improve its localization skills. Our approach relies on information about absolute positions of landmarks in the operational area of the robot. This information is suitable for a case base. With such a case base the robot can determine its position and orientation in its operational area based on perceived sizes and shapes of landmarks and on perceived angles between pairs of landmarks.

In this article we only describe the determination of the robots position. Its orientation can be determined easily in a next step using the already determined position or it can be included into the described approach.

We are using the fully autonomous “Sony Legged Robots” [3]. These are small dog-like robots, that play soccer in teams of 3 robots at the Robocup competitions. At the games the robots are entirely independent, global view or remote control is not allowed. Furthermore, extensions of the robots hardware are prohibited. The robots play on a small field, which contains some well defined markers and goals of different colors. The “Sony Legged Robots” utilize a camera for input and an integrated video processing unit for discriminating between the different colored markers and goals.

* This research has been supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316).

In the next section we describe the properties of the “Sony Legged Robots” and present their environment. In section 3 we describe our approach in detail. Afterwards we give an overview to some related work before we demonstrate some results of our approach in section 5.

2 The Robot and its Environment

Sony kindly provided us with four of their four-legged robots (ERS-1100), which are marketed by Sony in a similar configuration, known as AIBO (ERS-110 and ERS-111). With this platform there come some restrictions which are very close to the properties of biological systems. On the one hand these restrictions raise some problems, on the other hand they represent interesting challenges for the use of intelligent techniques.

The robot contains a camera, which has a very limited field of view¹. The head, to which the camera is affixed in the front, has three degrees of freedom. With these the robot is able to overcome the disadvantages of the small field of view of the camera. Therefore the robot has the possibility to support its self localization by turning its head into a direction where it expects additionally landmarks. This can usually be done without interfering with the robot’s current actions². The integrated video processing unit performs a fast color separation into 8 colors, which is highly sensitive to temporal and spatial changes of the light, like flashes, reflections and shading. Such changes can heavily influence the shape and size of a color region or, much worse, it is possible that a color region is assigned to the wrong color. Unfortunately the integrated color separation algorithm returns only information about the biggest coherent area for each of these 8 colors. So some information in the pictures maybe already lost at this stage.

As already mentioned, the robot uses four legs for the movement in contrast to common robot architectures. Each leg has 3 degrees of freedom, two at the shoulder joint and one at the knee joint. The walking causes some tilting during movement, which leads to virtual relative movements of objects on the field.

For all other computations besides the color segmentation the robot has a 100 MHz RISC-processor (MIPS R4300) and 16 MB main memory. Figure 1 shows the structure of the dog-like robot.

The field of the robot is equipped with different colored fixed objects, which allow the robot to orient itself. These objects consists of six two-color markers and two one-color goals, which are used for self localization. Furthermore, it is conceivable to use the borders of the field and some other field markings like the middle line or the penalty area line for localization. In figure 2 a picture of the field with its landmarks is given.

¹ The CCD-camera has a resolution of 176×120 pixels and a field of view of approximately $47^\circ \times 30^\circ$.

² Unfortunately this does not apply for actions which involves the head, as the search for the ball.

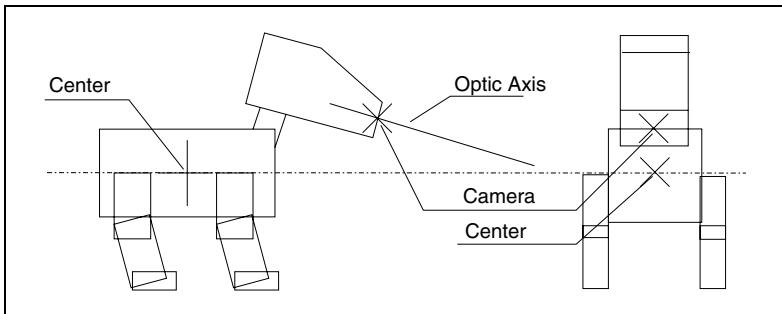


Fig. 1. Structure of the dog-like robot

The field has an effective length of 280 cm and an effective width of 180 cm. The dog itself is approximately 30 cm long and 14 cm wide.

3 Case Base Reasoning for the Self Localization

For the self localization we have divided the state space of field positions into a $14 \times 9 (X, Y)$ grid. This corresponds to a raster of 20 cm \times 20 cm. For each of these states a case was generated, that consists of information on the relative position of all landmarks for this state. To apply the technique of case base reasoning (CBR)[6] in this scenario, several design decisions have to be made. For that the following questions will be answered:

- What is a case? How will cases be represented?
- Where do the cases come from?
- How is the query to the case base determined?
- How is the similarity measure between the cases defined?
- How does the structure of the case base looks like?
- What is the final result?

3.1 Case Structure

Because the determination of the robots position depends on perceived sizes and shapes of landmarks and on perceived angles between pairs of landmarks, this information has to be a part of the case base. We are using the term landmark only for the elemental parts of markers and goals. Therefore a marker will be seen as two landmarks (one landmark for every color). With this view the uniqueness of landmarks seems to be lost, but it is still coded by small angles (angles of zero or near zero) between two landmarks that represent the same marker. The advantage of this view takes effect if parts of a marker are covered by other robots or if only parts of a marker are identified by the robot, due to shading for instance. In such cases the information about the identified part of the marker isn't lost. Furthermore this view is more universal, because it permits different landmarks with identical appearance. According to this view a case consists of the following properties:

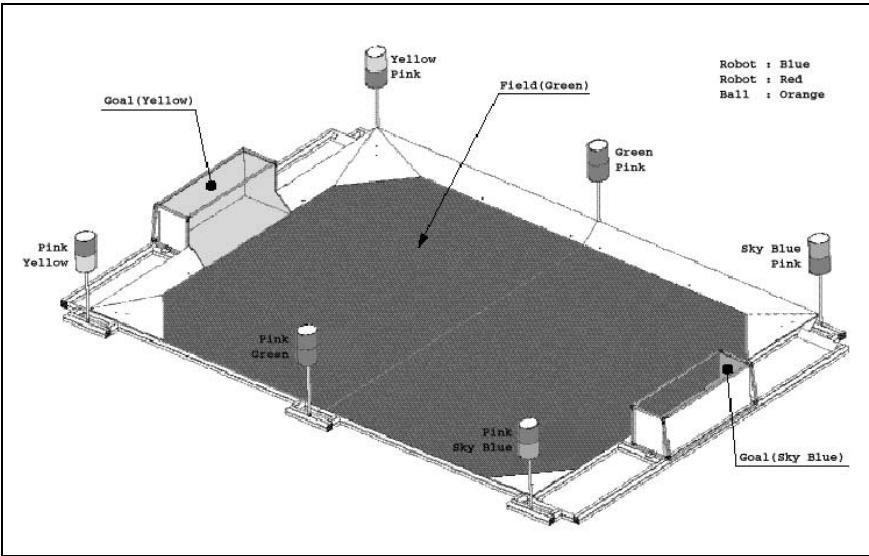


Fig. 2. The field with its landmarks

Size of all landmarks and their position in the picture: For every case the width and the height of all 14 landmarks (six times pink, three times yellow, three times sky blue and two times green) are saved. Width and height indicate the amount of pixels of that color in the corresponding dimension. With the separation of the markers into their elemental parts the information about whether the landmark corresponds to an upper or to a lower part of a marker is lost. For that reason this information is included in the case. For each landmark this is done by its position in the picture. If the landmark is a known part of a marker its position has the value “top” or “down”, if it is an unknown part of a marker its position has the value “unknown”. The last case only occurs in the queries. If the landmark is a goal its position has the value “irrelevant”.

Angle between pairs of landmarks: Whereas the perceived size of landmarks may have a relatively high error because of changes of the light, the error of the angles between pairs of landmarks is more limited. If the robot is moving a lot the error can get a higher value. For that reason we decided only to look at the pairs of landmarks which are directly or indirectly (exactly one marker between the two landmarks) neighboring and at pairs which represent the same marker. So we have 6 pairs where each pair represent one marker, 24 direct and 24 indirect neighboring pairs.

Every property which is assigned with a fixed value will be called an information entity (IE) in the following. A case consists of a set of IEs, here there are 68 IEs. Theoretical we could have up to *number of cases multiplied by number of IEs per case* different IEs ($14 \times 9 \times 68 = 8568$). But many IEs are shared

among different cases so we have a total of 859 different IEs in our case base. A case represents an omni-directional picture of the robot. The solution of the case is the position of the robot on the field from which the picture was taken.

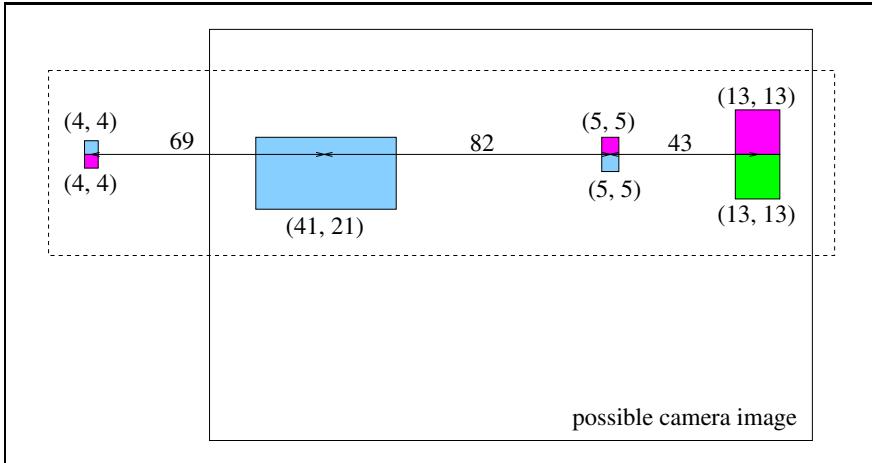


Fig. 3. Graphical excerpt of the case base

In figure 3, a graphical extract of the case base is presented for the case where the robot is at position (110, 60), which is the position marked with a black circle in figure 6. The values in figure 3 are given in pixels of the dog's camera. The figure consists already of 24 IEs, 7 of them are representing the size of landmarks and its position in the picture. The other 17 represent angles between pairs of landmarks (3 pairs represent the same marker, 8 pairs represent direct neighboring pairs, and 6 pairs represent indirect neighboring pairs).

3.2 Where do the Cases come from?

The case base was generated semi-automatically in two steps.

In the first manual step, a table was created which maps the distances of landmarks to perceived sizes of these landmarks. Using the robots camera, pictures of landmarks were taken from different distances (5 cm to 3,20 m distance) and the sizes (in pixel) of the landmarks were determined. These measurements were done with one marker and one goal three times and the mean value was calculated.

In a second automatic step, the distances to all markers and goals and the angles between all relevant markers and goals were calculated for all the 14×9 positions on the field. After that the markers were separated into landmarks and the distance to the landmarks were replaced by the determined sizes. Hence the case base for the self localization was developed.

3.3 Determination of the Case Base Query

The color detection module supplies the self localization module with data about landmarks in the picture. For every detected landmark the following values are supplied:

- its color identifying information,
- its size (width and height) in pixel,
- its balance point in the camera picture as (x, y) coordinate, and
- the orientation of the head at the picture creation time.

Using the x-coordinate of the balance points it is possible to determine the *angles between pairs of landmarks*. In a first step the angles will be normalized according to the optic axis of the robot. After that the angles of landmark pairs are calculated. Small angles indicate that this pair represents the same marker. By comparing the y-coordinates of such pairs of landmarks the *position of the landmark* (“top” or “down”) *in the picture* can be determined. The position of any landmarks whose position isn’t set in this manner is treated as “unknown”. The orientation of the head is used to normalize the relative angles (normalized to the optic axis) of landmarks according to the orientation of the robot. With this normalization it is possible to use informations about landmarks from older pictures despite head motions. This holds as long as the robot is only moving marginally.

The case base query is generated using as much visual information about landmarks as possible to smooth the errors of the observed data which is sometimes very big and other times very small.

In figure 4 an observation of two landmarks is shown. It shows a pink (dark) landmark at the top with an balance point of (72,96) and a size of (44,38), and a sky blue (light) landmark at the bottom with an balance point of (70,67) and a size of (40,21). Because the x-coordinates of these two landmarks does not differ so much, the angle between these landmarks is very small and therefore the two landmarks represent the same marker. Because the y-coordinate of the pink landmark is greater its position in the picture has the value “top” whereas the value is “down” for the sky blue landmark.

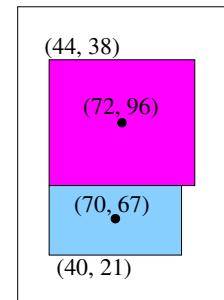


Fig. 4. Observation of two landmarks

3.4 Similarity Measure

The self localization module gets unambiguous color identifying information from the color detection module. Therefore a similarity comparison is only done between IEs with the same color identifying information. For this reason, we have separated the IEs into different categories.

The IEs, which represent the *size of landmarks and their position in the picture* has been separated into four categories (for every color, one category).

Two of these IEs of the same color are similar if their size difference is small and their position in the picture is equal or one of the positions is “unknown”. The smaller the size difference, the more similar the IEs are. An “unknown” position of the landmark in the observed picture decreases the similarity to all IEs with positions “top” or “down” in the case base. If the query consists of the IE for the pink (dark) landmark of figure 4, all pink IEs whose position is “top” and who have a small size differences to the size (44,38) are similar to the query IE. The IE with the size (42,38) is more similar to the query IE than the IE with the size (40,36).

For the IEs which characterize *angles between pairs of landmarks*, the separation into categories is done according to their color combination. Because the combination (green, green) does not exist among the selected pairs, we get 9 categories here. A high similarity is given here between two IEs if its angle difference is small.

Given these local similarity measures, we could define a composite similarity measure which computes the weighted sum over all local similarities where all IEs have the same relevance. We instead use a different composite similarity measure where an IE has less relevance the more often this IE appears on the field. Given this, the IEs for the sizes of landmarks with the color pink has much less relevance than the IEs for the sizes of landmarks with the color green.

3.5 Structure of the Case Base and Retrieval

A crucial point of the self localization is that it has to be done in real time. An efficient organization of the case memory and of the retrieval procedure is needed. For that reason we apply the Case Retrieval Net (CRN) model[7].

In a CRN the case base is represented by a net of nodes for the IEs and by nodes which mark the cases. IE nodes might be connected among each other by similarity arcs. Every case node is connected with all its IEs nodes by relevance arcs. Based on this structure the retrieval works as follows:

1. activation of the IEs from the query,
2. propagation of the activation through the net along the similarity arcs, and
3. propagation of the activation reached so far to the case nodes along the relevance arcs.

Whereas the similarity arcs represent the local similarity measure, the composite similarity measure is represented by the relevance arcs. Different strength of similarity and relevance can be represented by weighting the arcs.

3.6 Determination of the Final Result

After having done the case based retrieval we get a list of cases which represents most likely positions of the robot. Every case is activated by some value, the higher the value the more likely the robot is at the corresponding position.

For the determination of the robot’s position we consider all cases which have a similar activation to the maximum activated case (80% activation or more) and

are not more than 50 cm away from the maximum activated case. A weighted sum over all these cases is computed to determine the robot's position.

4 Related Work

All the groups we are aware off, which focus on similar or even the same problems as we do in our approach, are using probabilistic methods for the self localization and robot navigation. In [8] the authors are using partially observable Markov models to robustly track a robot's location in office environments. Furthermore Markov models have been employed successfully in various mobile robot systems like [1, 4, 8]. Even in the same domain, for the "Sony legged robots", these models have been applied [5].

With our completely novel approach we want to evaluate whether we can reach similar results by applying case based techniques. We predict that our approach has some advantages and some disadvantages in relation to the Markov model approach. For instance we expect that our approach is faster if the information set of perceived landmarks is low whereas our approach maybe slower in cases where many landmarks can be perceived. Furthermore we think that active localization [2] can be supported more easily by a case base structure than by a Markov model.

5 Results and Future Work

To test and evaluate our approach we performed three simulated experiments and one real-world experiment. The simulated experiments were done automatically without using the robot or the field, whereas the real-world experiment was done manually with the robot on the field.

In our first simulated experiment we randomly chose 1000 positions on the field. For each of these positions, 10 virtual pictures were generated which consisted of 2 to 14 landmarks. For each of these 10000 pictures, IEs were created using the landmarks and incorporating an error between -20% and 20% of the correct value. These IEs were used as queries for our localization approach and the error between the correct position and the computed position was determined. The average error's dependence on the number of landmarks in the virtual picture is shown in figure 5 with the solid line. In the second experiment we incorporated a 50% possibility that the information about whether a landmark is the "top" or the "bottom" of a marker was lost. The results of this experiment is represented by the dashed line in the same figure. In the last simulated experiment, whose results are represented by the dash-dotted line, all 10 pictures of one position had the same number of landmarks (randomly chosen between 2 and 14). The IEs were created using the same error as above but this time every IE of each of these 10 pictures was used as localization query. This experiment is quite natural because the robot usually takes more then 10 pictures in a second and all these pictures have errors but different ones. In figure 5, only the average errors for the three experiments are shown, but sometimes the error was much

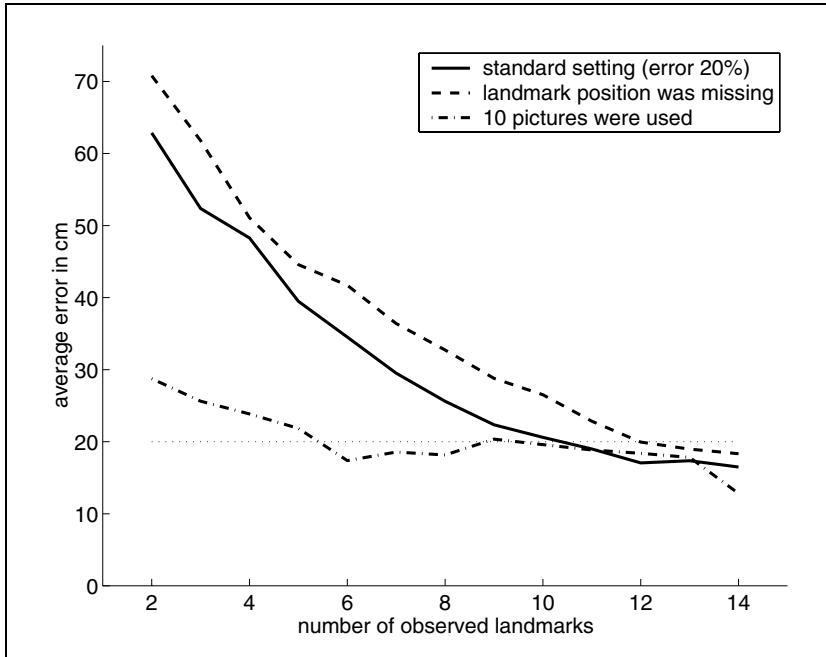
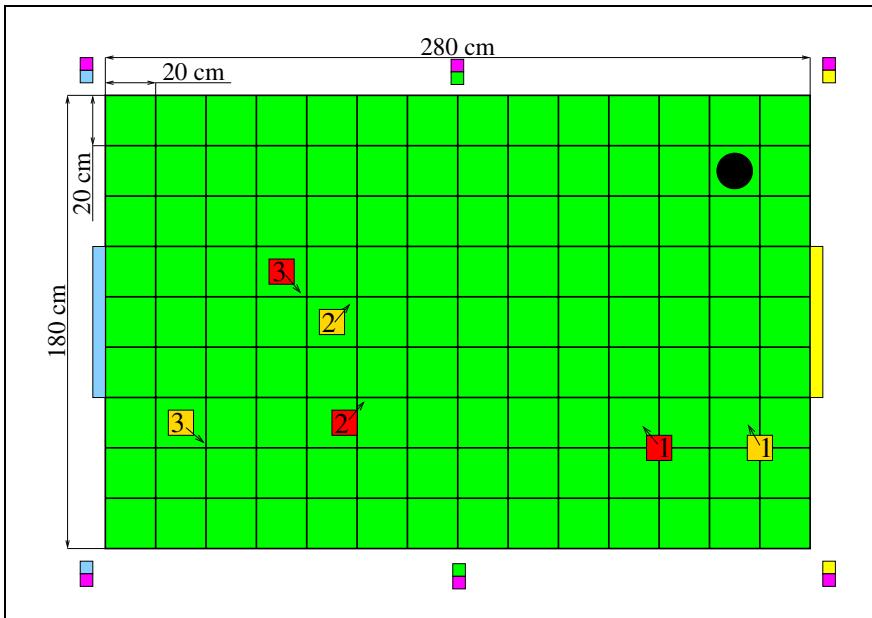


Fig. 5. Simulated Results

bigger. This happens if the incorporated errors modify the virtual picture in a manner such that it “looks” like a virtual picture from another position.

For our real-world experiments the robot was put onto the field and allowed some time to localize itself by turning its head and taking pictures. The actual location and the result computed by the robot were compared. Figure 6 shows three (out of 30) typical examples from this experiment. The dark squares show the actual and the light the computed positions of the robot. The figure also shows the given and the computed orientation of the robot. The orientation was computed using the determined position and one fully observed marker or goal. Although the robot sometimes had problems determining its position well enough (in example 3 it had an error of 75 cm), it was always pretty good at determining its orientation (maximum error of 10°).

The simulated and the real results already show the usefulness of the approach, even though we are not fully satisfied with it. During evaluation we discovered that the IEs which represent angles between landmarks don't include the orientation of the angle, so we lost the information about which of the landmarks were right or left. Furthermore the algorithm to determine the final result out of the activated cases needs improvement. Up to now the robot localizes itself using only the most current information about landmarks. We also want to incorporate older information and the robot's movements into the case base approach as well as enhancing our approach with active localization.

**Fig. 6.** Real Results

References

1. W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996.
2. D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. In *Robotics and Autonomous Systems*, 1998.
3. M. Fujita, S. Zrehen, and H. Kitano. A Quadruped Robot for RoboCup Legged Robot Challenge in Paris '98. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, volume 1604 of *LNAI*, pages 125–140. Springer Verlag, 1999.
4. L. Kaelbling, A. Cassandra, and J. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 1996.
5. S. Lenser and M. Veloso. Sensor Resetting Localization for Poorly Modelled Mobile Robots. In *Proceedings of ICRA-2000*, 2000.
6. M. Lenz, B. Bartsch-Spörl, H. D. Burkhard, and S. Wess. *Case Based Reasoning Technology. From Foundations to Applications.*, volume 1400 of *LNAI*. Springer, 1998.
7. M. Lenz and H. D. Burkhard. Lazy propagation in Case Retrieval Nets. In W. Wahlster, editor, *12th ECAI 1996*, pages 127–131. John Wiley & Sons, 1996.
8. R. Simmons and S. Koenig. Probabilistic Robot Navigation in Partially Observable Environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

PINO The Humanoid : A Basic Architecture

Fuminori Yamasaki^{1,2}, Tatsuya Matsui¹
, Takahiro Miyashita¹, and Hiroaki Kitano¹

¹ Kitano Symbiotic Systems Project, ERATO, JST,
yamasaki@symbio.jst.go.jp,

² Osaka University

Abstract. In this paper, we present a basic architecture and design principle behind Pino, a humanoid for RoboCup Humanoid League. Pino is designed to be a platform for research in robotics and AI. There are four major issues in Pino's design; (1) high DOF system to realize various behaviors, (2) exterior design, (3) cheap mechanical components, and (4) a basic behavioral control systems. These issues are specifically addressed in this paper in order to illustrate basic architecture and components for humanoid platform that can be widely used for many RoboCup researchers in the world.

1 Introduction

RoboCup Humanoid League [1], expected to start from 2002, is one of the most attractive research target, as well as potential major media attraction. We believe that the success of the humanoid league is critical for the future of RoboCup, and have major implications in robotics research and industries. Thus, we believe that it is important basic architectures for cheap and reliable research platform for humanoid to be quickly established, so that researchers in RoboCup community can initiative further investigation.

In this paper, we present an example of basic architecture of humanoid for one of RoboCup humanoid league. There are three major issues addressed in this paper.

First, humanoid should have a well designed exterior. As robot became everyday-life products, exterior designs plays major role in consumer choice. Since most existing research robots are functionally designed with minimal aesthetics, carefully designed robots are not yet exposed to general public, with exception of AIBO[2] and SIG [3]. We claim that "robot design" will be the major industrial design field, and wish to present archetype of humanoid design through Pino.

Second, the use of cheap, off-the-shelf components made humanoid more accessible for broad-range of researchers. It is a general impression that humanoid research is extremely expensive, thus it is beyond the capability of the most research group. This is partly true if we are to design all components from scratch aiming at Honda P3 [4] level humanoid. However, we believe that there is a room in the humanoid research that many interesting research can be done

using cheap and off-the-shelf components affordable to most research groups. The hidden name of our project is, in fact, "The Poorman's humanoid project".

Third, we are working on possible modular and basic set of primitive behaviors and their control systems that are essential for most humanoids. While humanoid has very high DOF, some of primitive and basic behaviors may be definable as a hopefully modular and primitive control sub-unit. If our attempt is successful, more complex behaviors can be built on top of library of primitive high-DOF control systems.

This paper describes how we approach these issues within Pino the humanoid.

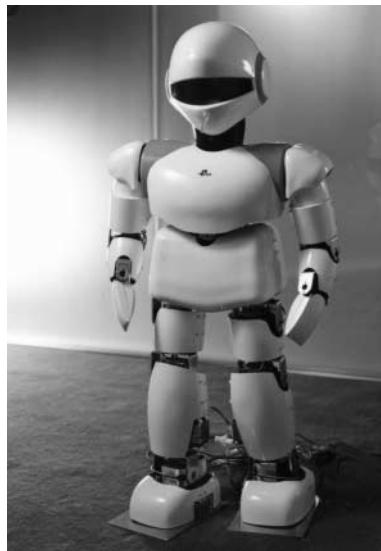


Fig. 1. PINO

2 Exterior design

2.1 Necessity of Exterior Design

Pino, the humanoid robot was developed to participation Robocup Humanoid League. The importance of exterior design is connected first and foremost to the protection of its inside system in much the same way a car or computer is similarly shielded from contact. Still, providing a mere protective shield to reduce the risk of damage to its inner systems during performance could not sufficiently express our research direction which aims to express the role of the humanoid robot in society in the future. Thus our method towards the aim of giving meaning to the existence of the robot necessitated the creation of a story to explain not only its design elements but its role in a society of the future.

2.2 Robot Design-”PINO” the Design Concept

Before Pino went into development, we discussed what form, and just as importantly what size would be necessary to ensure its comfortable integration into the human home. Its size, we discovered was a very specific element of our design research which was carried out simultaneously during our primary research into its walking functions.

The scale of a fully grown adult posed a threatening presence and would, we believed cause a general sense of unease, being less a companion than a cumbersome and overpowering mechanical object. Thereupon, we judged the ideal size for such a robot would be 70 [cm] tall; the approximate size of a one year old child taking its first steps. As for form, it was deemed necessary to design its proportions as recognizably human as possible; deviating too far from the instantly recognizable form of a human child could cause it to be seen as an altogether different object. From a psychological point of view, we noted that even the casual observer focused more attentively , and ultimately more affectionately upon a similarly structured form.

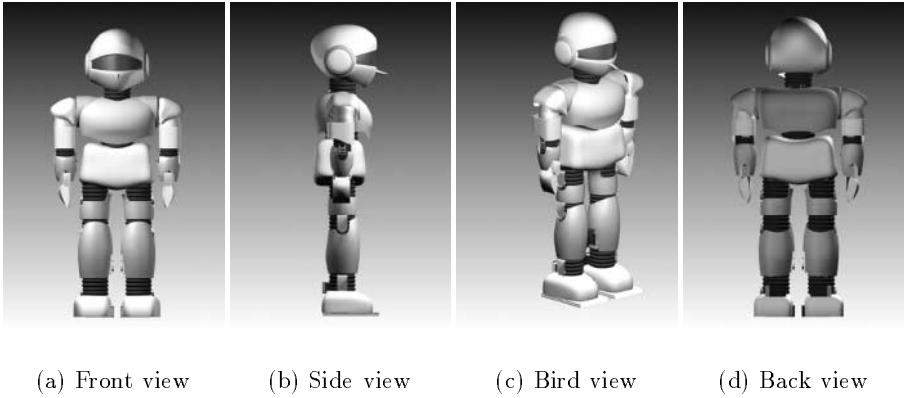
2.3 The origin of the PINO name

Before starting the design sketches we searched for a universal element in the representation of the human form. Images handed down from the archives of such representational forms, we believed, provided the key to integrating into the future what has universally been acknowledged as fundamental beauty. Ancient Greek sculptures and more significantly, as it turned out, the more folkloric marionettes to name a few were evoked, not only for their obvious beauty, but just as importantly for their mournful aspirations towards perfection. The marionette, with its mechanisms to facilitate movement and expression provided the ideal framework by which Pino could be adapted. Pinocchio, whose story needs no explaining, seemed an apt metaphor for our search for human qualities within the mechanical structures of our creation. We felt the necessity to place Pino somewhere in time, more specifically within the context of a story to elevate him above the static realm of the object. For this reason the infant sized cherub was discarded as a possibility despite the temptingly poetic implications of such a creature. An angel, knowing only flight finds his legs useless appendages, and Pino, whose primary function is walking, is unsuited for such dimensions. In his gestation, Pino symbolically expresses not only our desires but humankind’s frail, uncertain steps towards growth and the true meaning of the word human.

3 Structure of PINO

3.1 Configuration of DOFs

Humanoid should have high DOFs to achieve various kinds of behaviors. In Robocup, for example, it search a ball and two goals, and moves to its desired location with avoiding many obstacles. Then it shoots or passes the ball. These

**Fig. 2.** Computer graphic design

maneuver mainly involve following four fundamental motions; (1) keeping its body's balance, (2) moving the swing leg, (3) operating a grasping object, and (4) controlling visual attention.

To keep its body's balance, it needs 6 DOFs which denote motions of a center of gravity of itself. It also needs 6 DOFs to move the swing leg to its arbitrary location. Handling or operating a grasping object needs 6 DOFs, too. Controlling its visual attention needs 6 DOFs to move the visual sensor to arbitrary positions and orientations. Therefore humanoid needs 24 DOFs to achieve these motions in parallel. Thus, we decided the configuration of Pino as follows. Each leg has 6 DOFs, each arm has 5 DOFs, the neck has 2 DOFs and the trunk has 2 DOFs. Totally Pino has 26 DOFs (see Fig.3).

3.2 Actuators and Sensors

Pino has 26 motors which correspond to the number of DOFs. And it needs various kinds of sensors, For example, visual sensor for recognizing objects, posture sensor for detecting its body's balance, force and tactile sensor for detecting contact to others and falling down, and so on. If we are to design all components from scratch aiming at honda P3 level humanoid, it is extremely expensive. It is also better to use cheap components for actuators and sensors because it is considered that humanoid contact to obstacles or fall down. There are servo modules (SM) for radio control model as a cheap, compact and high torque servo motor. These SMs built in gearhead and its position controller, and these servo loops run at 50 [Hz]. These robots are already introduced which consist of SMs[5][6]. We adopted two kinds of Futaba SM for Pino which torque are 20 [kg·cm] and 8 [kg·cm] (see Fig.4 (a)). All of their gears are changed to metal for reinforcement them against high torque. Pino has 8 force sensors (FSR) (see Fig.4 (b)) attached

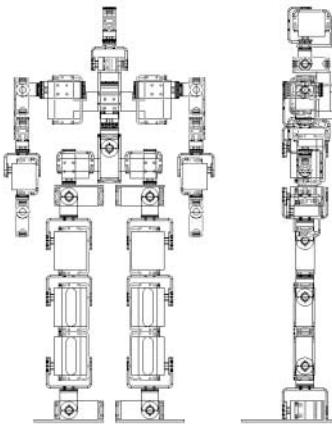
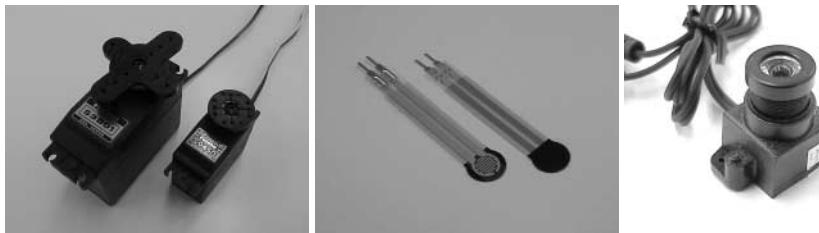


Fig. 3. Mechanical design

to each foot which can obtain foot forces, and has a posture sensor on chest so that it can sense its body's posture. we also use potentiometers in SMs as joint angle sensors. Pino also has a vision sensor mounted on head (see Fig.4 (c)). All information of these sensors are sent to host computer via A/D converters and tracking vision board.



(a) Servo module (b) Force sensor (FSR) (c) Vision sensor

Fig. 4. Actuators and sensors for Pino

3.3 Processing system

The control system for Pino consists of a host computer and a controller of SMs. The host computer obtains all information of sensors via A/D converters and tracking vision board. The host computer consists of Pentium II 450 [MHz] processors and 512 [MByte] memory, and its operating system (OS) is realtime

OS (RT-Linux). From these information of sensors, the host computer calculate angular velocity of each joint on realtime. Pino also has controller of itself which consists of a RISC micro-computer Hitachi SH2 (SH7050) and its slave ALTERA Programmable Logic Device (Flex10K30AQCC240-3, hereafter PLD). This controller communicates the host computer via RS-232C. The PLD has 26 submodules of SM controller in it. Each submodule generates position commands for SM. The servo loops run at 50 [Hz], and the position commands for 26 SMs are updated simultaneously. We can use the development tool for this PLD for free.

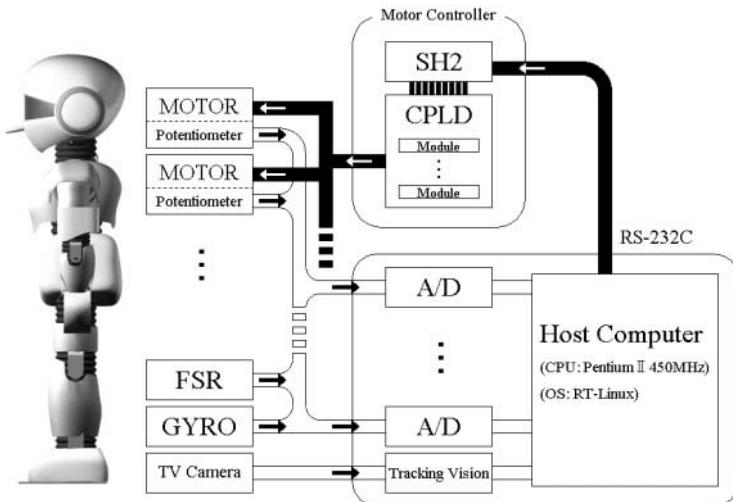


Fig. 5. System configuration

4 Control for the PINO

Actuators of Pino consist of 26 sets of motor and its driver for position control. However, speed control is more useful for controlling dynamic motions. Therefore, in order to realize speed control with them, we make quasi speed control system by utilizing direct kinematics of Pino and Jacobian matrices which denote velocity relationships between joint angles and positions and posture of each foot with respect to a robot coordinate frame fixed to the robot body. In this section, we explain the foot position controller of the Pino briefly.

Let Σ_R , Σ_{R*} , Σ_{L*} , l_{R*} , l_{L*} , ${}^R\mathbf{p}_{RE}$, ${}^R\mathbf{p}_{LE}$, ${}^R\mathbf{R}_{RE}$ and ${}^R\mathbf{R}_{LE}$ denote a robot coordinate frame fixed to the robot body, link frames, link lengths of right and left legs, positions and rotation matrices of right and left legs with respect to

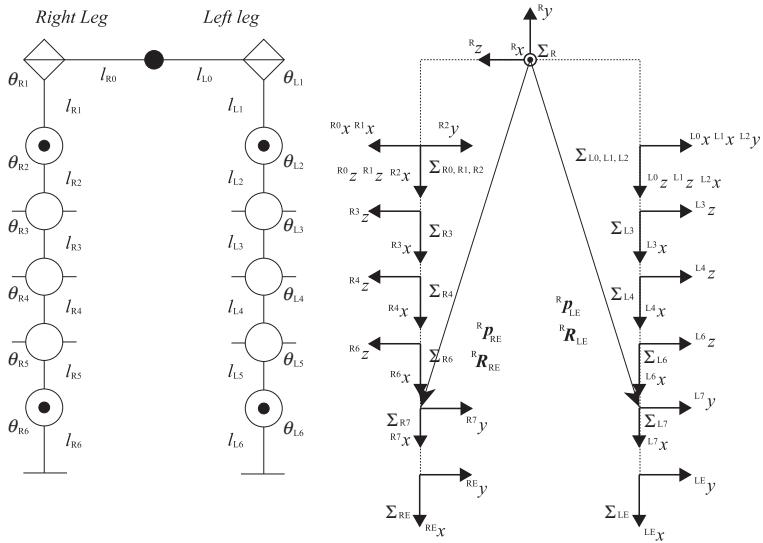


Fig. 6. Coordinate frames of legs of the Pino

the robot coordinate frame, respectively (see Fig.6). We can calculate homogeneous coordinate transformation matrices, ${}^R\mathbf{T}_{RE}$ and ${}^R\mathbf{T}_{LE}$ which denote the positions and posture of each foot with respect to Σ_R ,

$${}^R\mathbf{T}_{RE} = \begin{bmatrix} {}^R\mathbf{R}_{RE} & {}^R\mathbf{p}_{RE} \\ \mathbf{O} & 1 \end{bmatrix} = {}^R\mathbf{T}_{R0} {}^{R0}\mathbf{T}_{R1}(\theta_{R1}) \dots {}^{R5}\mathbf{T}_{R6}(\theta_{R6}) {}^{R6}\mathbf{T}_{RE}, \quad (1)$$

$${}^R\mathbf{T}_{LE} = \begin{bmatrix} {}^R\mathbf{R}_{LE} & {}^R\mathbf{p}_{LE} \\ \mathbf{O} & 1 \end{bmatrix} = {}^R\mathbf{T}_{L0} {}^{L0}\mathbf{T}_{L1}(\theta_{L1}) \dots {}^{L5}\mathbf{T}_{L6}(\theta_{L6}) {}^{L6}\mathbf{T}_{LE}, \quad (2)$$

where ${}^i\mathbf{T}_j$ denotes homogeneous coordinate transformation matrix from Σ_j to Σ_i calculated from link parameters. Differentiating equations (1) and (2), we can obtain velocity relationship between $\dot{\boldsymbol{\theta}} = [\dot{\theta}_{R1} \dots \dot{\theta}_{R6} \dot{\theta}_{L1} \dots \dot{\theta}_{L6}]^T \in \Re^{12}$ and ${}^R\dot{\mathbf{r}} = [{}^R\dot{\mathbf{p}}_{RE} \ {}^R\omega_{RE} \ {}^R\dot{\mathbf{p}}_{LE} \ {}^R\omega_{LE}]^T \in \Re^{12}$,

$${}^R\dot{\mathbf{r}} = \mathbf{J}_{r\theta}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}, \quad (3)$$

where ${}^R\omega_i$ and $\mathbf{J}_{r\theta}(\boldsymbol{\theta})$ denote angular velocity vector of Σ_i with respect to Σ_R and $\partial {}^R\mathbf{r}/\partial \boldsymbol{\theta} \in \Re^{12 \times 12}$, respectively. From equation (3), a feedback controller for joint velocities to move each foot to desired positions ${}^R\mathbf{r}_d$ can be derived as

$$\mathbf{u}_r = \mathbf{J}_{r\theta}^+ \mathbf{K}_r ({}^R\mathbf{r}_d - {}^R\mathbf{r}) + (\mathbf{I}_{12} - \mathbf{J}_{r\theta}^+ \mathbf{J}_{r\theta})k, \quad (4)$$

where $\mathbf{J}_{r\theta}^+$, \mathbf{K}_r , \mathbf{I}_{12} and k denote a pseudo inverse matrix of a matrix $\mathbf{J}_{r\theta}$, a gain matrix, a 12×12 identity matrix and an arbitrary vector that describes redundancy of the robot with respect to this servoing task. An input vector of

the actuators of Pino is, however, a joint angular vector. Therefore we make the input vector by multiplying \mathbf{u}_r by sampling rate ΔT and adding previous joint angular vector. The block diagram of it is shown in Fig.7.

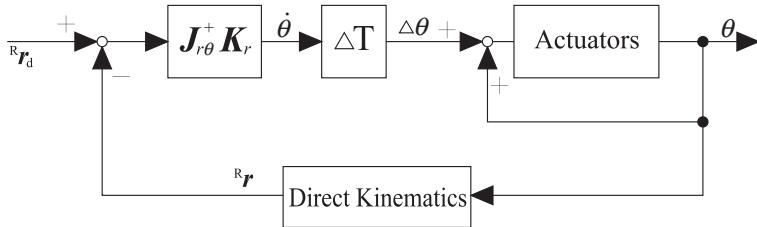


Fig. 7. A block diagram of controlling foot positions

If Pino goes into the real world and achieve their complicated tasks described in section 1, it needs capabilities to generate many motions and adapt to various situations of the environment. In previous work for the legged robot, Miyashita et al. [7] proposed a reflexive walk of a quadruped robot based on reflexes to realize an adaptive walk in a dynamic environment. They applied two reflexes, a vision-cued swaying reflection and a reflective gait, to the robot. A combination of them makes the robot walk reflexively without programming the exact motion of each joint of the legs.

5 Experiments and Discussion

In this section, We had the preliminary experiment that we made Pino walk to give the trajectory of each joint in order to verify the Pino's system performance. We built a kinematic model of Pino and applied it to the controller of positions and posture of each foot described in section 4. We assume that the motion sequence

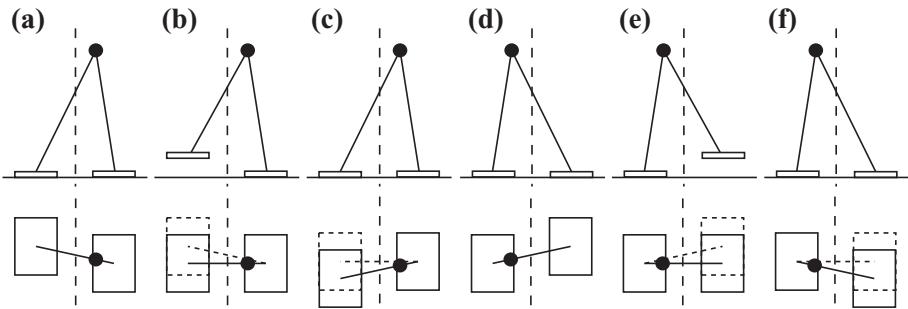


Fig. 8. The motion sequence of one cycle

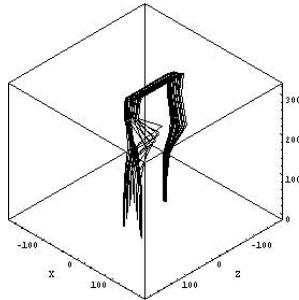


Fig. 9. Trajectories of each joint (the first step)

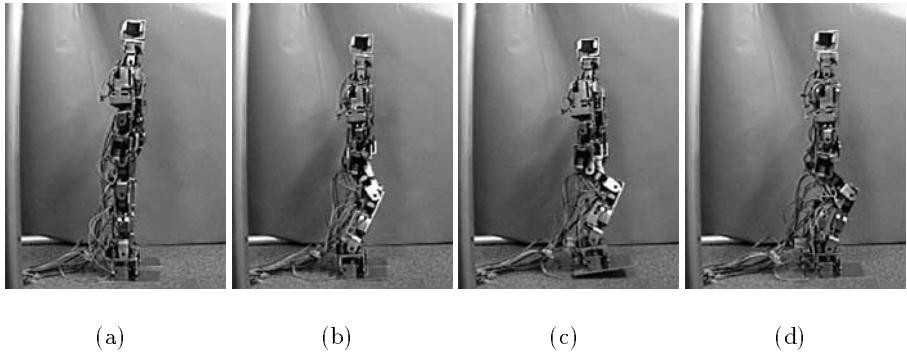


Fig. 10. Experimental result (the first step)

of one cycle of walking consists of 6 phases (see Fig.8). We generated the trajectories based on the motion sequence in advance. The example of trajectories of each joint is shown in Fig.9. We gave the trajectories to Pino, and made it walk in the real environment. As the result, Pino can walk based on them. Fig.9, Fig.10 show the first step of the result of this experiment.

In this experiment, Pino often falls down because we only use the interoceptive information. Therefore it needs to use exteroceptive information in order to realize adaptive walk. Besides, it needs to make the online system which can generate various kinds of behavior using them.

6 Conclusion

In this paper, we described how we approach our issues within Pino the humanoid. Pino was designed so that it is enable for most researchers to study for

the RoboCup Humanoid League or some humanoid researches. It consists of 26 SMs, 8 force sensors, a posture sensor, a vision sensor and its control system, and all of them are cheap components. In the preliminary experiments, we verified the performance of Pino that it can walk in the real environment.

In the future work, we would like to make the online system which can generate various kinds of behavior based on exteroceptive information using a vision sensor, force sensors, a posture sensor and potentiometers.

References

1. Kitano, H., Asada, M.: RoboCup Humanoid Challenge: That's One Small Step for A Robot, One Giant Leap for Mankind Proc. of International Conference on Intelligent Robots and Systems (IROS-98) (1998)
2. Fujita, M., Kitano, H.: Development of an Autonomous Quadruped Robot for Robot Entertainment. Autonomous Robots, 5, (1998)
3. Kitano, H., Okuno, H. G., Nakadai, K., Fermin, I., Sabish, T., Nakagawa, Y., Matsui, T.: Designing a Humanoid Head for RoboCup Challenge Proc. of Agent 2000 (2000)
4. Hirose, M., Takenaka, T., Gomi, H., Ozawa, N.: Humanoid Robot. The Journal of the Robotics Society of Japan (JRSJ) vol.15, no.7 (1997) (in Japanese)
5. Inaba, M., Kanehiro, F., Kagami, S., Inoue, H.: Two-armed Bipedal Robot that can Walk, Roll Over and Stand up. Proc. of International Conference on Intelligent Robots and Systems (IROS-95) (1995)
6. Inaba, M., Igarashi, T., Kagami, S., Inoue, H.: A 35 DOF Humanoid that can Coordinate Arms and Legs in Standing up, Reaching and Grasping an Object. Proc. of International Conference on Intelligent Robots and Systems (IROS-95) (1995)
7. Miyashita, T., Hosoda, K., Asada, M.: Reflective walk based on lifted leg control and vision-cued swaying control. Proc. of 1998 BSMEE International Symposium on Climbing and Walking Robots (CLAWAR'98) (1998)

Team/goal-keeper coordination in the RoboCup mid-size league

Giovanni Adorni[†], Stefano Cagnoni ^{*},
Monica Mordonini ^{*} and Maurizio Piaggio[†]

^{*} Department of Computer Engineering, University of Parma,
[†]Department of Communication, Computer and System Science, University of Genoa,
E-mail: {adorni,cagnoni,mordonini}@ce.unipr.it, piaggio@dist.unige.it

Abstract. In this paper we describe the coordination strategies that were designed to achieve effective cooperation between a robot goal-keeper and the rest of the ART (Azzurra Robot Team) team that participates in the RoboCup F-2000 (mid-size) competitions.

The paper introduces the multi-robot environment on which cooperation in ART is based and, in particular, its communication sub-system. Some case studies and the results of their application in the ART team are then described.

1 Introduction

In this work coordination is discussed in the framework of the RoboCup, the Robot World Cup initiative, an international research activity aimed at fostering robotics and artificial intelligence technologies using soccer as a common task [1]. Azzurra Robot Team (ART), the Italian RoboCup team, is a national project involving several academic groups in a consortium¹. The goal of the project is to exploit the expertise and ideas from all groups and build a team where players have different hardware and software features but have the ability to coordinate their behavior within the team.

More precisely, in this paper we describe some general coordination strategies used in ART and, more in particular, the ones that have been designed to achieve cooperation between the goal-keeper and the rest of the ART team in the RoboCup mid-size competition.

2 Coordinating a team of heterogeneous robots

Coordination of robot soccer players in the ART Team was particularly challenging because of a unique characteristic of the team which was clearly visible during the matches: each robot is the research effort of one member of the ART consortium. Therefore, each robot in ART differs from its team mates in many ways: mechanics, sensors,

^{*} Address for correspondence: Parco Area delle Scienze 181/a, 43100 Parma, Italy. Phone: +39 0521 905731. Fax: +39 0521 905723

¹ ART (<http://robocup.ce.unipr.it>) is a cooperative effort of “Consorzio Padova Ricerche”, Polytechnic of Milan, University of Genoa, University of Milan “Bicocca”, University of Padua, University of Parma, and University of Rome “La Sapienza”

computer hardware and control software but most of all designer. The designer in particular was not only different but also operated in a substantially independent way and with not so frequent communication with the other members of the team (due to the distance between the different university sites). This led to a truly heterogeneous team in which the role of coordination became therefore very significant. Coordination was dealt with at two different levels described in the next subsections.

2.1 Low-level communication framework

The low level communication framework is strictly related to the ETHNOS [2] real-time [3, 4] software architecture which was at the base of inter-robot communication in ART. ETHNOS in fact exploits a message-based communication protocol (the EIEP - Expert Information Exchange Protocol [5]) which deals transparently both with inter-process communication within a single robot and with inter-robot communication. Messages are exchanged with a publish/subscribe technique and dynamically distributed by the system to the appropriate receivers.

It is worth noticing that, even though ETHNOS is platform-dependent (it requires a Posix RT(r) compatible kernel such as Linux with RT threads), the EIEP is a more general protocol that runs on any system that provides socket communication. Moreover ETHNOS does not constrain the Robot physical or higher level software control architecture and it has in fact been exploited on the different robots that compose the ART Team. Another important property of the system is the automatic management of agents connection and disconnection. In fact, whenever we want to add (remove) a player to (from) the team, it is not necessary to explicitly inform each player about the modifications in the team's composition. Players just have to agree about the type of information they are ready to send and receive, and ETHNOS deals automatically with the lower level communication details. This has been very important in ART in which more than four types of robots were available, with different characteristics of play, and thus different team compositions were selected for the single matches and also modified during the matches to better contrast the opponent.

2.2 Coordination protocol between the players

Coordination in ART was based on the underlying EIEP described above. The goal-keeper was coordinated in a separate way as described in section 3, and the other three players were organized in a formation with specific roles dynamically assigned during the match. Utility functions and hysteresis were exploited to carry out the dynamic role assignment every 100 milliseconds in an efficient and stable manner. More details on this coordination strategy can be found in [6].

3 Team/goal-keeper coordination: case studies

This section describes some cases of coordination between the goal-keeper and the other players that were specifically studied in designing the autonomous robot shown in figure 1 that plays as goal-keeper in the mid-size ART team [7].

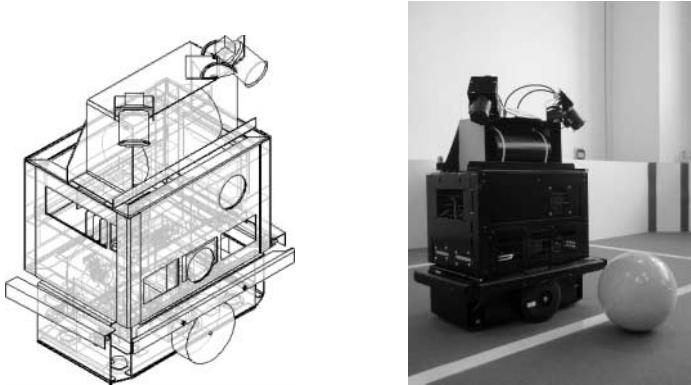


Fig. 1. The ART goalie Galavrón and its CAD model: two wide-angle cameras are on the top. The wheels are mounted in the middle of the chassis. The pneumatic kicking devices located in the lower part are driven by an air tank placed inside the camera holder.

The goalie's role and position inside the field has strongly influenced and constrained its mechanical design, its motion strategy, as well as its software architecture. The latter consists of a multi-Agent system composed of three Agents: a Visual Perception Agent that provides the robot with vision; a Goal-Keeper Agent that provides decision-making skills and controls the motion of the goalie; and a Cooperative Agent which enables the robot to cooperate and co-ordinate its activities with the other robot players [8, 6].

The robot operation is influenced by both the cooperation among the software agents within the robot and the interaction of the robot with its team-mates, that permits to integrate and enhance information about the environment that the robot acquires autonomously.

Critical situations in which goalkeeper/players coordination strategies can be effectively used fall into two major categories, which we may term *implicit* and *explicit coordination*, respectively:

- Goalie's replacement or support: these are situations in which the goalie's performance is impaired by hardware faults, self-localization faults or perceptual limitations. In this case team-mates can realize what is happening and take appropriate countermeasures.
- Defensive behaviors of the players triggered by explicit requests from the goalie.

In figure 2-a.1 the goalie gets stuck and has to be taken out of the field. In this case, the hardware failure is detected when no messages are received from the goalie for a preset amount of time. Consequently (figure 2-a.2), the other team members back up: one player patrols the goalie's operating region (as far as rules allow to do so, which means just outside the goal-keeper area), while one of the other two players takes the Main Defender role.

In figure 2-b.1, the goalie has moved away from the goal, possibly as a consequence of a failure of its self-localization system. This event can be detected by the vision

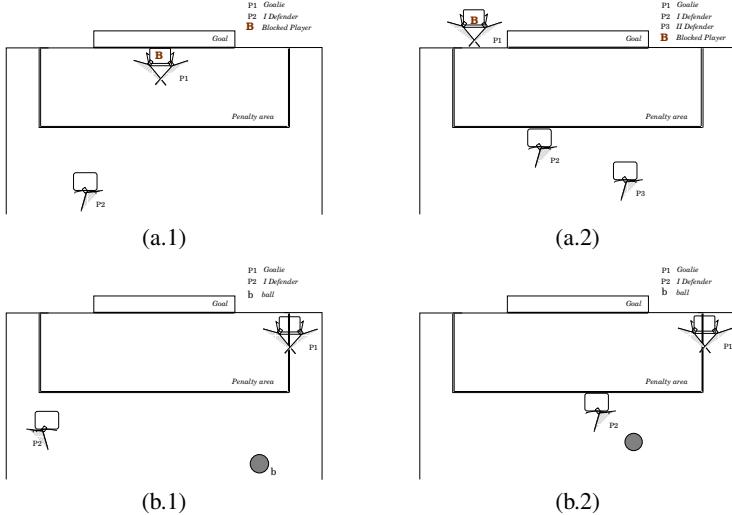


Fig. 2. Two examples of implicit team/goalie coordination: a) the goalie is stuck; b) the goalie is out of position.

system of the closest team-mate. In this case, the team-mate can go to shield the goal (within the above-mentioned limits), until the goalie recovers its correct position.

Another critical situation is the one in which the goalie's view is occluded by another robot that keeps the ball out of the goalie's sight. In this case the goalie can use information about ball coordinates provided by the team-mate and move to the optimum position. This is a very significant example of cooperation that has been successfully tested even in an extreme situation in which the goalie is totally blind (see also section 4).

Figure 3 shows two examples of defensive actions triggered by an explicit request from the goalie. In the first example the goal is threatened by two opponents that could easily elude the goalie's intervention by passing the ball between each other. In this case the goalie explicitly calls back a defender; the latter backs up and obstructs the free space between the opponents, preventing the pass from occurring. In the second example a team-mate lies on the optimal trajectory that the goalie should give the ball in order to send it away from the goalkeeper area (figure 3-a). The goalie then sends a message to the team-mate to make it back up and leave enough space for its sweeping shot to occur (figure 3-b).

4 Experimental results

Some of the previously described situations have been tested in the practice. In a particularly significant experiment the vision system of the goal-keeper had been switched off, so that the goal-keeper was totally blinded and could perceive the surrounding world only through the messages sent by a team-mate that was observing the ball. Such messages replaced the input to the Goal-Keeper Agent, that is usually provided by the Vi-

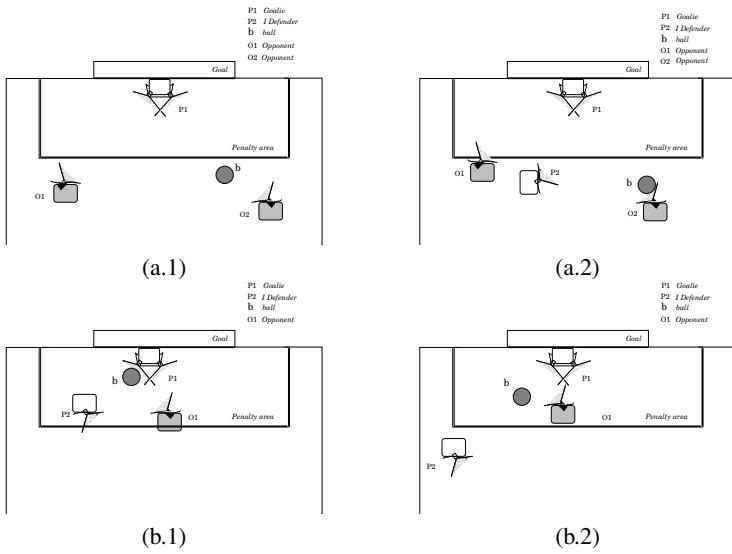


Fig. 3. Two examples of explicit goalie/team coordination: a) the goalie calls back a player; b) the goalie sends a mate away.

sual Perception Agent. Figure 4 shows the setup that has been used for the experiment. The goalie performed quite well, showing basically the same behaviors as in normal operating conditions, with no apparent decay of the reaction time.

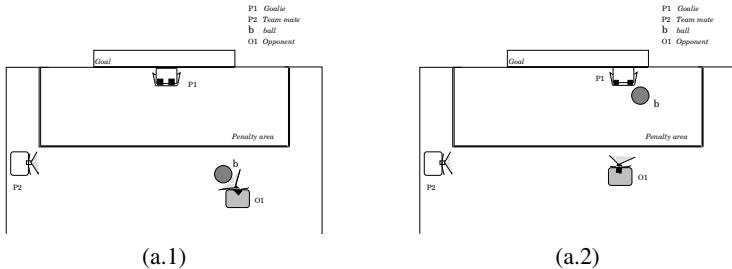


Fig. 4. The goalie is blinded but can operate using the team-mate's messages

This experiment, besides showing the intrinsic effectiveness of the coordination strategies, reflects situations that can occur in several real-world environments in which robots operate. The opportunity of supporting a temporarily or permanently impaired robot in a team with information coming from its mates, creates a “perceptual redundancy”. Exploiting such redundancy may be very important, if not crucial, in several critical tasks, especially when robots operate in regions that are unsafe for humans, or possibly in unmanned space exploration missions.

5 Final remarks

In teams of heterogeneous robots where each player typically has to play a different role an effective coordination strategy is a key component of the team's successful design. The importance of cooperation strategies is getting more and more significant in the F-2000 RoboCup League, where game quality is rapidly evolving from simple individual "locate the ball and shoot" behaviors to more complex and team-oriented ones.

In this paper we have focussed on the coordination between the ART goalkeeper and the rest of the team. Examples have been shown of situations in which such coordination is essential, both when it occurs implicitly as the result of a one-side reasoning process derived from a message broadcast and when it is explicitly implemented through a message exchange between players. Some of the described situations could be already coped with successfully by the coordination strategy used in the past competitions; the global coordination strategy is now being updated to cope with the remaining ones and with new ones in future editions of RoboCup.

Acknowledgements

The authors wish to thank all members of ART, "Azzurra Robot Team".

This work has been partially supported by MURST under the "CERTAMEN" grant and by ENEA under the "Sensori intelligenti" grant.

References

1. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup: the robot World Cup initiative", in *Proc. of the 1st International Conference on Autonomous Agents*. 1997, pp. 340–347, ACM.
2. M. Piaggio, A. Sgorbissa, and R. Zaccaria, "A programming environment for real time control of distributed multiple robotic systems", *Advanced Robotics Journal*, vol. 14, no. 1, pp. 75–86, 2000.
3. C.L. Liu and J.W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment", *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.
4. J.P. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior", in *Proc. IEEE Real Time Systems Symposium'89*. 1989, pp. 166–171, IEEE CS Press.
5. M. Piaggio and R. Zaccaria, "An information exchange protocol in a multi-layer distributed architecture", in *Proc. 30th IEEE Hawaii Conf. on System Sciences*. 1997, pp. 230–238, IEEE CS Press.
6. G. Adorni, S. Cagnoni, M. Mordonini, and M. Piaggio, "Coordination strategies for the goal-keeper of a RoboCup mid-size team", in *Proc. IEEE Intelligent Vehicles Symposium*. 2000, pp. 486–491, IEEE.
7. G. Adorni, S. Cagnoni, and M. Mordonini, "Landmark-based robot self-localization: a case study for the RoboCup goal-keeper", in *Proc. IEEE Int'l Conference on Information, Intelligence and Systems*. 1999, pp. 164–171, IEEE CS Press.
8. G. Adorni, S. Cagnoni, and M. Mordonini, "Purposive visual perception and co-operative behavior: some issues for the design of physical agents", in *Human and Machine Perception*: 2, V. Cantoni et al., Eds., pp. 107–123. Plenum Press, 1999.

RescueModel: A Multi-Agent Simulation of Bushfire Disaster Management

Gary Au¹, Simon Goss¹, Clint Heinze¹, Adrian R. Pearce²

¹ Air Operations Division

Defence Science and Technology Organisation
Melbourne, Australia

`firstname.lastname@dsto.defence.gov.au`

² Department of Software Engineering and Computer Science
The University of Melbourne, Melbourne, Australia
`pearce@cs.mu.oz.au`

Abstract. The RescueModel project is a vehicle for research into multi-agent systems, architectures, and strategies. It builds on the theoretical, practical, and experimental base of a decade of beliefs-desires-intentions (BDI) agent systems development. This paper describes a project that will bring together the environmental richness found usually in large scale military operations research simulations with the architectural richness of agent models often researched in universities. Proposed applications of RescueModel include search and rescue and disaster response studies.

1 Introduction

The scenario is large scale bushfires and fire fighting in the Australian environment. This is currently studied in human decision making and human-in-the-loop (H-I-L) simulation [8]. The RescueModel complements the RoboCup-Rescue simulator [7].

Several fire modelling systems exist. They include the Experimental Knowledge Systems Laboratory's Phoenix system [2] for Yellowstone National Park. La Trobe University's networked FireChief program has been used by Australia Defence Science and Technology Organisation's (DSTO's) Information Technology Division to study command and control issues [8]. CSIRO has developed a system called SiroFire [4], a physical model of fire perimeter spread.

Some aspects of each of these approaches are incorporated into the RescueModel framework [1]. For instance, CSIRO's SiroFire simulator is being integrated as a physical model. However, in SiroFire the cultural features like roads, tracks and railways, and certain natural features such as rivers and creeks are displayed but not used in calculating the fire spread. If researchers are interested in collaboration and commitment studies in the context of fire fighting tactics for instance, these features become centrally important. Equally important is three-dimensional visualisation from the point of view of each agent. Thus a polymorphic representation of agent friendly features is required. In other words, there is

a need for a representation containing polygon information for scene rendering, and also feature labels which agents can reason about.

The main drawback to many of the above fire approaches is that the representation of the environment is not accessible to agents in a structured way. In operational simulations of other domains, such as military operations, it has proved beneficial to clearly define activity in the simulation in terms of an agent model. Such agent systems have proven valuable for operational simulations in air combat [11].

The simulation infrastructure of RescueModel is a variation of a model developed by the Defence Science and Technology Organisation of Australia, named BattleModel. BattleModel was developed as a simulation of military operations, and used in the tender evaluation of a multi-billion dollar acquisition program. The simulation infrastructure takes care of legacy code integration issues, timing and simulation control, and data distribution. Whilst not restricted to agent models of human operators, BattleModel, and hence RescueModel, has been designed and constructed with agents in mind. It is a suitable vehicle for the exploration of agent architecture issues such as: access to the environment [1], recognition of intention, teams and team-work [3, 6, 10], and command and control [5].

A number of associated tools handle scenario specification, scenario control, and visualisation. The BDI agent model has been extended to meet the functional requirements of air combat modelling [11]. Primary amongst these is the necessity of developing agent models that are understood by the domain experts whose operational expertise they embody. RescueModel is currently being developed for the exploration of social issues in agent oriented systems. A design choice in the construction of RescueModel was to balance the requirements for semantic and environmental fidelity with computational tractability. Relative to existing computer science testbeds for evaluating agents, such as TileWorld [9], RescueModel is significantly richer. It is populated with adequate environmental complexity to study situated social semantics and engineering but has reduced complexity in the physical world models compared to those currently used by military simulations.

2 BattleModel

BattleModel implements a simulation framework for the integration of discrete heterogeneous time-stepped models. These models are “client-server” in the sense that they are both producers and consumers of data. BattleModel can be viewed as a set of data channels that are opened between models to facilitate data communication.

The BattleModel controls the execution thread of the system by “ticking” its models in a sequence that ensures that client-server data relationships are maintained. These models are time stepped and increment, dt , is not constrained and can be dynamically variable. Concepts involved in the operation of the model

include information servers, server registration, client registration, subscription, data setting and model wrapping. These key concepts are defined in Fig. 1.

Information Servers

Information servers provide the conduit through which data is passed between models. The Information Servers define and manage the external interfaces of a model. All data provided by one model to external elements leaves through an Information Server, all external data entering the model arrives through an Information Server. Registration Registration prepares the links to external interfaces used by the model. Models register to information servers, through their Model Wrapper, as the server or client of the data through an Information Server Manager.

Server Registration

A model that provides data through an external interface registers to the Information Server, through its Model Wrapper, as the server of the data. For example, an aerodynamic physical model determines the position (latitude, longitude, altitude) and orientation (heading, pitch, roll) of an aircraft. There is only one server for a given element of data. For example, only the aircraft platform physical model can serve the position data for the platform. The server of the data is not concerned with the destination of the data. The Information Server is responsible for distributing the data to models that have registered as clients of the data. Servers register as providing current or future data. Current data is relevant for the current simulation time step, future data is relevant for the following simulation time step. For example, the pilot model uses current data (own and target position, orientation, speed, etc) to determine what manoeuvre to execute in the following simulation time step.

Client Registration

A model that requires data through an external interface registers to the Information Server providing the data, through its Model Wrapper, as the client of the data. For example, an aerodynamic physical model requires a manoeuvre command (eg fly to a position) to update its state. There may be multiple clients for a given element of data. For example, the aircraft position is of interest to many models, including: radar; missile; RWR; ESM; etc.

Subscription

While registration provides the links to external data, subscription indicates when the external data is required. Only data sets which have been registered to as clients can be subscribed to. Each subscription is made with a filter that determines when the data subscribed to is of interest. In its simplest form the filter requires the data to be provided on all occasions. A more complex filter may be used, for example, by a radar model: If a radar model is range limited the filter can be defined to only provide target data when the range to a target is less than the range limit for the radar model. The filter determines the rate at which a model receives data to which it has subscribed. The filter reduces the communications bandwidth in the system as only the data that will be used by the model will be transmitted to the model.

Getting and Setting Data

Setting data to and getting data from is the process by which data enters the model and data exits the model respectively. All input/output to a model are passed through get/set operations defined by the model.

Model Wrapping

In practice there should be minimal impact on existing stand alone models as the models can be integrated by simply placing interface code before and after the models top level function call. Because of this integrating existing models into the Battle Model architecture is also known as wrapping of the models. The model wrapper is a class that encapsulates the stand-alone model and the model control, state and interface parameters. It defines the form in which models to be integrated into the Battle Model Architecture are to be delivered.

Fig. 1. The major concepts of the BattleModel illustrate its general client server nature)

To ensure that a model is stand-alone its interfaces are well defined. The Battle Model Architecture requires three operations to be defined for each model, namely: `init()` - for initialising the model in preparation for running the model; `compute()` - for running the model for one simulation time step; `close()` - for closing and cleaning up all model information. To support the above operations set operations are required to pass data into the model wrapper and get operations to obtain data from the model wrapper.

Multiple instances of models may be instantiated in a scenario. Where models are developed specifically for the Battle Model Architecture in C++ the model and model wrapper are typically on and the same. Existing C++ can be adapted to the wrapped form by defining the appropriate `init`, `compute`, `close`, `get` and `set` operations, and ensuring all interactions between the model and external environment occur through these interfaces. Where existing models are in a language other than C++ a wrapper is required to encapsulate the existing model. Mixed language programming techniques are used to interface the model to the model wrapper.

The model/architecture interface is illustrated in Fig. 3, that shows the relationship between Information Servers, the Model/Architecture Interface, the Model Wrapper and Model. The direction of the arrows in Fig. 3 show the directions in which function calls are made, as described in the caption. This design also allows simple development of test harnesses for stand-alone models.

3 Fire Disaster Scenarios and Models

Although the BattleModel framework has been used primarily for air combat studies in support of the Australian military, it is possible to use the framework for other purposes such as RoboCup-Rescue. The RescueModel project was developed with this in mind. It consists of the BattleModel framework, with newly developed models suitable for RoboCup-Rescue studies. The RescueModel framework is intended as a bushfire disaster complement to the official RoboCup-Rescue simulator, whose primary exploration area is earthquakes. The modularity of the RescueModel framework means that interchanging models will require less effort.

The scenarios currently under investigation involve fire fighting for the Australian environment. RescueModel is a ground breaking initiative because it is the very first one to combine detailed models of the environment with tactical procedures embodied in agent reasoning. Fire teams might consist of combined aircraft and ground units fighting widely separated fires in highly dynamic weather conditions and limited logistic support. There might also be civilians in need of rescuing. Some areas of investigation which might overlap with RoboCup-Rescue could be communications difficulties, appropriate distribution of limited resources, logistics, and risk management. There are also important issues in human decision making in these environments, whose reaction and decision time scales are on the order of air combat and modern war-fighting in general. Refer to Fig. 2) for details of the RescueModel components.

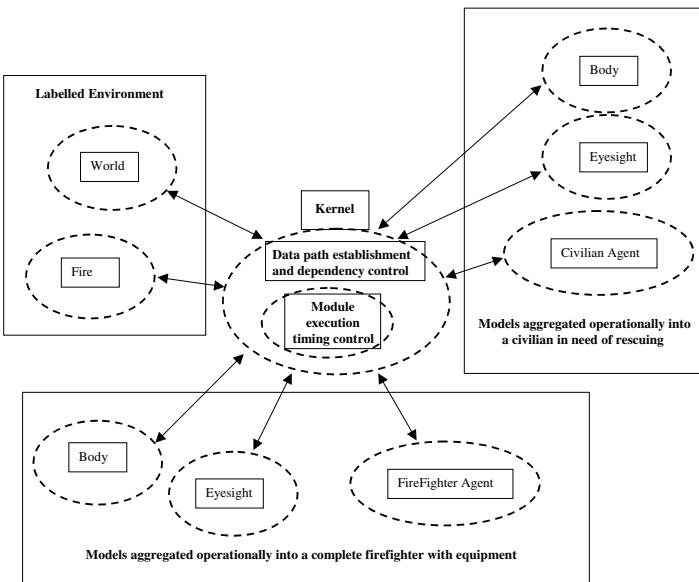


Fig. 2. Models are divided into the labelled environment (top left) and reasoning, sensor and platform models (right and bottom.)

Acknowledgements

The authors would like to thank the kind assistance of Michael Papasimeon, Gil Tidhar, Liz Sonnenberg, Leon Sterling and the BattleModel Design Team of DSTO.

References

1. G. Au and S. Goss. Rescuemodel: A simulation framework for the exploration of agent-environment and agent-agent interations in team situations. In *Proceedings of the fifth International SimTecT Conference, Sydney*, 2000.
2. P. R. Cohen, M. L. Greenberg, D. M. Hart, and A. E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48, 1989.
3. P. R. Cohen and H. J. Levesque. Teamwork. *Nous: Special Issue on Cognitive Science and Artificial Intelligence*, 25(4):473–512, 1991.
4. J. Coleman and A. Sullivan. The csiro bushfire spread simulator. In *Proceedings of the Institutie of Foresters of Australia 16th Biennial Conference, Canberra*, 1995.
5. H. Cottam and N. R. Shadbolt. Knowledge acquisition for search and rescue planning. *International Journal of Human-Computer Studies*, 48:449–73, 1998.
6. Francois F. Ingrand, Michael P. Georgeff, and Anand S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, pages 34–44, December 1992.

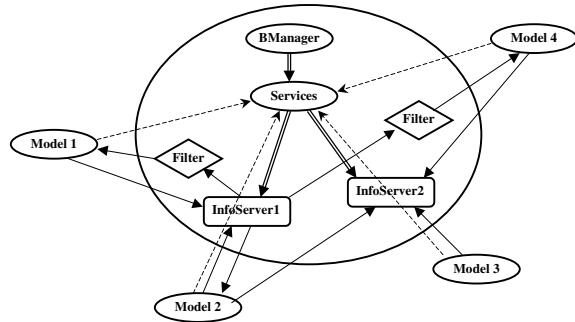


Fig. 3. RescueModel Architecture: A thread of execution showing interactions between the Information Servers, Model/Architecture Interface, and Model Wrapper is as follows. Step 1: When data managed by the Information Server satisfies the filter conditions specified by the subscribing model, the Information Server calls the appropriate set operation on the Model/Architecture Interface; Step 2: The Model/Architecture Interface converts the received data from the architectural form to the model form, and pushes the data into the model through a call to the appropriate set operation in the model; Step 3: Once the model has completed its compute cycle the Model/Architecture Interface obtains data from the model through a call to the appropriate get operation in the model. Step 4 The Model/Architecture Interface converts the data from the model form to the architectural form and pushes it to the information server through a call to the appropriate set operation in the information server. The architecture provides the: Model Architecture Interface Set operations (step 1), and Information Server Set operations (step 4). The models provide: Model Set operations (step 2), and Model Get operations (step 3).

7. H. S. Kitano, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada. Robocup-rescue: Search and rescue for large scale disasters as a domain for multi-agent research. In *Proceedings of the IEEE Conference on Man, Systems, and Cybernetics (SMC-99)*, 1999.
8. M. M. Omodei and A. J. Wearing. The fire chief microworld generating program: An illustration of computer-simulated microworlds as an experimental paradigm for studying complex decision-making behaviour. *Behaviour Research Methods, Instruments and Computers*, 27:303–316, 1995.
9. M. E. Pollack and M. Ringuette. Introducing the tileworld: Experimentally evaluating agent architecture. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI), Boston*, 1990.
10. M. Tambe, G. A. Kaminka, S. Marsella, I. Muslea, and T. Raines. Two fielded teams and two experts: A robocup response challenge from the trenches. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.
11. Gil Tidhar, Clinton Heinze, Simon Goss, Graeme Murray, Dino Appla, and Ian Lloyd. Using intelligent agents in military simulations or "using agents intelligently". In *Proceedings of the Eleventh Innovative Applications of Artificial Intelligence Conference, American Association of Artificial Intelligence (AAAI), Deployed Applications paper*, 1999.

Vision-based Localization in RoboCup Environments

Stefan Enderle¹, Marcus Ritter¹, Dieter Fox², Stefan Sablatnög¹,
Gerhard Kraetzschmar¹, Günther Palm¹

¹Dept. of Neural Information Processing
University of Ulm
D-89069 Ulm, Germany

²Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract. Knowing its position in an environment is an essential capability for any useful mobile robot. Monte-Carlo Localization (MCL) has become a popular framework for solving the self-localization problem in mobile robots. The known methods exploit sensor data obtained from laser range finders or sonar rings to estimate robot positions and are quite reliable and robust against noise. An open question is whether comparable localization performance can be achieved using only camera images, especially if the camera images are used both for localization and object recognition. In this paper, we discuss the problems arising from these characteristics and show experimentally that MCL nevertheless works very well under these conditions.

1 Introduction

In the recent past, Monte-Carlo localization has become a very popular framework for solving the self-localization problem in mobile robots [4, 5]. This method is very reliable and robust against noise, especially if the robots are equipped with laser range finders or sonar sensors. In some environments, however, for example in the popular RoboCup domain [6], providing a laser scanner for each robot is difficult or impossible and sonar data is extremely noisy due to the highly dynamic environment. Thus, enhancing the existing localization methods such that they can use other sensory channels, like uni- or omni-directional vision systems, is a state-of-the-art problem in RoboCup. In this work, we present a vision-based MCL approach using visual features which are extracted from the robot's unidirectional camera and matched to a known model of the RoboCup environment.

2 Monte Carlo Localization

Monte Carlo localization (MCL) [5] is an efficient implementation of the general Markov localization approach (see e.g. [4]). Here, the infinite probability distribution $Bel(l)$ expressing the robot's belief in being at location l is represented by a set of N samples $S = \{s_1, \dots, s_N\}$. Each sample $s_i = \langle l_i, p_i \rangle$ consists of a

robot *location* l_i and *weight* p_i . As the weights are interpreted as probabilities, we assume $\sum_{i=1}^N p_i = 1$.

The algorithm for Monte Carlo localization is adopted from the general Markov localization framework. Initially, a set of samples reflecting initial knowledge about the robot's position is generated. During robot operation, the following two kinds of update steps are iteratively executed:

Sample Projection across Robot Motion: As in the general Markov algorithm, a motion model $P(l|l', m)$ is used to update the probability distribution $Bel(l)$. In MCL, a new sample set S is generated from a previous set S' by applying the motion model as follows: For each sample $\langle l', p' \rangle \in S'$ a new sample $\langle l, p' \rangle$ is added to S , where l is randomly drawn from the density $P(l|l', m)$.

Observation Update and Weighted Resampling: Sensor inputs are used to update the robot's beliefs about its position. All samples are re-weighted by incorporating the sensor data o and applying the observation model $P(o|l')$. Given a sample $\langle l', p' \rangle$, the new weight p for this sample is given by

$$p = \alpha P(o|l') p' \quad (1)$$

where α is a normalization factor which ensures that all beliefs sum up to 1. These new weights for the samples in S' provide a probability distribution, which is then used to construct a new sample set S . This is done by randomly drawing samples from S' using the distribution given by the weights.

3 Vision-Based Localization

In RoboCup where laser range finders are often not available and sonar data are too unreliable due to the highly dynamic environment, a natural candidate is the visual channel, because many robots include cameras as standard equipment. An example for using visual information for MCL has been provided by Dellaert et al. [1].

An interesting and open question is whether the MCL approach still works when the number of observations is significantly reduced and when particular observations can be made only intermittently. In the following, we show how to adapt the MCL approach in order to overcome these problems.

3.1 Feature-Based Modeling

As described in Eq. 1, the sensor update mechanism needs a sensor model $P(o|l)$ which describes how probable a sensor reading o is at a given robot location l . This probability is often computed by estimating the sensor reading \tilde{o} at location l and determine some distance $dist(o, \tilde{o})$ between the given measurement o and the estimation \tilde{o} .

As it is not possible to efficiently estimate complete camera images and then compute image distances for hundreds of samples, we use a feature-based approach. After evaluating various feature detectors on our robots (see [3]), we decided to use the following features: 1) goal posts of blue and yellow goal, 2) corners, and 3) distances to field edges.

Feature Detection: In a first step the camera image is segmented in order to simplify the feature detection process. Based on the segmented image, we use different kinds of filters detecting a respective color discontinuity.

The *goal post detector* detects a vertical *white-blue* or a *white-yellow* transition for the blue or yellow goal post, respectively (see left image in Figure 1). The *corner detector* searches for vertical *green-white-green* transitions in the image (middle image in Figure 1). The *distance estimator* estimates the distance to the field edges based on detected horizontal *green-anything* transitions in the image. At the moment, we select four specific columns in the image for detecting the field edges (right image in Figure 1).



Fig. 1. Detection of post, corner and edge features.

Weight Update Let the sensor data o be a vector of n features $f_1 \dots f_n$. If we assume that the detection of features depends solely on the robot's position and does not depend on the detectability of other features, then the features are independent and we can conclude:

$$\begin{aligned} P(o|l') &= P(f_1 \dots f_n | l') \\ &= P(f_1 | l') \dots P(f_n | l') \end{aligned} \quad (2)$$

The sensor model $P(f_i | l)$ describes how likely it is to detect a particular feature f_i given a robot location l . In our implementation, this sensor model is computed by comparing the horizontal position of the detected feature with an estimated position as determined by a geometrical world model. The distance between the feature positions is mapped to a probability estimate by applying a heuristic function as illustrated in Figure 2.

The application of these heuristics to the examples used in Figure 1 are illustrated in Figure 3. Probabilistic combination of evidence for several features

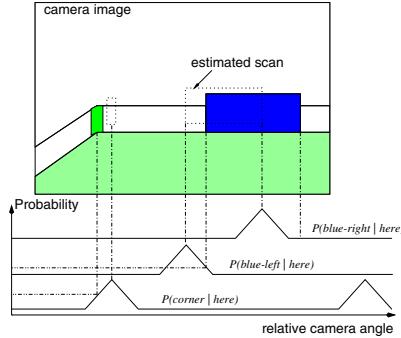


Fig. 2. Heuristics for estimating the probabilities $p(f_i | l)$ from the current camera view.

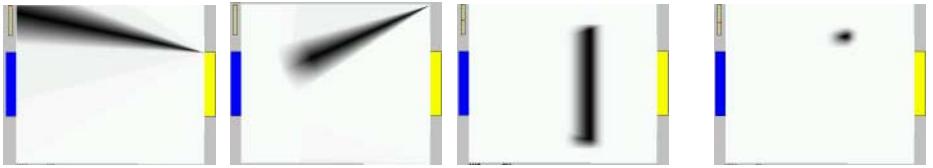


Fig. 3. Probability distributions $P(f_i | l)$ for all field positions l (with fixed orientation) given the detection of a particular feature f_i or all features, respectively. From left to right: goal posts, corners, distance measurements, all three combined.

yields significantly better results, as convincingly demonstrated by the rightmost image in Figure 3.

The figure illustrates various properties of our approach. The shape of the function causes all samples with comparatively high angular error to be drastically down-valued and successively being sorted out. Thus, detecting a single goal post will reshape the distribution of the sample set such that mostly locations that make it likely to see a goal post in a certain direction will survive in the sample set. Secondly, there is only a single heuristic function that captures all of the ambiguous corner features. Each actually detected corner is successively given a probability estimate. If the corner detector misses corners that we expected to see, this does not do any harm. If the detector returns more corners than actually expected, the behavior depends on the particular situation: detecting an extra corner close to where we actually expected one, has a small negative influence on the weight of this sample, while detecting a corner where none was expected at all has a much stronger negative effect.

4 Experiments

The described method was implemented and evaluated on a Sparrow-99 robot, a custom-built soccer platform equipped with an uni-directional camera [2].

Experiment 1: Number of Features In this experiment, we show that the robot can localize robustly and that the accuracy can be improved by adding more features. A Sparrow-99 robot was placed in one corner of the field (the right top circle of Figure 4). In order to have an accurate reference path, we moved the robot by hand along a rectangular trajectory indicated by the dots.

The first image in Figure 4 displays the odometry data when moving four rounds and shows the drift error that occurs. The second image displays the corrected trajectory which does not drift away. The third image displays the first round corrected by the localization algorithm using only the goal posts as features. In the fourth image we can see a more accurate path found using all three feature types.

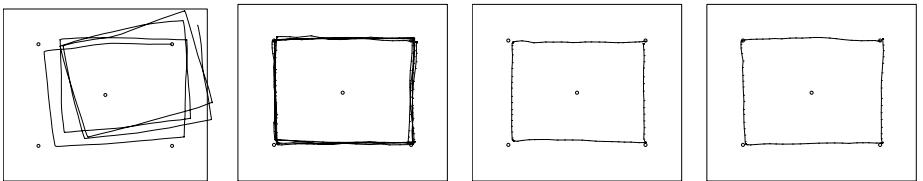


Fig. 4. A single round of the Sparrow-99 being pushed along a rectangular path in the RoboCup field.(Odometric path, corrected path using goal post features, corrected path using all features).

Experiment 2: Number of Samples Implementing sample-based localization, it is important to have an idea of how many samples you need. Obviously, a small number of samples is preferred, since the computational effort increases with the number of samples. On the other side, an appropriate number of samples is needed in order to achieve the desired accuracy. In this experiment, we moved four rounds exactly as in the previous experiment. This time, we used five different numbers of samples: 50, 100, 150, 1000, 5000

In the left image of Figure 5, the average localization errors of the different sample numbers are shown. One can see that the error decreases when more samples are used. On the other hand, the difference in accuracy does not increase very much from 150 samples over 1000 to 5000. The right image of Figure 5 shows both the average error and the maximum error of the same five runs (50, 100, 150, 1000 and 5000 samples). Again, you can see that above 150 samples, the accuracy hardly increases.

5 Conclusions

In this paper, we used the Monte-Carlo approach for vision-based localization of a soccer robot on the RoboCup soccer field. Unlike many previous applications,

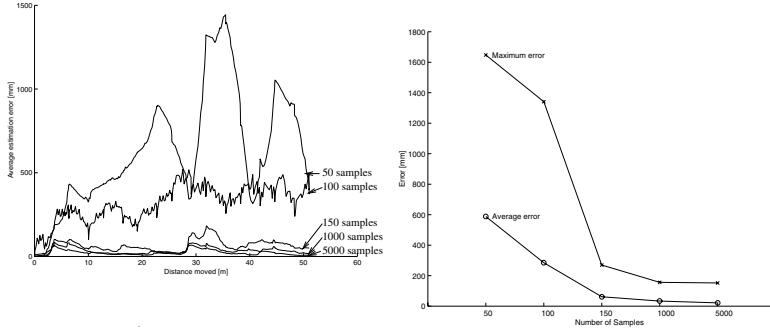


Fig. 5. Left: Average localization errors of the different sample numbers. Right: Average error and maximum error of the same five runs (50,100,150,1000 and 5000 samples used).

the robot could not use distance sensors like laser scanners or sonars. Also, special camera setups were not available. Instead, the onboard camera was used for localization purposes in addition to object recognition tasks. As a consequence, sensor input to update the robot's belief about its position was low-dimensional and sporadic. Nevertheless, the experimental evaluation demonstrated that Monte Carlo localization works well even under these restrictive conditions.

We could also show that even with a small number of detected features leading to sporadic observation updates, the localization results are usable. However, by increasing the number of visual features the accuracy enhances dramatically.

References

- [1] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999.
- [2] Stefan Enderle. The sparrow 99 robot. Technical report, University of Ulm, 1999. internal report.
- [3] Stefan Enderle, Marcus Ritter, Dieter Fox, Stefan Sablatnög, Gerhard Kraetzschmar, and Günther Palm. Soccer-robot localization using sporadic visual features. *Proceedings of the IAS-6 International Conference on Intelligent Autonomous Systems*, 2000.
- [4] Dieter Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, University of Bonn, Bonn, Germany, December 1998.
- [5] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *In Proc. of the National Conference on Artificial Intelligence*. AAAI, 1999.
- [6] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup a challenge problem for ai. *AI magazine*, 18(1):73–85, 1997.

Collaborative Emergent Actions between Real Soccer Robots

M. Ferraresto, C. Ferrari, , E. Pagello (+),
R. Polesel, R. Rosati, A. Speranzon, W. Zanette

Intelligent Autonomous Systems Laboratory
Dept. of Electronics and Informatics, The University of Padua, Italy
(+) also with LADSEB-CNR, Padua, Italy
epv@dei.unipd.it

Abstract We discuss how to induce a set of collaborative emergent actions between two soccer robots. Cooperative abilities, like exchanging a ball, can be achieved through the use of efficient collision avoidance algorithms implemented on two players able to frequently swap their roles. These algorithms have been tested on Bart and Homer, designed at IAS Lab. of Padua Univ., that played quarter, semifinals, and finals with ART at RoboCup'99. The interaction with the ball was made easy by a directional kicker which allowed to hit the ball both frontally and laterally.

1. A Directional Kicker.

An *intelligent multi-robot system* can be constructed by some mobile robots that cooperate to solve a given complex task thanks to *communication* among individuals, and a dynamic *group reconfigurability*. If we indicate with the term *cooperative behavior* a collective behavior that is characterized by cooperation [4], then a behavior is *emergent* if it can only be defined using descriptive categories which are not necessary to describe the behavior of the constituent components [12]. In [5] a cooperative ability without communication is obtained through the use of a BDI approach, while, in [14] the same ability is obtained by using explicit communication. In [8], and [9], we have illustrated how to induce some emergent cooperative behavior through an implicit communication. In this paper, we discuss the problem of how to give cooperative abilities to the individual robots in the group through the use of emergent behaviors, by illustrating the solutions that has been adopted to control the two robots *Bart* and *Homer*, designed at IAS Lab. of Padua University, that played very successfully in the ART Team on RoboCup'99, by scoring a total of 5 goals in 9 games. Homer won also the Technical Challenge against Friburgh University.

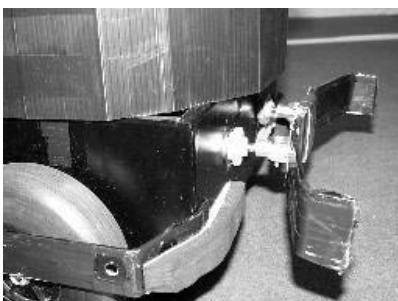


Fig 1

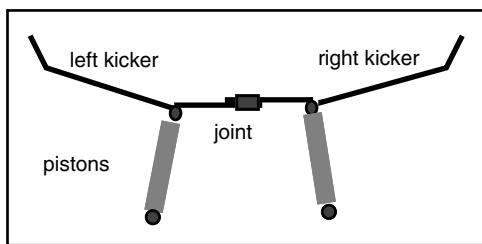


Fig 2

In these games, a very effective emergent cooperative ability was achieved by Bart and Homer through the use of efficient *collision avoidance algorithms* activated while they were mutually exchanging their play roles. Basic behaviors, like *find_ball*, *go_to_ball*, and *carry-ball*, were used in conjunction with a smart *collision avoidance* ability, that was the reason why the collective emergent were induced into the two robots. In particular, the *carry_ball* behavior was able to carry the ball along a path hitting it only when a full contact between the left and/or right kicker was detected. The most effective tools which made easier to control the ball, during robot actions, was the *directional kicker*, the only one among all competing teams. The kicking system, Fig. 1., is composed by an air tank, a couple of pneumatic pistons, and a directional effector, which allows to kick frontally or laterally, depending on the activated piston (both for a front kick, or left or right, only for a lateral kick). The effector is composed by two separated symmetric pieces with a small corner to facilitate the control of the ball. The two pieces may slide one on each other allowing high flexibility and accuracy of contact with the ball, Fig. 2. The accurate ball control allowed to carry the ball without loosing it, along the direction detected by the collision-avoidance algorithms.

2. Function Q and the Dynamic Role Assignment of ART.

Bart and Homer were programmed by using a *behavior-based* approach, [3], to the aim of obtaining a robotics social organization, [10]. A set of different robot roles has been introduced, according the definition of a role given in [13], by specifying a set of behaviors. An Arbiter activates the basic behaviors according to the data received from the sensor module, based on a simple F.S.M. Each basic behavior is realized as an Expert in the Real-time Kernel Ethnos [11].

A *measure of quality* Q_i , able to triggers the prop role, was introduced at IAS Lab. of Padua Univ., for evaluating how much work must be done by a robot to get the ball in the best position to score. Q_i is a weighed linear function of:

- distance of robot i to ball
- relative position of robot i w.r.t. a right approaching configuration to the ball
- ball recorded position when the ball is not visible by robot i
- presence of other robots on the way of robot i to goal
- number of failure of robot i to get moving around without colliding
- previous role played by robot i .

Each robot i computes independently Q_i based on its local estimation. The robot sends this value of Q_i to its teammates 10 times per second, and decides autonomously how to behave comparing its own estimation of Q_i with the other value of Q_i . Function Q was used in Stockholm by the whole ART Team [6], to distribute among its team members a specific role set depending from its value communicated using Ethnos. Players were able to assume *three different roles*: - *Role #1* must play the ball, either if it is defending or it is attacking; *Role #2* must cooperate with robot that has role #1, *Role #3* must locate itself far from the place where the ball is played.

A generalization of the function Q has been introduced in [7], as a set of *utility functions*, able to give through an explicit communication some *utility values* that indicate the usefulness of each role. Thus, ART robots may decide to distribute among its team members a specific role set depending from the values assumed by some functions transmitted among the members.

Bart and Homer were able to show a cooperative action, like a *ball exchange*, by coordinating their basic behaviors through the dynamic assignment of the above three roles realized by a set of behaviors that exploit some smart collision-free motion strategies, based on the computation of field vectors. An Obstacle Avoidance module implemented these motion planning algorithms as an Ethnos Expert.

3. Field vector-based collision-avoidance.

Indeed, each ART member was left free to specialize the basic behaviors for the robots of their local team. At IAS Lab of Padua Univ., we developed an original *design of the behaviors* for Bart and Homer, based on collision-avoidance algorithms that use *field vectors* and generate *schema-based behaviors* [2]. Both *target* (the attractor) and *obstacles* (the repulsors) generate their specific vectors. The target generates a purely attractive field, proportional to the distance, while the obstacles generate a rotational field

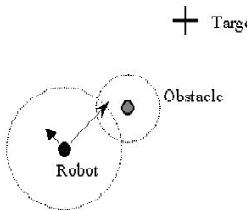


Fig. 3

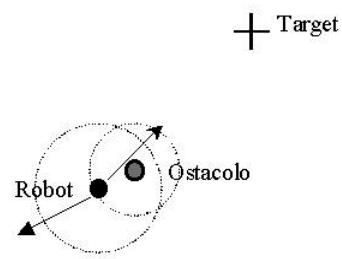


Fig. 4

All active behaviors send to the Obstacle Avoidance Expert (OAET) the coordinates of a target. The attractive field vector is bounded by the max. robot velocity. Then, OAET realizes the motion towards the target according to the corrections induced by the obstacles on the attractive field vectors. Each obstacle is surrounded by a circular zone that indicates its affected area (OA), equal to 1 meter. The robot-affected area (RA) is computed according to a sum of robot dimension with its instant velocity: OA_ray = 1000 mm; RA_ray = V-robot + Dim-robot (see Fig. 3). The rotational field is computed according to Coulomb Laws: $I = (K * H) / r^2$ where K and H are constants. K was determined experimentally in such a way that its value could be comparable with the intensity of the attractive field, while H is equal to 0, to 1/Distance-from-object, or to 1, w.r.t. to the Distance-from-obstacle. That is, if the robot is too much close to an obstacle, then it must be forced to make a back-step, as in Fig. 4, since the games rules penalizes a robot if it collides with other robots. Then, obstacle's field vector become repulsive instead of rotational, if the distance between robot and obstacle is ≤ 550 mm. Sometimes two alternative paths can be equally selected to get the target, as in Fig 5, but only one is safe due to other constraints, as in Fig 6. A decision is achieved by blocking a direction on one rotational vector, in order to force all other vector directions to follow. A frequent delicate case is in Fig. 6, where the robot must decide if it is more convenient to turn around the obstacle. Because of the wall, it may turn right towards the free area. But, if walls is considered as a particular kind of repulsive obstacles, then the resultant becomes parallel to the wall itself, as in Fig 7.

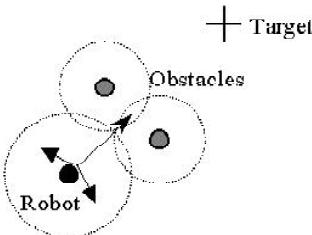


Fig. 5

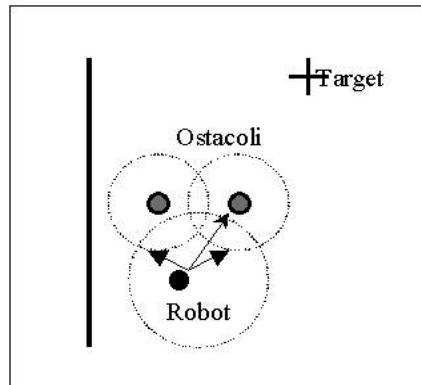


Fig. 6

A serious problem may arise if both two opposite directions are blocked due to some difficult configuration like the one of Fig. 8 (and its symmetric configuration where the target and the robot are swapped). We decided the robot does not move for a while, waiting the opponents' move, or a game restart by the arbiter.

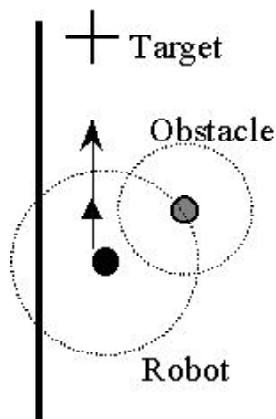


Fig. 7

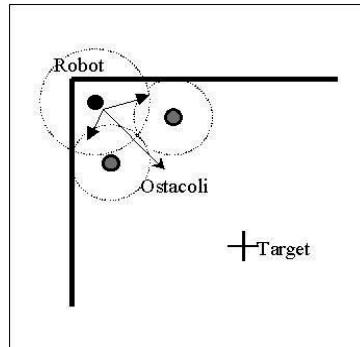


Fig. 8

To enhance role swapping robustness, and avoid the system instability, Q was made sensitive to the previous role played by robot i . To make the role #2 very effective, at IAS Lab, we did the following assumptions: - *A*): when a robot plays role # 2, it must never obstruct robot with role #1; - *B*): when a robot plays role # 2, it must try to quickly take the ball if it realizes that robot with role #1 does not succeed to perform its task; - *C*): when a robot plays role # 2, it must keep itself close to robot with role #1 to recover the ball if robot with role #1 loses it; - *D*: when a robot plays role # 2, it must never interfere on the path between robot with role #1 and opponent's goal. To reinforce the robustness of *robot role-swapping*, robot navigation algorithms were based on a different evaluation of the attractive and repulsive potential fields of robot that plays role #2 versus the one of robot that plays role #1. Robot #2 is much more influenced by obstacles than robot #1. Then, robot #2 also moves directly towards the ball, but it does not obstruct robot #1, since it is affected by a stronger repulsive force than robot #1. Indeed, if robot #2 meets robot #1 along its path to the ball, he treats robot #1 as an obstacle, and its repulsive force prevents it to

interfere with the action of #1. If, for any reason, robot #2 does not meet robot #1 along its path to the ball, then robot #2 is able to assume role #1. When robot #1 meets an opponent, while keeping the ball, often it makes a back-step, to avoid collision, while robot #2, its supporter, succeeds to move to a better approaching position to the ball. Thus, robot #1 makes room to its supporter robot #2, that may take the ball, and swap its role.



Fig. 9

In Fig 9 and 10, robot #1 keeps the ball, but robot #2 is in a better scoring position, although far from the ball, the value of Q computed by robot #1 becomes lower than the one computed by robot #2. Thus, the two robots swap their roles, and they succeed to *exchange the ball*, as an emergent behavior. This configuration really happened several times when Bart and Homer had the opportunity to play together with ART team. We verified it (*and video recorded in www.dei.unipd.it/~robocup*) several times in Stockholm during the quarter-finals and semi-finals against the Universities of Ulm and Friburg .

4. Conclusions

We illustrated our approach based on the emergent behaviors for controlling Bart and Homer that played together successfully at RoboCup'99 competition with ART team. To easily control the ball, a special kicker was designed for Bart and Homer, which allows to kick frontally or laterally. A set of basic skills, like `carry_ball`, and others, were implemented. We showed how to achieve an emergent cooperative ability, and to realize a ball exchange, through the use of efficient collision avoidance and robot role swapping.

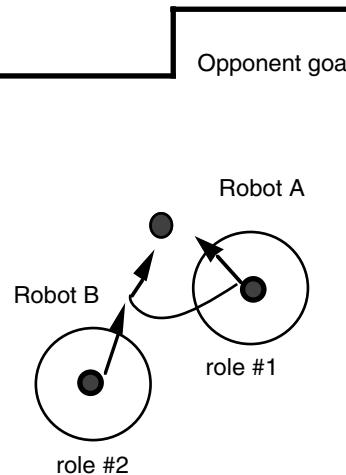


Fig. 10

Acknowledgements

This research was supported partially by MURST (Certamen Project), by CNR (Special Project), and by the ENEA (Parallel Computing Project). We acknowledge P. De Pascalis, P. Filippini, M. Lorenzetti, A. Modolo, P. Patuelli, N. Scattolin, A. Vaglio, and in particular Matteo Peluso, all students at the Eng. School of Padua Univ., who contributed to the researches illustrated in this paper. We like also to acknowledge all the members of ART, in cooperation with which we have developed and tested the dynamic role assignment, in particular G. Adorni, A. Bonarini, D. Nardi, and M. Piaggio.

References

- [1] Arai T., Ota J.: Let-us Work Together - Task Planning of Multiple Mobile Robots. IROS'95, pp. 298-303, Pittsburgh, Aug.1995
- [2] Arkin, R.C.: Behavior-Based Robotics. The MIT Press, 1998
- [3] Brooks, R.: A robust layered control system for a mobile robot. IEEE Jour. of Robotics and Automation. Vol. RA-2, No. 1, pp. 14-23
- [4] Cao, Y.U.: Cooperative Mobile Robotics: Antecedents and Directions. Autonomous Robots, Special Issues on Robot Colonies, R.C. Arkin and G.A. Bekey Eds., Vol 4, No. 1, March 1997
- [5] Hannebauer, M., Wendler, J., Gugenberger, P., Burkhard, H.D.: Emergent cooperation in a virtual soccer environment. in DARS 3, T. Lueth, et al. Eds., Springer 1998, pp. 341-350
- [6] Nardi, D., Adorni, G., Bonarini, A., Chella, A., Clemente, G., Pagello, E., and Piaggio, M.: ART99 Azzurra Robot Team. in RoboCup-99: Robot Soccer World Cup III. M. Veloso et al. Eds., L. N. on A. I., Springer 2000, pp. 695-698.
- [7] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, A. Sgorbissa: Communication and coordination among heterogeneous Mid-size players: ART99. This book.
- [8] Pagello, E., D'Angelo, A., Montesello, F., Garelli, F., Ferrari, C.: Cooperative behaviors in multi-robot systems through implicit communication. Robotics and Autonomous Systems, Vol. 29, No. 1, pp. 65-77, 1999
- [9] Pagello, E., Ferrari, C., D'Angelo, A., Montesello: Intelligent Multirobot Systems Performing Cooperative Tasks. Invited paper of Special Session on "Emergent Systems - Challenge for New System Paradigm". IEEE/SMC, Tokyo, Oct. 12-15, 1999, pp. IV-754/IV-760
- [10] Parker, L.E.: ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. IROS'94, Munich 1994, pp. 776-783
- [11] Piaggio, M., Sgorbissa, A., Zaccaria, R. : ETHNOS: a light architecture for real-time mobile robotics. IROS'99, Kyonju (Korea), Oct. 1999
- [12]. Steels, L.: The artificial life roots of artificial intelligence. A. I. Lab., Vrije Universiteit Brussel, Nov. 1993
- [13] Stone, P. Veloso, M.: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence, 110(2), 1999, pp. 241-273
- [14] Yokota, K., Ozaki, K., Matsumoto, A., Kawabata, K. Kaetsu, H., Asama, H.: Modeling environment and tasks for cooperative team play. in DARS 3, T. Lueth et. al. eds., Springer 1998, pp. 361-370

The Statistics Proxy Server

Ian Frank¹, Kumiko Tanaka-Ishii², Katsuto Arai³, and Hitoshi Matsubara⁴

¹ Complex Games Lab, ETL, Tsukuba, Japan 305, ianf@etl.go.jp

² Tokyo University, Japan, kumiko@ipl.t.u-tokyo.ac.jp

³ Chiba University, Japan, karai@cogsci.l.chiba-u.ac.jp

⁴ Future University, Hakodate, Japan, matsubar@fun.ac.jp

Abstract. We present a real-time statistical analysis tool for soccer. This system is designed to promote the advancement of RoboCup by facilitating fundamental research on issues such as learning and team evaluation and assessment. Analysis of a game is carried out by a central server, to which clients can connect to request data. We describe the operation of the system and give examples of its potential applications.

1 Introduction

The fundamental goal of RoboCup is scientific research. In this paper, we present a tool that promotes this goal by providing real-time, in-depth statistics on soccer games. We expect this tool to have many uses, including the evaluation of team modules and the design of automated coaches. At the simulator league of RoboCup 2000 we successfully demonstrated a real-time, web-based interface for visualising the statistics produced by our system. We have released all of our code as open source to provide maximum accessibility and functionality to all users (for downloads, see <http://www.etl.go.jp/etl/suiron/~ianf/Mike/>).

The original motivation for this work was our research on the automatic commentary system MIKE [1, 2]. In commentaries, high-level analysis of a game is clearly important for making game summaries and for identifying the noteworthy aspects of the game. At the same time, however, we realised that high-level information on a game would also be indispensable to other researchers. We therefore split out all of MIKE's statistical analysis code and used it as the core for a separate analysis program. We chose the server-client model for this program because it provides the maximum benefit for the smallest possible system load. Since it is designed to run in tandem with the RoboCup's Soccer Server, we call our system the Statistics Proxy Server (or sometimes also 'pre-MIKE', in its role as a pre-processor for MIKE).

The Statistics Proxy Server analyses a game and makes statistics available in real-time. Any number of client modules can then attach to the server and request some subset of this data at configurable intervals. We describe both the server and the client protocol below, giving examples of their use (§2). We anticipate that real-time access to such statistics will find many uses within RoboCup; we mentioned above our own work on soccer commentary, and we have also previously used the statistics to demonstrate the progress of the RoboCup simulation

league to date [3, 4]. In §3, we describe how we have now used the Proxy Server to generate a database of statistics on previous RoboCup games, and how the current Proxy Server utilises this database to increase the options available to clients. Finally, in §4 we discuss the possibilities for further developments, before giving our conclusions in §5.

2 System Details

The relationship between the Statistics Proxy Server and the core code of MIKE is shown in Figure 1.

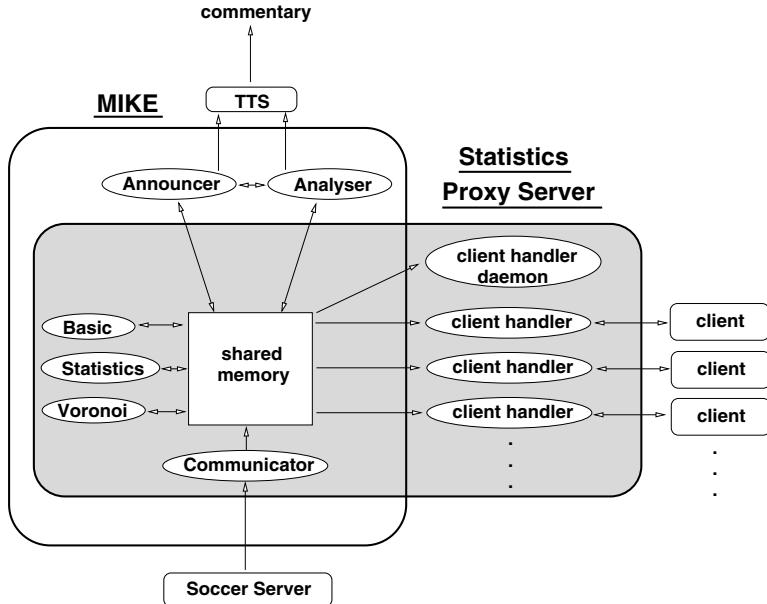


Fig. 1. The Statistics Proxy Server and its relationship to the core code of MIKE

The *client handler daemon* in this figure creates separate *client handler* threads to communicate with each client that connects to the Statistics Proxy Server. Thus, the server is extremely flexible: it can provide different data on different time schedules to many different clients. The most convenient way for a client to make its connection is in *configuration script mode* (there is also a *command-line mode*, which we describe in full detail in the manual bundled with the code). The configuration script specified by the client on the command-line must contain a series of lines each starting with one *command* and followed by a number of *flags*. Each *command* in the configuration script corresponds to a particular statistic. In practice, these commands are divided into three groups.

- **Team Statistics.** This first, and largest, group of commands relates to statistics about entire teams. Examples of team-based commands are `TScore team`, and `TXAvr team`, which request statistics on the score and average X position of the players of the specified *team*, respectively. The variable *team* must be either `left` or `right`, but an easy method for simultaneously requesting statistics on both teams is also implemented: *team* can be set to either `'*'` or `'_'`.
- **Player Statistics.** The second group of commands relates to statistics on individual players. Examples of player-based commands are `P1ShootNum player`, and `P1Distance player`, which request statistics on the number of shots made by *player*, and the distance that *player* has covered, respectively. Here, the variable *player* identifies players via the numbers displayed by the Soccer Server monitor. Again replacing the variable with `'*'` or `'_'` defaults to requesting information on all possible players. To make it easier to simultaneously obtain the same statistics for large numbers of players (*e.g.*, all those on one team), the specification of ranges for the *player* variable is supported. For example, `P1Distance 1-11` will give the distances covered by all the players on one team.
- **Other Statistics.** The final group of commands covers extra statistics that concern the interactions between two individual players. Examples are `Bigram player1 player2`, and `CorXX player1 player2`. Here, *player1* and *player2* are once more the player numbers as displayed by the Soccer Server monitor, and the `'*'`, `'_'` or range notation are again available. Bigrams and correlations, together with the complete list of all the other statistics, are described in the Statistics Proxy Server manual.

In total, there are 33 team statistics, 14 player statistics and 5 other statistics. There is also one extra command, `all`, that requests the entire set of current statistics to be sent (and also the commands `all_team` and `all_player` or their equivalents `all team` and `all player` for requesting the complete set of just the team statistics or just the player statistics). In the configuration script, each `command` must be accompanied by at least one `flag`. These flags allow clients to control the way that statistics are sent by the server, as follows:

1. `T num` — a mandatory flag for every command in the configuration file. It specifies the time interval (*num*, in Soccer Server game steps) at which the statistic should be sent to the client.
2. `L num` — specifies a lower bound. The statistic is only sent to the client (at the time determined by the `T` flag) if it is greater than or equal to *num*.
3. `U num` — specifies an upper bound. The statistic is only sent to the client (at the time determined by the `T` flag) if it is less than or equal to *num*.
4. `C num` — specifies a minimum change. The statistic is only sent to the client (at the time determined by the `T` flag) if it has changed by an amount greater than *num* since it was last reported (or if it has never been reported).

These flags allow for the construction of sophisticated reporting regimes. For instance, the following command will produce statistics every 10 time steps on all the left team players except the goalkeeper (player 1) with 5 or more shots on goal: `PLShootNum 2-11 T10 L5`.

In Figure 2 we give a simple example of a complete configuration script. Commented lines start with a hash (#) and blank lines are allowed. When a client successfully connects to the server using such a script, it receives the requested data for the duration of the connection whilst play is still in progress. The format for sending information to the clients is as follows:

$$\langle \text{command} \rangle \langle \text{variables} \rangle \langle \text{value} \rangle \langle \text{time} \rangle$$

where `command` is simply an echo of the command that requested the data (just the label identifying the statistic itself), `variables` identifies the subject (a team for team statistics, a player for player statistics, or a pair of players for other statistics), `value` is the result computed by the Statistics Proxy Server, and `time` is the current game time.

```
# Comment: Simple proxy configuration script
# Requests average X and Y location of the left team's players
# every 5 game steps. Also requests, every 10 game steps, the number
# of goals scored by any goalscorer (player with one or more goals).

TXAvr left T5
TYAvr left T5
P1Score * T10 L1
```

Fig. 2. A simple script for a client connecting in configuration script mode

3 The Data Centre

We have already used pre-release versions of the Statistics Proxy Server to analyse previous RoboCup tournaments [3, 4]. The Server's analysis allowed us to demonstrate how teamwork has improved significantly along the long side of the field but not along the shorter, how teams carry out more marking, and how the link between ball-handling skills and performance has strengthened.

In addition to carrying out such analysis, however, we can also now make data on past games available in real-time through the Proxy Server. To do this, we analysed the logs of all the 243 completed games from the first three years of RoboCup simulator contests (a database of over 500Mbytes). For each game, we recorded the values of each of the Proxy Server statistics. Further, we created two extra sets of data by collecting separately the data for the winners and losers of each game. (Actually, since team designers are allowed to manually change their team settings and code at half-time, we built these extra databases by

looking at the winners and losers of each *half-game*, excluding the 97 half-games that were drawn). The Proxy Server makes the results of this analysis available via the following extra flags:

- **S yr-percent** — specifies a significance level for reporting a statistic from the current game. The value of *percent* must be greater than 0 and less than 100. If *percent* is greater than or equal to 50, the statistic is only reported (at the time determined by the T flag) if its value in the current game ranks in the top *percent*% of stored values for year *yr* in the data centre. For values less than 50, the statistic is only reported if its value in the current game ranks in the bottom *percent*% of stored values.
- **A yr-percent** — specifies a significance level for reporting a statistic. Same as the S flag except that the value of the statistic must be significant amongst just *winning* teams.
- **B yr-percent** — specifies a significance level for reporting a statistic. Same as the S flag except that the value of the statistic must be significant amongst just *losing* teams.
- **Y yr** — requests data on a statistic directly from the data centre for year *yr* (a four digit number) rather than on the current game. The average of the values in the data centre is returned. Overrides all other flags, which are ignored if Y is specified.

These flags greatly increase the power of the system. For instance, a script containing just the commands ‘all S1999-5 T5’ and ‘all S1999-95 T5’ will allow a client to receive (every 5 game steps) just the statistics on the current game that are in the top or bottom 5% when compared to the teams in the previous year’s RoboCup. Future versions of the Statistics Proxy Server will also include a tool for creating extra data sets from arbitrary collections of log files. This will enable developers to easily compare the performance of new updates of their programs against established benchmark versions.

4 Developments

The uses of the Statistics Proxy Server are limited only by the imagination of RoboCup developers. Already, we have demonstrated at RoboCup 2000 a web-based interface for visualising the system’s statistics using graphs, charts and pitch diagrams. Note that machine vision systems for creating textual logs of real robot games will also soon allow the Statistics Proxy to be used with other leagues. We have room for just a small list of other possible developments:

- Add a module that gives information on how various statistics change *during* a game.
- Use the Statistics Proxy as the basis for a strong on-line coach client.
- Use the statistics as a testing tool to identify more precisely the strengths and weaknesses of individual teams. At the current stage of RoboCup, winning is taken to identify the teams that employ good techniques. However, the

reverse is not true: teams that lose do *not* necessarily employ bad techniques. The ability to identify strong aspects of teams whose overall performance does not compare to the champions will help prevent good research ideas being overlooked. In our most recent work, we have used the Statistics Proxy to analyse the log files of the RoboCup evaluation sessions, showing that there is a negative correlation between team performance and robustness.

- Develop a testing tool that can play two teams against each other to test a given, concrete hypothesis (*e.g.*, team A is stronger than team B, team A passes more accurately than team B) at a pre-determined confidence level.

Eventually, we hope the Statistics Proxy Server will form the core of an advanced RoboCup expert analysis module incorporating techniques developed by many RoboCup researchers. A good starting point for this would be the work done by the developers of ISAAC [5], who have used C4.5 to build decision trees that describe how teams behave in a game. Since it is hard to build such trees incrementally during a game, ISAAC is typically used as a post-game learning module.

5 Conclusions

We have described the Statistics Proxy Server we have developed for RoboCup. We gave details of the implementation and examples of the use of the system. We discussed how the server has already been used in projects such as MIKE and to assess the progress of RoboCup to date. We also showed that it is possible to incorporate data from past RoboCups and listed some of the promising potential applications of the Statistics Proxy Server system.

It is our hope that the Statistics Proxy Server will be a significant and useful tool in the development of RoboCup. To maximise the chances of this actually happening, we have released our source code into the public domain.

References

1. K. Tanaka-Ishii, I. Noda, I. Frank, H. Nakashima, K. Hasida, and H. Matsubara. MIKE: An automatic commentary system for soccer. In *Proceedings of ICMAS-98*, pages 285–292, 1998.
2. H. Matsubara, I. Frank, K. Tanaka-Ishii, I. Noda, H. Nakashima, and K. Hasida. Automatic soccer commentary and RoboCup. In *Proceedings of the Second International Workshop on RoboCup*, pages 34–49, 1998.
3. K. Tanaka-Ishii, I. Frank, I. Noda, and H. Matsubara. A statistical perspective on the RoboCup simulator league: Progress and prospects. In *Proceedings of the Third International Workshop on RoboCup*, pages 114–127, 1999. Scientific Challenge Award Paper.
4. K. Tanaka-Ishii, I. Frank, and K. Arai. Trying to understand RoboCup. *AI Magazine*, 21(3):19–24, Fall, 2000.
5. M. Tambe, G. Kaminka, S. Marsella, I. Muslea, and T. Raines. Two fielded teams and two experts: A RoboCup challenge response from the trenches. In *Proceedings of IJCAI-99*, volume 1, pages 276–281, Stockholm, Sweden, 1999.

Using Simulated RoboCup in Undergraduate Education

Fredrik Heintz[†], Johan Kummeneje[‡], and Paul Scerri[†]

[†] *Department of Computer and Information Science*

Linköping University

SE-581 83 Linköping, Sweden

E-mail: {frehe, pausc}@ida.liu.se

[‡] *Department of Computer and System Sciences*

Stockholm University and the Royal Institute of Technology

Electrum 230, SE-164 40 Kista, Sweden

E-mail: johank@dsv.su.se

Abstract. We argue that RoboCup can be used to improve the teaching of AI in undergraduate education. We give some examples of how AI courses using RoboCup can be implemented using a problem based approach at two different Universities. To reduce the negative aspects found we present a solution, with the aim of easing the burden of grasping the domain of RoboCup for the students, RoboSoc which is a general framework for developing simulated RoboCup agents.

1 Introduction

Several World Cup 1999 teams made extensive use of undergraduate skills and time in their development, often resulting in undergraduates theses [Hei00, Lyb99, Ril99]. Another, larger, group of interested undergraduate students come into contact with RoboCup during normal undergraduate courses. RoboCup has been successfully used in an AI course at Linköping university since 1997 [CM99] and in two courses on agents at Stockholm University with good results during the years 1999-2000. The courses focus on the simulated RoboCup competition because it allows students to study AI and agent technologies in an exciting framework without the expense and expertise required by the real robot leagues. By using a problem based learning approach to courses using RoboCup we argue that RoboCup's educational value can be further increased. However to do this effectively requires a generic library, like RoboSoc, that handles the basic interactions with the simulator, so that the students can focus on the main topic of the course.

2 Using Simulated RoboCup to Teach AI

Using RoboCup in education is generally a positive experience for all involved, albeit with some caveats. One way of improving RoboCup courses is to use

problem based learning (PBL) teaching techniques. Although not universally agreed on, for the purposes of this paper we define PBL as *posing questions to be answered or problems to be solved, with the role of the teacher changed to be a coach rather than an expert teacher* [Amb92].

A problem with teaching abstract science is finding illustrative examples for the concepts presented, i.e. finding real problems to solve with the abstract methods provided by the sciences. The prevailing teaching paradigm is based on pedagogy (meaning to teach children). Within this paradigm, the teacher instructs the student on how to solve a problem step by step as well as telling what problems are solvable with this problem-solving method. Thus, the teacher performs the greatest part of the cognitive processing. This is the way most children are taught in first grade.

Andragogy (the teaching of adults) differs from pedagogy in that it is more of a process. In the process the teacher becomes a coach and the students are the driving force. Closely related to andragogy is the concept of PBL. In PBL the student acquires knowledge in a search for solutions to a problem. The problem is formulated with the task being to find suitable solutions to the problem. This contrasts to pedagogy where the task is to first acquire knowledge then, hopefully, find a problem requiring the acquired knowledge.

Since it is not always obvious what knowledge is required, RoboCup is well suited for PBL. We have used PBL in all the courses and found that it increases the interest among the students as they are allowed to formulate their own specific problem and solutions. As RoboCup poses a problem that has no “right” solutions, students are challenged to try to improve on both the ideas they read about and hear from their peers. In RoboCup, AI is mixed with more traditional computer science concepts such as sockets and multi-threading, exposing students to a variety of topics at the same time – which is both good and bad.

PBL decreases the course-specific preparational load but increases the general preparational load for a teacher. For the RoboCup courses, as is common with PBL courses [Amb92], the teachers are or were active researchers in the field hence had already done basic preparational work. Furthermore, as [Amb92] notes in the domain of medical education, PBL is also “fun” for all involved. A teacher is made to look at the area from a new perspective, as the subject is discussed and innovative solutions proposed [Amb92]. The students-teacher discussion constitutes yet another academic conversation, yet often one with a novel angle. This has actually led to the researcher/teacher being aided in his current RoboCup research by teaching a PBL course.

A problem with PBL is that the students usually needs more support during the course. To minimize the extra work for the teachers it is very important to make goals, assignments and expectations on the students as clear and thorough as possible. If the requirements are unclear the students might lose interest in the course or do something totally different than what was expected by the teachers.

3 Description of the Courses

Three AI courses have been held all with RoboCup as a vital element, though with slightly different focus. The courses *Agent Programming I and II* were taught at Stockholm University and the course *AI Programming* was taught at Linköping university.

In *Agent Programming I* [Int00], held Spring 2000, 10 of the 40 participants chose to take the RoboCup based assignment. The aim of the assignment was to let the students learn more about agents and AI by hands on experience. To this end students used an existing team, UBU [KLY99], to design and perform their own experiments. The resulting student reports will be made available to future students as a source of inspiration, for reference and as a launching pad for future experimentation.

Agent Programming II [Int99], held Spring 1999, had 14 participants. The students chose from three tracks, namely Social Aspects of Agents, RoboCup, and Artificial Decision Makers. Six students took the RoboCup-track, three of whom ended up working on UBU, a 1999 World Cup participant [BKLY99]. The other three students researched RoboCup relevant areas, including genetic programming and neural nets.

Since 1997 an AI programming course based on RoboCup has been taught at Linköping university [CM99, AIP]. The course is project based, with students in groups of two or three designing and implementing a RoboCup team. Initially, a series of lectures on relevant topics are given. At the end of the course there is a competition between the student teams and some publicly available teams. Together with a written report, the team's design and performance in the competition constitutes the student's examination.

The Students Perspective From a student's perspective, the major problem with using RoboCup in a course is that building RoboCup agents requires a range of non-AI knowledge, like process programming and networking. Hence a large effort and code is required just to get a simple agent doing things like communicating with the server, parsing the server messages, calculating it's position and sending simple commands at the right time. It is not surprising that the most requested RoboCup course improvement from students has been for more help with the practical problems of developing RoboCup agents [CM99].

To solve these problems, RoboSoc, a framework for developing RoboCup teams was developed [Hei00]. In 1999, the AI Programming course in Linköping used an incomplete version of RoboSoc. In an informal evaluation after the course students reported that RoboSoc improved their ability to focus on the issues they were interested in. The students had less problems creating working teams, of surprisingly high quality, and implementing their own skills and decision making within the RoboSoc framework. There were, however, some concerns about the implementation and documentation of RoboSoc.

At Stockholm University the existing team UBU was made available to the students. The students were very positive about the assignment, generally finding it to be very relevant to the course material, for example, getting insights in

the application of neural nets in real-time environments. The students appreciated working on a real software project as opposed to working on an artificial assignment, though some commented that it was a too large a system to grasp in a couple of weeks – especially given the poor documentation. Other students were, however, a little disappointed that they could not implement their own teams from scratch.

4 RoboSoc

For RoboCup beginners there is a need for a well documented, well structured and generic library which helps and encourages developers to create and share their code. Previous attempts to build RoboCup libraries have usually begun with a team releasing parts of their code. The problem with such efforts is the often limited documentation and the released code not being as generic as one would hope, simply because it was not intended to be generic. People attempting to use the released code are too heavily tied to the assumptions and design decisions of original developers. RoboSoc was designed to fill this gap and was subsequently used to develop a World Cup team, rather than the usual opposite ordering. The use of RoboSoc also lowers the amount of extra support needed from the teachers and reduces the risk of the students loosing interest in the course due to too many possibilities and unclear tasks, since it creates a framework within which the students should develop their RoboCup agents.

Students struggle with the practical problems of working with the RoboCup simulator – RoboSoc aims to solve those problems. By using RoboSoc it becomes possible to share solutions of the students with others and make it possible to extend them since the a growing body of students will become familiar with the framework. Using RoboSoc in RoboCup relayed courses lets the students focus on the interesting AI-aspects of the problem. Since RoboSoc has a modular structure, choosing an area of interest and working on that area is possible without affecting the overall system. Trying out different solutions to the problem of interest is thus straightforward. The resulting collection of modules can be made available to other users both in the current year and in the future, allowing the teams to improve over the years. The design means students can go deeply into a single area rather than being forced to learn a little about a lot.

The features of RoboSoc are:

- The interface between the decision maker and the rest of RoboSoc is based on events, which makes it very general.
- RoboSoc handles the interactions with the soccer server, including timing of the agents reactions.
- Modularized world model, with support for sharing, extending, and expressing dependencies between modules.
- Modularized procedural actions, with support for sharing, extending, and expressing dependencies between modules.

The resulting system consists of the three parts described below, the library, the basic system, and the framework shown in figure 1.

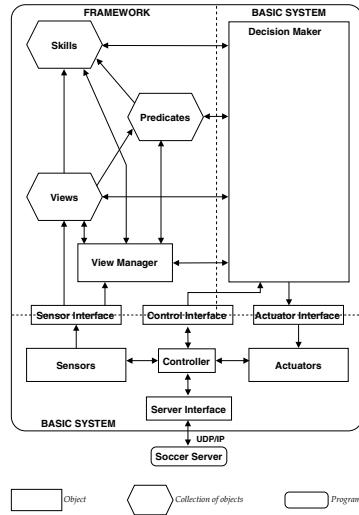


Fig. 1. The RoboSoc architecture.

The RoboSoc library is the foundation for the rest of the system. It provides the necessary types and tools needed by the other parts of the system. There are classes for representing geometric objects, the objects found in the soccer environment and the commands that can be sent to the soccer server. It also defines basic types (integers and floating-point numbers) that can handle unknown values, and uncertainty via the use of probabilities.

The basic system takes care of the interactions with the server, i.e., sending and receiving data. It is also responsible for the timing and provides the basic support for decision making by generating events when the environment changes.

On top of the basic system a framework is designed to support and modularize world modeling and adding support for decision making. By providing building blocks, the user can use existing blocks, extend them with new functionality and create completely new blocks. The framework defines three concepts: *view*, *skill*, and *predicate*. They each encapsulate an important concept used in the information processing and the decision making. A view is a way of looking at information, focusing on some special property or object, from a larger collection of data (which is dependent either on the current time or on the history of the agent). A skill is a complex action, combined of several primitive actions, that will take an agent towards a certain goal state. A predicate can be used either by the decision maker or the skills as a test on the state of the world. The framework depends on both the library and the basic system.

5 Summary

RoboCup is a good domain for demonstrating AI techniques and to let the students get a feeling for what AI is and how it can be used in real applications.

We believe that RoboCup can be used with a traditional style of teaching and we know that it can be used with a problem based learning approach.

As [Kum99, Hei00, Bom99] note, the main problem of using RoboCup in education is the time it takes to come to terms with the peculiarities of the domain, and the focus that is put on improving low level skills, like passing the ball in an optimal way. However RoboCup enables a PBL based learning approach to be used for teaching undergraduate students. We, as have others, have found this way of teaching, especially when RoboCup is used, to be fun and rewarding for both teachers and students. With the development of generic libraries, like RoboSoc, and the use of existing teams some of the negative aspects of using RoboCup and PBL are reduced while the positive aspects are magnified.

References

- [AIP] AI-Programming course web-page. <http://www.ida.liu.se/~TDDA14/>. verified 18th July 2000.
- [Amb92] George Ambury. Beginning to Tutor Problem-Based Learning: A Qualitative Investigation of Andragogy in Medical Curriculum Innovation. presented at the annual meeting of the CASAE, Ottawa, Canada., June 1992.
- [BKLY99] Magnus Boman, Johan Kummeneje, David Lybäck, and Håkan L. Younes. UBU Team. In *RoboCup-99 Team Descriptions*, pages 133–138, 1999.
- [Bom99] Magnus Boman. Agent Programming in RoboCup'99. *AgentLink NewsLetter*, (4), November 1999.
- [CM99] Silvia Coradeschi and Jacek Malec. How to make a challenging AI course enjoyable using the RoboCup soccer simulation system. In *RoboCup-98: The Second Robot World Cup Soccer Games and Conference*, pages 120–124. Springer verlag, 1999.
- [Hei00] Fredrik Heintz. RoboSoc a System for Developing RoboCup Agents for Educational Use. Master's thesis, Department of Computer and Information Science, Linköping university, March 2000.
- [Int99] Agent Programming II Course Home Page. <http://www.dsv.su.se/~mab/7v99/Index.html>, 1999. verified 18th July 2000.
- [Int00] Agent Oriented Programming Course Home Page. <http://www.dsv.su.se/~mab/4v00/>, 2000. verified 18th July 2000.
- [KLY99] Johan Kummeneje, David Lybäck, and Håkan L. Younes. UBU - an object-oriented RoboCup Team. In Johan Kummeneje and Magnus Boman, editors, *Int7 1999 Papers*. 1999.
- [Kum99] Johan Kummeneje. Simulated Robotic Soccer and the Use of Sociology in Real Time Mission Critical Systems. In L. R. Welch and M. W. Masters, editors, *Proceedings of RTMCS Workshop*. IEEE, December 1999.
- [Lyb99] David Lybäck. Transient Diversity in Multi-Agent Systems. Master's thesis, Department of Computer and Systems Sciences, Stockholm University and the Royal Institute of Technology, September 1999.
- [Ril99] Patrick Riley. Classifying adversarial behaviors in a dynamic, inaccessible, multi-agent environment. Technical report, Computer Science, Carnegie Mellon University, 1999. CMU-CS-99-175.

Path Planning of a Mobile Robot as a Discrete Optimization Problem and Adjustment of Weight Parameters in the Objective Function by Reinforcement Learning

Harukazu Igarashi

School of Engineering, Kinki University,
Higashi-Hiroshima, Hiroshima, 739-2116, Japan
igarashi@info.hiro.kindai.ac.jp

Abstract. In a previous paper, we proposed a solution to path planning of a mobile robot. In our approach, we formulated the problem as a discrete optimization problem at each time step. To solve the optimization problem, we used an objective function consisting of a goal term, a smoothness term and a collision term. This paper presents a theoretical method using reinforcement learning for adjusting weight parameters in the objective functions. However, the conventional Q-learning method cannot be applied to a non-Markov decision process. Thus, we applied Williams's learning algorithm, REINFORCE, to derive an updating rule for the weight parameters. This is a stochastic hill-climbing method to maximize a value function. We verified the updating rule by experiment.

1 Introduction

There have been many works on the navigation [1] and path planning [2] of mobile robots. In a previous paper, we proposed a solution to path planning and navigation of a mobile robot[3]. In our approach, we formulated the following two problems at each time step as different discrete optimization problems: 1) estimation of position and direction of a robot, and 2) action decision. For the first problem, we minimized an objective function that includes a data term, a constraint term and a prediction term. This approach was an approximation of Markov localization[4].

For the second problem, we minimized another objective function that includes a goal term, a smoothness term and a collision term. While the results of our simulation showed the effectiveness of our approach, the values of weights in the objective functions were not given theoretically. This paper presents a theoretical method using a reinforcement learning to adjust the weight parameters used in the second problem.

In our reinforcement learning, the value function is defined by the expectation of a reward given to a robot's path. The path is generated stochastically because we used a probabilistic policy with a Boltzmann distribution function for determining the actions of the robot. This optimizes the local objective function stochastically to search for the globally optimal path. However, the stochastic process is not a Markov decision process because the objective function includes an action at the preceding

time in the smoothness term. The usual Q-learning method cannot be applied to such a non-Markov decision process. Thus, we applied Williams's learning algorithm, REINFORCE[5], to derive an updating rule for the weight parameters. This is a stochastic hill-climbing method to maximize the value function. The updating rule was verified by our experiments.

2 Objective Function for Path Planning

We assume that a series of actions decided at each time step would derive a desirable path or trajectory of a robot. We define the following objective function E_v of a velocity v_t at time t ,

$$E_v(v_t; r_t, v_{t+1}, r_{goal}) \square b_1 E_{goal} \square b_2 E_{smth} \square b_3 E_{clsn}. \quad (1)$$

The minimal solution of Eq. (1) gives an estimate for the robot's action at time t .

The first term represents an attractive force to the goal r_{goal} . It is defined as

$$E_{goal}(v_t; r_t, r_{goal}) \square \text{sgn}[G(v_t)] \square G(v_t)^2, \quad (2)$$

where $\text{sgn}(x)$ denotes the sign of x and $G(v_t)$ is defined by

$$G(v_t) \square \|r_{goal} \square r_{t+1}(v_t; r_t)\| \square \|r_{goal} \square r_t\|. \quad (3)$$

In Eq.(3), r'_{t+1} is defined by $r'_{t+1}=r_t+v_t$. The second term in Eq. (1), E_{smth} , is defined by

$$E_{smth}(v_t; v_{t+1}) \square \|v_t \square v_{t+1}\|^2 \quad (4)$$

to minimize changes in a robot's velocity vector. The last term in Eq. (1), E_{clsn} , represents a repulsion force for avoiding collisions with obstacles and walls. We define the term as

$$E_{clsn}(v_t; r_t) \square \begin{cases} D_{clsn} & \text{if } Dist(r_{t+1}) \square 0 \\ \square Dist(r_{t+1})^2 & \text{if } 0 \square Dist(r_{t+1}) \square L \\ \square L^2 & \text{if } Dist(r_{t+1}) \square L \end{cases} \quad (5)$$

where $Dist(r)$ means the shortest distance from the robot's position r to obstacles and walls. The constant D_{clsn} represents a degree of penalty given when the robot collides with obstacles or walls. The size L means a range within which the repulsion force starts to work on a robot. In our simulation, we set $D_{clsn}=100000$ and $L=15$.

Thrun et al. had proposed another optimization approach for motion planning [6]. In their approach, a non-Markov term, such as E_{smth} was not taken into account. We restricted and discretized a search space when minimizing the function $E_v(v_t)$. We only took into account velocity vectors whose lengths were smaller than a constant[3].

3 Learning Weights of Terms

3.1 Value Function and Probabilistic Policy

Trajectories given by minimizing $E_v(v_t)$ depend on weights of terms, $\{b_j\}$ ($j=1,2,3$)[3]. In order to control the weights properly, we applied a reinforcement learning, REINFORCE[5], which was proposed by Williams, 1992, to our path planning.

We define a value function $V(\square)$, which is an expectation of a reward $R(l_i)$ given to a trajectory l_i , as

$$V(\square) \square E[R(l_i)] \square \square_i P(l_i) \square R(l_i), \quad (6)$$

where $P(l)$ is a probability that trajectory l is produced by a policy \square . A trajectory l_i is a series of robot's positions at times $t(t=0, 1, \dots, N_i)$.

We define the policy \square using a Boltzmann distribution function as

$$\square(v_t; r_t, r_{t+1}, \{b_k\}) \square \frac{e^{\square E_v(v_t)/T}}{\prod_{v_t} e^{\square E_v(v_t)/T}}, \quad (7)$$

where $E_v(v_t)$ is the objective function shown in Eq. (1) and T is a parameter to control the randomness in choosing an action v_t at each time.

3.2 Steepest Gradient Method

We use a steepest gradient method for maximizing the value function in Eq. (6). We have to calculate the right-hand side of the following equation:

$$\frac{\square}{\square b_k} V(\square) \square \prod_i \left\{ \frac{\square}{\square b_k} P(l_i) \right\} R(l_i). \quad (8)$$

The probability distribution $P(l_i)$ is expressed by a product of $P(r_b, r_{t+1})$'s as

$$P(l_i) \square P(r_0, r_1) P(r_1, r_2) \dots P(r_{N_i-1}, r_{N_i}), \quad (9)$$

where $P(r_b, r_{t+1})$ is a probability that a robot moves to position r_{t+1} when it stands at r_t and takes a policy \square . By differentiating Eq. (9) with b_k , we obtained

$$\frac{\square}{\square b_k} P(l_i) \square \prod_{t=0}^{N_i-1} \left[\prod_{n=0, n \neq t}^{N_i-1} P(r_n, r_{n+1}) \frac{\square}{\square b_k} P(r_t, r_{t+1}) \right]. \quad (10)$$

By the following equation,

$$P(r_t, r_{t+1}) \square \prod_{v_t} P^{v_t}(r_t, r_{t+1}) \square (v_t), \quad (11)$$

we obtained that

$$\begin{aligned} \frac{\square}{\square b_k} P(r_t, r_{t+1}) &\square \prod_{v_t} P^{v_t}(r_t, r_{t+1}) \left\{ \frac{\square}{\square b_k} \square(v_t; r_t, r_{t+1}, \{b_k\}) \right\} \\ &\square \prod_{v_t} P^{v_t}(r_t, r_{t+1}) \square(v_t) \frac{\square}{\square b_k} [\ln \square(v_t; r_t, r_{t+1}, \{b_k\})] \\ &\square \prod_{v_t} P^{v_t}(r_t, r_{t+1}) \square(v_t) e_k(t), \end{aligned} \quad (12)$$

where $e_k(t)$ is called the *characteristic eligibility* of b_k [5]. Substituting Eq. (12) into the right-hand side of Eq. (10), we obtained from Eq. (8) that

$$\frac{\square}{\square b_k} V(\square) \square E \left[R(l_i) \prod_{t=0}^{N_i-1} e_k(t) \right] \quad (13)$$

This is the same result that Williams derived as his episodic REINFORCE algorithm [5]. He used a neural network model for a probabilistic policy \square .

However, we obtained a different updating rule from the REINFORCE algorithm because we used Eq. (7) instead of a neural network model for action decision. Using Eqs. (1) and (7), the characteristic eligibility of b_k is expressed as

$$e_k(t) = \frac{1}{b_k} [\ln E(v_t; r_t, r_{t+1}, \{b_k\})] - \frac{1}{T} \left[\frac{\partial E_v}{\partial b_k} \left\langle \frac{\partial E_v}{\partial b_k} \right\rangle_{T, \{b_k\}} \right], \quad (14)$$

where operation $\langle \dots \rangle$ refers to the expectation weighted with a Boltzmann factor, i.e.,

$$\langle X \rangle_{T, \{b_k\}} = \frac{\sum_{v_t} X e^{\frac{\partial E_v(v_t; r_t, r_{t+1}, \{b_k\})}{T}}}{\sum_{v_t} e^{\frac{\partial E_v(v_t; r_t, r_{t+1}, \{b_k\})}{T}}}. \quad (15)$$

By substituting Eq. (14) into $e_k(t)$ in Eq. (13) and using the steepest gradient method, we can derive the following learning rule of weights $\{b_k\}$ ($k=1,2,3$):

$$\frac{\partial b_k}{\partial \alpha} = \frac{\partial V(\alpha)}{\partial b_k} - \frac{1}{T} E \left[R(l_i) \sum_{t=0}^{N_i-1} \left\{ \frac{\partial E_v}{\partial b_k} \left\langle \frac{\partial E_v}{\partial b_k} \right\rangle_{T, \{b_k\}} \right\} \right]. \quad (16)$$

The constant α is a learning rate factor to be set at a positive small number.

Moreover, we can prove that b_k 's converge without the averaging operator $E[\dots]$ in Eq. (16) by analogy with a back propagation rule used in multi-layer perceptrons. We used the learning rule without operator $E[\dots]$ in the experiment in the next section.

4 Simulation

We consider a sample problem where a single robot moves to a goal from a starting position while avoiding static obstacles. Figure 1 shows the locations of obstacles, walls, the start point, and the goal point.

4.1 Reward Function

The objective of this sample problem is to find the path that minimizes the robot's moving time from the start point to the goal point and keeps a safe distance from obstacles and walls. We consider the following reward function $R(l)$ reflecting these two requirements : $R(l) = c_1 R_{\text{time}}(l) + c_2 R_{\text{dist}}(l)$, where $R_{\text{time}}(l)$ represents the degree of a user's satisfaction to the moving time from the start to the goal in a trajectory l . The function $R_{\text{dist}}(l)$ represents the degree of a user's satisfaction to the shortest distance from the robot to obstacles and walls if a robot moves along a path l . We call these functions *achievement functions*. We used the trapezoid-like functions shown in Fig. 2 as achievement functions. They take a value between 0 and 10 and are characterized by the parameters, a , b , c and d .

In our experiment, we set $a=0, b=20, c=50$ for $R_{\text{time}}(l)$ and $a=0, b=15, c=2000$ for $R_{\text{dist}}(l)$. This means that trajectories whose moving times are shorter than 20 and trajectories that force a robot to keep a distance longer than 15 are most desirable. The weights c_1 and c_2 are set at 0.5 to find a path that is balanced between moving time and safety.

4.2 Experimental Conditions

We set initial values of b_k 's as $b_1=b_2=b_3=1.0$ and set α in Eq. (16) at 0.00001. Parameter T is fixed to 5.0 and we did not carry out any annealing procedure. Under these experimental conditions, we repeated the learning cycle one million times. It

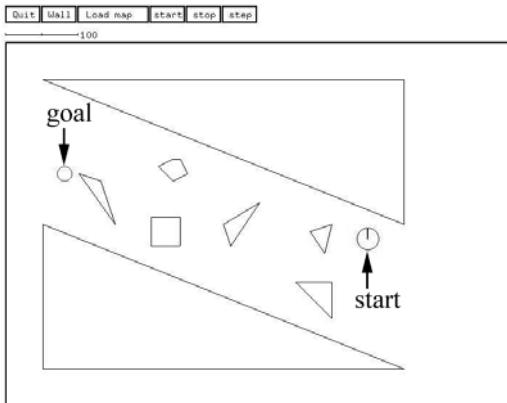


Fig. 1 Robot's world in our simulation

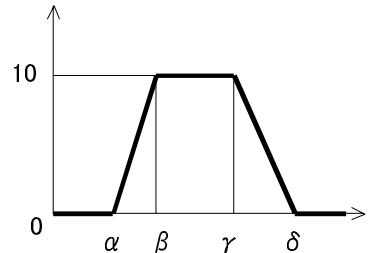


Fig. 2 General shape of achievement function

took about 17 hours to complete one experiment using a work station, SUN Ultra Sparc 30 (CPU: Ultra Sparc-II, 248 MHz).

4.3 Experimental Results

Figure 3 shows changes in R , R_{time} and R_{dist} . The values are averaged over each period of 10000 updating steps. Changes in parameters $\{b_k\}$ ($k=1,2,3$) are shown in Fig. 4.

In Fig. 3, the expectation of the reward function R increases as the learning proceeds. It increased from 4.35 to 8.17 at the end of learning. This value is about two times larger than that obtained before the learning. This increase comes from the improvement in the second term R_{dist} . The function R_{dist} forces a robot to keep a certain distance from obstacles and walls for safety. Figure 4 shows that the weight b_3 of the collision term E_{clsn} increased gradually. This improved the safety of the path.

Figures 5 and 6 show the robot's trajectories obtained using unlearned values and learned values of $\{b_k\}$ ($k=1,2,3$), respectively. The parameter T was set to a very small positive value when we obtained these two trajectories. Because we considered that the trajectory obtained at $T=0$ is a center point in the solution space for searching an optimal path and can be used to check whether values of weights $\{b_k\}$ ($k=1,2,3$) in $E_v(v_t)$ are optimized in producing desirable trajectories.

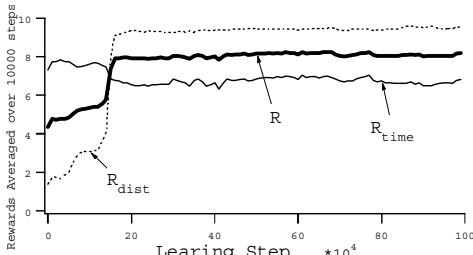


Fig.3 R , R_{time} and R_{dist} averaged over 10000 learning steps

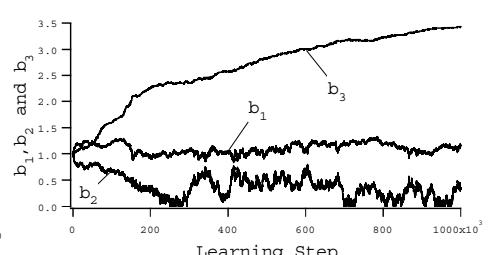


Fig.4 Learning of weights b_1 , b_2 and b_3

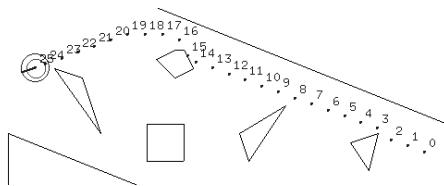


Fig. 5 A path planned deterministically with weights that had not been learned

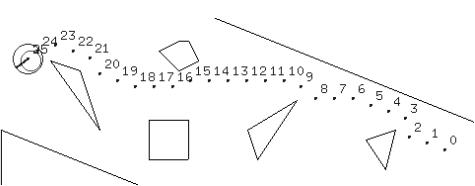


Fig. 6 A path planned deterministically with weights that had been learned

In Fig. 5, one can see that the value of the unlearned weight b_3 is so small that the robot ran into an obstacle at time 15. Figure 6 shows that the problem in safety at time 15 was solved by learning without delaying a robot's arrival time. Moreover, the trajectory was not improved locally. A global change in trajectory was achieved because the policy \square in Eq. (7) does not depend on the robot's position. Thus, if we would like to change trajectories locally to search for an optimal trajectory, we had better consider term weights $\{b_k(x,y)\}(k=1,2,3)$, which depend on a robot's position (x,y) . Our updating rule can be applied to $b_k(x,y)$'s in the same way as applied to b_k .

5 Future Work

We plan to apply our method to path planning problems of multi-robot systems. We can express interactions between robots in the objective function E_v . If a user wants to move two robots while keeping them close to or avoiding each other, it is sufficient to introduce an attractive or repulsion force between the robots into E_v . This shows the flexibility that our method can be applied to many scheduling problems in the wide-range field of engineering.

References

1. Kortenkamp, D., Bonasso, R.P., and Murphy, R. (eds.), Artificial Intelligence and Mobile Robots. AAAI Press/The MIT Press (1998).
2. Hwang, Y.K. and Ahuja, N.(1992), Gross Motion Planning—A Survey. ACM Computing Surveys 24(3): 219-291
3. Igarashi, H. and Ioi, K., Path Planning and Navigation of a Mobile Robot as Discrete Optimization Problems. In Sugisaka, M. and Tanaka, H., editors, *Proceedings of the Fifth International Symposium on Artificial Life and Robotics*, ISBN4-9900462-0-X, Oita, Japan, 2000: 297-300.
4. Burgard, W., Fox, D. and Thrun, S. (1997), Active Mobile Robot Localization. In: *Proceedings of 15th Joint Conference on Artificial Intelligence (IJCAI97)*, Nagoya, Japan, August 23-29, 1997, Vol.2, pp. 1346- 1352
5. Williams, R.J. (1992), Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning 8: 229-256.
6. Thrun, S. et al., Map Learning and High-Speed Navigation in RHINO, in [1], pp. 21-52.

Simulator Complex for RoboCup Rescue Simulation Project – As Test-Bed for Multi-Agent Organizational Behavior in Emergency Case of Large-Scale Disaster

Toshiyuki KANEDA¹, Fumitoshi MATSUNO², Hironao TAKAHASHI³,
Takeshi MATSUI⁴, Masayasu ATSUMI⁵, Michinori HATAYAMA⁶,
Kenji TAYAMA², Ryousuke CHIBA⁷, Kazunori TAKEUCHI⁷

¹ Faculty of Engineering, Nagoya Institute of Technology

² Interdisciplinary Graduate School of Sci. and Eng., Tokyo Institute of Technology

³ Port and Harbor Research Institute, Ministry of Transport

⁴ Division of Information System, Applied Technology Co.

⁵ Faculty of Engineering, Soka University

⁶ Disaster Prevention Research Institute, Kyoto University

⁷ Graduate School of Engineering, University of Tokyo

Abstract. In the RoboCup Rescue Simulation Project, several kinds of simulator such as Building-Collapse and Road-Blockage Simulator, Fire Spread Simulator and Traffic Flow Simulator are expected to provide a complicated situation in the case of the large-scale disaster through their synergistic effects. It is called Simulator Complex. This article addresses, first, system components of the prototype version of this Simulator Complex, then, explains each of the simulators and the Space-Time GIS(Geographical Information System) as DBMS(DataBase Management System). In the demonstrations, we have shown the performance enough for a test-bed for multi-agent system development.

1 Introduction

RoboCup-Rescue Simulation Project (RCRS) is being prepared as an open-competition style research activity [8] [9] [5] [7]. In RCRS, several kinds of simulators, such as Building-Collapse and Road-Blockage Simulator, Fire Spread Simulator and Traffic Flow Simulator, are expected to provide a complicated situation of a large-scale disaster through their synergistic effects. It is called Simulator Complex. On the other hand, competitors' role is to develop teams of multi-agent system, such as Fire Brigade, Ambulance Team and Rubble Cleaner, in order to simulate their emergency behaviors in cases of the large-scale disaster, i.e. Hanshin Awaji Earthquake.

Open competitions are planned, in which the competitors make their agent teams to interact one another in this Simulator Complex, not only for evaluating the agents' performances, but also for testing and evolving appropriate strategies and tactics for organizational behaviors in the various kinds of emergency cases.

Accordingly, the role of Simulator Complex is assigned to provide competitors a test-bed for development of multi-agent systems.

This article addresses, first, the system organization of the prototype version of RoboCup Rescue Simulation Project, then, explains the Building Collapse and Road-Blockage Simulator, Fire Spread Simulator, Traffic Flow Simulator, and Space-Time GIS (Geographical Information System). The GIS is organized as DBMS(DataBase Management System) in our RCRS. Demonstrations of the performance check are also referred.

2 System Organization of the Simulator Complex

Figure 1 shows the system organization. All simulators, the GIS, viewers and agents are distributed components and plugged in to the kernel, that coordinates distributed computation through a communication protocol based on UDP/IP.

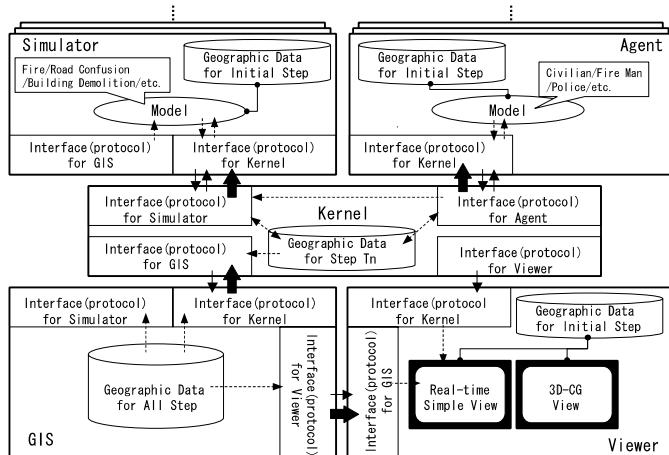


Fig. 1. System organization of the prototype version

The case area is chosen to 1.5×1.5 km width in Nagata Ward in Kobe city, where is an typical Japanese inner-city area, and mostly damaged by the fire in 1995's Hanshin Awaji Earthquake. The simulation covers the first three days after the occurrence of earthquake, by simulating phenomenon and activity for each minute by each simulation step.

3 Building Collapse and Road-Blockage Simulator(BRS)

BRS¹ calculates the collapse of buildings in the case of earthquake, and generates situations of the road-blockage.

Buildings and poles collapse and occupy the roads in the case of earthquake. Thus, as the seismic motion becomes stronger, the more collapse occurs. Figure 2 (a) shows ratio of collapse as a function of the strength of the seismic motion. As

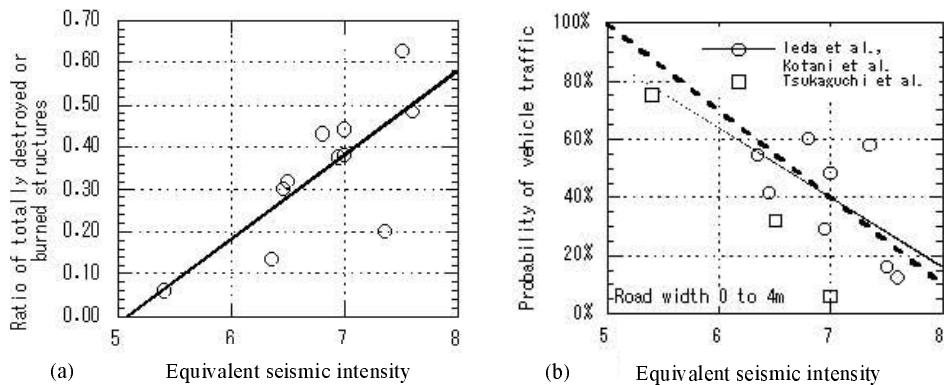


Fig. 2. Seismic intensity effects for collapse and blockade

for road width, the wider the road is, there is more possibility that a passage at the center of these roads will remain intact for traffic. The role of the blockade occurrence model is to compute the probability of vehicle traffic through the roads, which are separated into seven classes based on their width. Figure 2 (b) shows probability of vehicle traffic as a function of the seismic intensity in case of 0-4m road width [4].

4 Fire Spread Simulator(FSS)

FSS² calculates many fires at the same time in the case of the earthquake and simulates the spread process of the fires, and change the situations also depending on fire-fighting of Fire-Brigade agents.

FSS is designed as a general-purpose system that can be applied to the fire not only at earthquake cases but also at ordinary cases.

¹ The Port and Harbor Research Institute of the Ministry of Transportation has developed BRS.

² Applied Technology Co. Ltd. in collaboration with some academic researchers has developed FSS.

The computation is shown in Figure 3, which consists of the following processes: 1) combustion process, 2) spread process, 3) ignition process, and 4) effect of fire-fighting.

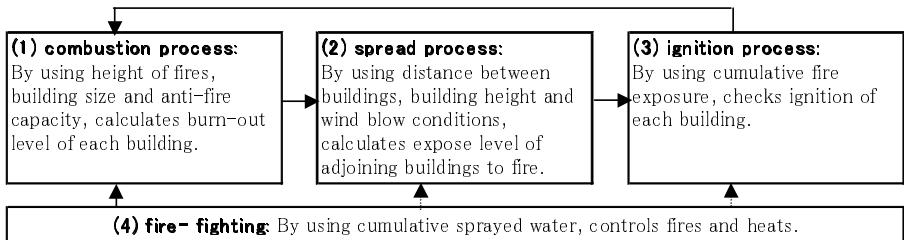


Fig. 3. Computation in each sub-process in FSS [2]

5 Traffic Flow Simulator(TFS)

Briefly saying, the role of TFS is to resolve conflicts among every agents' move plans. TFS aims at simulating traffic flows of not only vehicles but pedestrians, in various situations of the large-scale disaster, especially (1) blocked by rubbles of buildings; (2) blackout of traffic signals; (3) passing of emergency cars such as fire brigades; and (4) refuge in heaps of rubble.

A road consists of nodes in connection with the road, road width, blockage width of rubble, number of lanes of 'upward' line, number of lanes of 'downward' line, and width of sidewalk. A node consists of roads in connection with the node, existence of signal, 'split' pattern of the signal (if it has a signal), number of lanes of short-cut for left-turn, number of lanes of pocket for right-turn, and lane length of the pocket.

At the start of simulation and each simulation step, the simulator receives information from the kernel and renews information on location of every kinds of agents. Traffic is simulated each one second, which is the same as TRANSIMS [6] that is a cellular automata-based micro-simulator. The simulator sends back to the kernel simulated results in each step.

As further direction of development, we should incorporate mechanisms to simulate effects of emergency traffic control, for pursuing the possibilities of near future's information technologies such as personal information devices and autonomous monitoring system and so on. Much higher level of traffic control systems are expected by applying multi-agent system technology[1].

6 Space-Time GIS as DBMS

In order to manage spatial temporal information of a concerned simulation area, Space-Time GIS(Geographical Information System) is devised as DBMS(DataBase

Management System). We call this system DiMSIS [3], that conforms to KIWI format. The essential feature of DiMSIS is that spatial information is described based on topology calculation and temporal information is managed based on the space-time approach.

To manage temporal information, there exist the following two approaches (see also Figure 4)

Snapshot View Model: In this approach, each spatial data set is connected with the temporal information. Almost all the commercialized spatial temporal GISs have adopted this approach.

Space-time Approach Model: In this approach each element of the spatial data is connected with the temporal information. The spatial temporal GISs adopting this approach start studying recently.

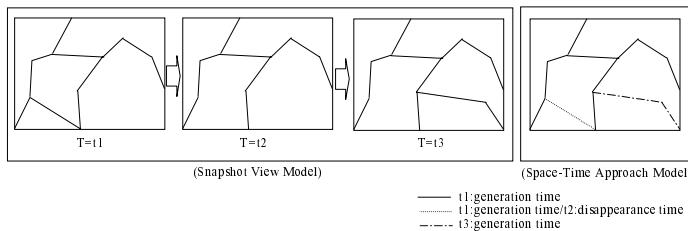


Fig. 4. Snapshot view and space-time approach for GIS data structure

Results of some theoretical study suggested that DiMSIS method (Topology Calculation & Space-Time Approach) was superior to other types.

7 Concluding Remarks

Our prototype version of the Simulator Complex had been demonstrated at RoboCup Japan Open 2000 in June, and RoboCup Workshop 2000 in August at Melbourne. In the both cases, we had connected plural PCs through 10 Base-T network, set one quarter size of the target area and near 100 agents, so it had achieved the performance one simulation step per second. It simulates a dynamic process of complicated disasters, caused by building-collapse, road-blockage and fire spread, and to evaluate various agents' activities not only refugees but also rescue organizations such as fire-brigades, ambulance teams and rabble cleaners (Figure 5). We conclude this prototype realizes the followings; (1) plug-in mechanism (changeable component); (2) distributed simulation system; and (3) comprehensive simulation of complicated multi-factor on disasters, each of which is an item of the requirements mentioned as design methodology of Artificial Society [8]. The Simulator Complex is now planned to be prepared toward Japan Open 2001 and RoboCup 2001, the first official rescue-simulation competition, through some minor revisions.

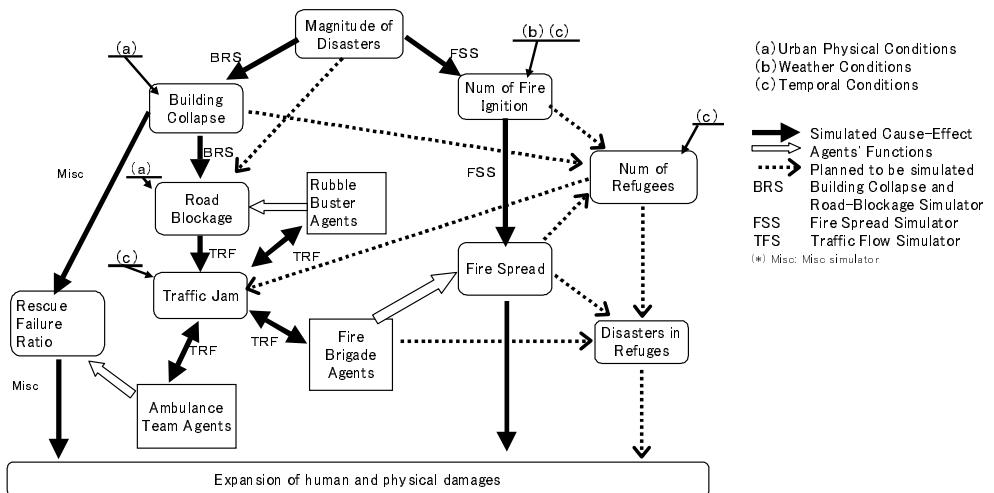


Fig. 5. Interaction among disaster simulators and agents realized in the prototype

References

1. Atsumi, M.: Adaptive Coordination of Distributed Truth Maintenance among Agents - Experimental Evaluation with Expressway Control Agency Testbed - IPSJ SIG-AI, 95-AI-102 (in Japanese). (1995) 37-42
2. Fire department, Kobe city office (eds.): Official Report on Fire in Hanshin Awaji Earthquake. Tokyo Horei Shuppan (in Japanese).
3. Kakumoto, K., Hatayama, M., Kameda, H., and Taniguchi, T.: Development of Disaster Management Spatial Information System (DiMSIS). GIS'97 Conference Proceedings. (1997) 595-598
4. Kotani, M., et.al.: Analysis on Blockage of Neighborhood Road-Networks in Hanshin Awaji Earthquake. Journal of Traffic Engineering Association (in Japanese). (1996) 101-105
5. Kuwata, Y., Shinjoh, A.: Design of RoboCup-Rescue Viewers - Toward a Real World Emergency System. Proceedings in the Fourth International Workshop on RoboCup. (2000) 42-51.
6. Nagel, K., Beckman, R., Barrett, C.: TRANSIMS for Transportation Planning. 6th Int. Conf. on Computers in Urban Planning and Urban Management '99. (1999)
7. Ohta, M.: RoboCup-Rescue Simulation: in Case of Fire Fighting Planning. Proceedings in the Fourth International Workshop on RoboCup. (2000) 119-124.
8. Tadokoro, S., et. al.: The RoboCup-Rescue Project. Journal of Japanese Society for Artificial Intelligence, Vol.15, No.5. (in Japanese). (2000) 798-806.
9. Takahashi, T., et. al.: RoboCup-Rescue Disaster Simulator Architecture. Proceedings in the Fourth International Workshop on RoboCup. (2000) 100-105.

A Quadratic Programming Formulation of a Moving Ball Interception and Shooting Behaviour, and its Application to Neural Network Control

Frederic Maire and Doug Taylor

Queensland University of Technology,
School of Computing Science, Faculty of Information Technology,
2 George Street, GPO Box 2434
Brisbane Q 4001, Australia
`{f.maire,di.taylor}@qut.edu.au`,
WWW home page: <http://www.fit.qut.edu.au/{~maire,~taylord}>

Abstract. A desirable elementary behaviour for a robot soccer player is the *moving ball interception and shooting behaviour*, but generating smooth, fast motion for a mobile robot in a changing environment is a difficult problem. We address this problem by formulating the specifications of this behaviour as a quadratic programming optimisation problem, and by training a neural network controller on the exact solution computed off-line by a quadratic programming problem optimiser. We present experimental results showing the validity of the approach and discuss potential applications of this approach in the context of reinforcement learning.

1 Introduction

Robotic soccer is a challenging research domain which involves teams of agents that need to collaborate in an adversarial environment. The fast paced nature of robotic soccer necessitates real time sensing coupled with quick behaving and decision-making, and makes robotic soccer an excellent test-bed for innovative and novel techniques in robot control. For example, the behaviours and decision making processes can range from the most simple reactive behaviours, such as moving directly towards the ball, to arbitrarily complex reasoning procedures that take into account the actions and perceived strategies of teammates and opponents. In order to be able to successfully collaborate, agents require robust basic skills. These skills include the ability to go to a given place on the field, the ability to avoid obstacles and the ability to direct the ball.

This paper concentrates on a neural network approach for the design of a robust navigation system. We show how to compute off-line an optimal trajectory using quadratic programming and how to train a neural network in order to generate a fundamental elementary behaviour for a soccer player; the *moving ball interception and shooting behaviour*. The objective of this elementary reactive

behaviour is to get the robot to intercept a moving ball and kick it in the direction of the goal at a certain time and using as little energy as possible.

1.1 Paper Outline

The paper is organised as follows. Section 2 explains the behaviour control approach and describes our system. The experimental results are presented in section 3. Section 4 puts this work in the perspective of reinforcement learning.

2 Behaviour Control

2.1 Moving Ball Interception and Shooting Behaviour

Most of the successful robot soccer teams have adopted a bottom up hierarchical approach to agent behaviours [2–5]. Some of them (see chapter 5 of [2] for a review) have chosen neural networks to implement interception and shooting behaviours, and train their control networks by reinforcement learning (whereas our off-line computation of optimal command sequences allows us to use a supervised training method).

If a robot is to accurately direct the ball towards a target position, it must be able to approach the ball from a specified direction. The task of the *moving ball interception and shooting behaviour* module is to do this economically, with the temporal constraint that the trip to the ball should take n time-steps.

The input to this behaviour module is a state vector containing the following information. The position and velocity vectors of the ball and the player, and the number of time steps (denoted by n) before the desired interception. The output of the module is an acceleration vector for the next time step. The goal is assumed to be at position $(0, 0)$. Note that the goal in this context is simply a location, that can correspond either to the structure into which the ball is driven to score, or to a location next to a team-mate to whom the ball should be passed. The assumption that the goal is at position $(0, 0)$ is not really a restriction as the general case reduces to that particular case by considering the relative positions with respect to the goal position.

In simulation, we have to make sure that the velocity and acceleration vectors of the player stay bounded. In our mathematical formulation, inequality constraints on the norms of the acceleration and velocity vectors are used to guarantee a realistic behaviour. By minimising the sum of the norms of the robot acceleration vectors, we obtain the most economical (energy-efficient) sequence of acceleration vectors to complete the interception and shooting task. An added benefit of this approach is that the robot trajectory is very smooth.

2.2 Mathematical modelling

The *state* of the playing field at time t is determined by the position, velocity and acceleration vectors of the robot and the ball at time t . We will use the following

convention for these variables; the subscript R refers to a robot variable, the subscript B refers to a ball variable, time is indicated as a superscript. According to Newtonian mechanics, the above vectors are related by $P_R^t = P_R^{t-1} + V_R^{t-1}$ and $V_R^t = V_R^{t-1} + A_R^{t-1}$. From these relations, it is easy to derive that $V_R^i = V_R^0 + \sum_{j=0}^{i-1} A_R^j$ and $P_R^t = P_R^0 + t \times V_R^0 + \sum_{i=0}^{t-2} (t-1-i) \times A_R^i$. These equations show that the position and velocity vectors depend linearly on the acceleration vector. We are looking for a sequence of bounded acceleration vectors $A_R^0, A_R^1, \dots, A_R^{n-1}$ such that the corresponding sequence of velocity vectors is also bounded by some constants V_{\max} and A_{\max} . We are using the norm sup in order to have to deal only with linear constraints. The most economical sequence of accelerations is the solution to the following quadratic programming minimisation problem;

$$\min \sum_{i=0}^{n-1} \|A_R^i\|_2^2 \quad (1)$$

$$\text{s.t. } \begin{cases} \forall i, \|A_R^i\|_\infty \leq A_{\max} \\ \forall i, \|V_R^i\|_\infty \leq V_{\max} \\ V_R^n = \frac{V_{\max}}{\|G - P_B^n\|_2} (G - P_B^n) \\ P_R^n = P_B^n \end{cases} \quad (2)$$

The constraint $P_R^n = P_B^n$ expresses the requirement that the robot hit the ball at time $t = n$. The constraint $V_R^n = \frac{V_{\max}}{\|G - P_B^n\|_2} (G - P_B^n)$ expresses the requirement that the velocity vector of the robot at time $t = n$ be in alignment with the goal. The system that we have derived is easily solved with standard mathematical packages like Matlab optimisation toolbox (see [6] for an introduction to optimisation theory).

2.3 The Time Oracle

Given an initial state, we are faced with the problem of choosing a suitable value for n . To make our system as general as possible, we made the following design decision. An auxiliary neural network, that we call *the time oracle*, is trained to learn t_{\min} the minimum value of n such that there exists a feasible solution to the system of inequalities (2). The input of the time oracle is the same state vector as for the control neural network without the entry n . The control neural network is fed with the n computed by the time oracle neural network. The time oracle acts as a critic who predicts how long it will take the robot to complete the interception task. In fact, the oracle learn $1/t_{\min}$, so that it outputs $0 = 1/\infty$ when the robot will fail to intercept the ball. The oracle is not superfluous, as the robot should not waste its energy trying to complete tasks beyond its physical capabilities.

3 Experimental Results

The architecture of the control NN was determined by trial and error. The 2-hidden layer networks seem to perform better than the 1-hidden layer networks. The control network used to produce the figures below had 30 neurons in each hidden layer (the time oracle was the same size).

In all figures of this paper, the solid line represents the trajectory of the robot. The circle on the solid line represents the initial position of the robot. The dotted line represents the trajectory of the ball. The circle on the dotted line represents the initial position of the ball. The star at position (0,0) represents the (punctual) goal. The sequence of velocity vectors of the robots can be seen on the right hand side subplots of each figure.

In figure 1, the initial velocity of the robot is South, whereas the ball is going North. To arrive at a shooting position, the robot has first to make a U-turn, then overtake the ball, then make a second U-turn and align itself in the direction of the goal. The left-top diagram was obtained using directly the QP optimiser. The left-bottom was obtained with the control NN. In figure 2, the initial velocity of the robot is North-East, whereas the ball is going West. In figure 3, the initial velocity of the robot is South-East, whereas the ball is going North-East.

All these figures correspond to non training data initial conditions. We observe that the QP optimiser and the NN control trajectories are very similar. This demonstrates that the neural network has successfully learnt the mapping from state-space to action-space that produces the optimal trajectories and generalises well.

4 Discussion and Future Work

Although our neural controller works well in simulation, we expect that when used on a real robot, the shooting performance will have room for improvement. We plan to use recently developed reinforcement learning techniques for continuous action spaces [7, 8] to ameliorate the shooting skills (accuracy) of the robot. In reinforcement learning, the outcome of the training of a neural network controller is very sensitive to the initial weights. The agent is more likely to improve its policy if the neural network implementing the policy is initialised with a reasonably good behaviour.

Extending control systems to include learning capabilities is becoming an increasingly more important issue in autonomous agent control. Our proposed system is a step in this direction.

References

1. Brooks, R.: Cambrian Intelligence. MIT Press, 1999. ISBN: 0262024683
2. Stone, P.: Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer. MIT Press, 2000. ISBN: 0262194384

3. Wyeth G.F., Browning B. and Tews A. UQ RoboRoos: Preliminary Design of a Robot Soccer Team. Lecture Notes in AI: RoboCup '98, 1604, (1999)
4. Robocup 1999 Team Description: Middle Robot League
<http://www.ida.liu.se/ext/robocup/middle/intro/index.html>
5. RoboCup-98: Robot Soccer: World Cup II Lecture Notes in AI: RoboCup '98, 1604, (1999)
6. Fletcher, R.: Practical Methods of Optimisation, 2nd Edition. Wiley, 1987. ISBN 0471915475
7. Maire, F.: Bicephal Reinforcement Learning. QUT FIT Technical Report, FIT-TR-00-01.
8. Gaskett, C., Wettergreen, D., and Zelinsky, A.: Q-Learning in Continuous State and Action Spaces. in Proceedings of 12th Australian Joint Conference on Artificial Intelligence, Springer-Verlag, Sydney, Australia, (1999).

5 Figures

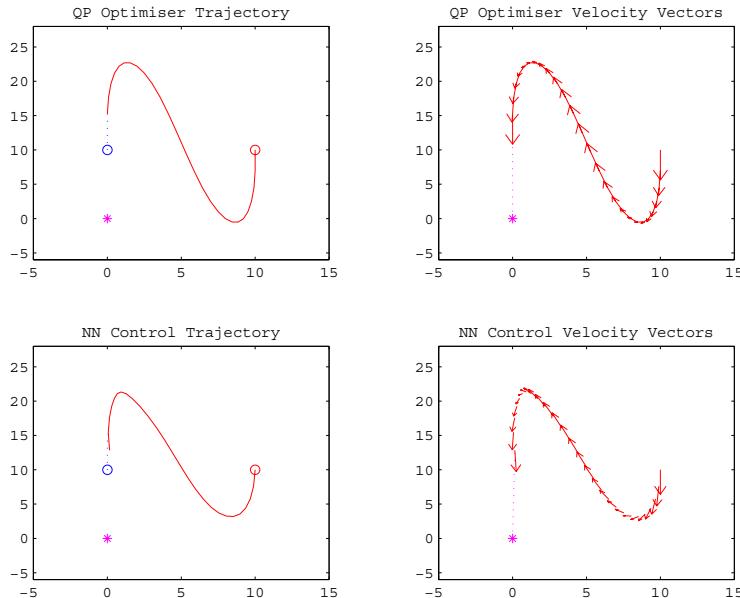


Fig. 1. Initial velocity of the robot is South, initial ball velocity is North

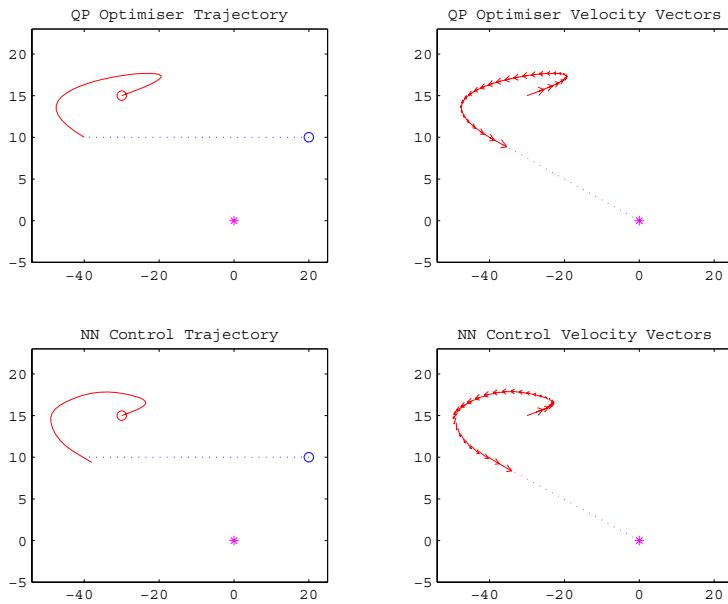


Fig. 2. Initial velocity of the robot is North-East, initial ball velocity is West

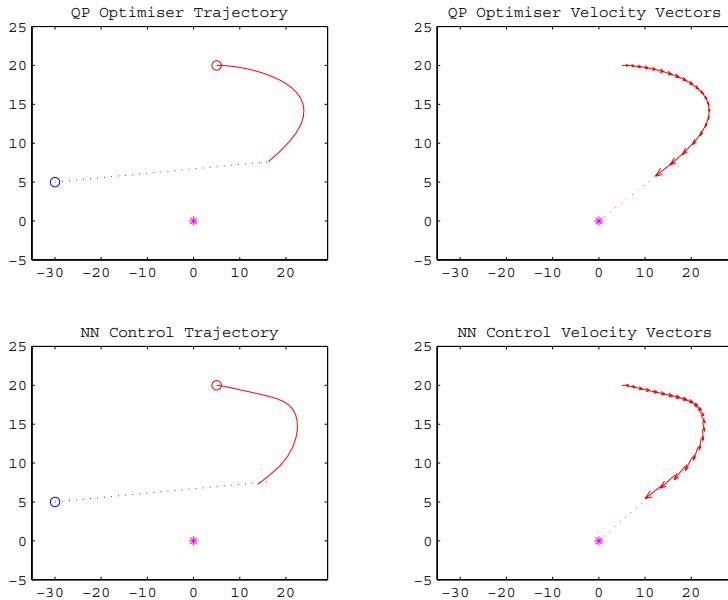


Fig. 3. Initial velocity of the robot is South-East, initial ball velocity is North-East

Keeping the Ball from CMUnited-99

David McAllester and Peter Stone

AT&T Labs — Research
180 Park Ave.
Florham Park, NJ 07932
dmac,pstone@research.att.com
<http://www.research.att.com/~dmac,~pstone>

Abstract. This paper presents preliminary results achieved during our current development of a team for simulated robotic soccer in the RoboCup soccer server [2]. We have constructed a team that plays a simplified “keepaway” game. Playing keepaway against the 1999 RoboCup champion CMUnited-99 team, our new program holds the ball for an average of 25 second with an average distance of 24 meters from the opponents end of the field. CMUnited-99 playing against itself holds the ball for an average of only 6 seconds. Here we describe the design of the keepaway team. The principal technique used is the vector sum of force-fields for governing player motion when they are not in possession of the ball.

1 Introduction

As a first step in developing a team for simulated robotic soccer we have constructed a team for playing a simplified “keepaway” game. We begin by justifying the use of the keepaway game as a simplified model of full soccer. There are two reasons for focusing on the keepaway game. First, it is much easier to measure performance in a keepaway game than in the full game. This means that a hill-climbing approach to program development is more feasible and high performance programs can be developed more rapidly. Second, a team that can hold the ball for extensive periods of time close to the opponent’s goal is likely to have more scoring opportunities than the opponents. So good performance in keepaway should translate into good performance on the full game.

As mentioned above, the first motivation for focusing on keepaway is that keepaway performance is more easily measured than goal scoring performance. One of the great frustrations in the development of a team for simulated robotic soccer is the difficulty of determining whether a given change improves performance. When the effect of a change is dramatic then one can often tell that the change is significant (usually bad) by simply watching the modified team play for a while. However if the effect on performance is only moderate, say a 20% change in scoring rate, then it can be very difficult to see the effect of the change either by watching the play or by counting goals in simulated play. To get statistical confidence intervals for performance measures, such as the rate at which goals are scored, we view the game as a Markov process and assume that the

mixing time of this process is comparable to the possession time — the average period of time between reversals in possession of the ball. We also assume that the expected time between goals is long compared to the possession time (and hence the mixing time). Under these assumptions, detecting a 20% change in scoring rate (up to 95% confidence) requires a run long enough to contain about 100 goals. In high level play, goals are infrequent and running long enough to get 100 goals can take days. For smaller improvements the situation is worse. Detecting a 5% improvement requires about 800 goals.

Using holding time and nearness to the goal as performance metrics results in a simplification of the game of soccer that we call “keepaway.” The keepaway game does not involve scoring but is similar enough, we hope, to the full game that good performance in keepaway can be converted into a high scoring team in the full game.

Our approach to keepaway is based on action generation using vector sums of “force fields”. For example there sia force field “repelling” players from the edge of the field and from each other when players get too close together. Previous research has explored action generation via vector sums. For example, the Samba control architecture [3] uses two behavior layers: the reactive layer which defines action maps from sensory input to actuator output; and the task layer which selects from among the action maps. In the robotic soccer application, a vector sum of action maps is used to determine the player’s actual motion. In this case, the vector sum is not of forces, but of low-level actions.

A previous force-field approach considering sums of attractive and repulsive forces among players and the ball is called strategic positioning using attraction and repulsion, or SPAR [6]. In contrast to our work reported here, these forces were only active over limited regions of the field, and boundaries, such as out-of-bounds and offsides, were treated as hard constraints. SPAR was implemented both in simulation and on real robots.

2 The Keepaway Game

For the experiments described in this paper, we use the RoboCup soccer server [2]. In the keepaway game used here, there is a distinguished offensive team and a distinguished defensive team. The game is played in a series of “trials.” At the beginning of a trial, the ball is placed next to the most open offensive player, i.e., the player farthest from the nearest defensive player. The trial lasts until a defensive player gains control of the ball (is within kicking range of the ball for half a second); the ball is passed in a way that violates the offside rule; or the ball goes out of bounds. When one trial ends a new trial is started by moving the ball to the most open offensive player.

A first objective for the offensive team is to hold the ball as long as possible, i.e., to make each trial last as long as possible. A second objective is to move the ball as far downfield as possible. In the experiments described here, the players are assigned random positions at the start of the first trial. However, the runs are sufficiently long that performance is dominated by an “equilibrium” player

positioning achieved after the first few trials. The keepaway game has no rules other than those ending a trial as described above. When the defensive team (CMUnited-99) gains possession of the ball, it simply holds the ball in order to end the trial, rather than trying to pass and score. Otherwise, the CMUnited-99 team plays as it would in tournament play, which includes trying to take the ball away from the offensive team.

3 The Basic Keepaway Program

The players in our experiments are built using CMUnited-99 agent skills [4] as a basis. In particular, their skills include the following:

HoldBall(): Remain stationary while keeping possession of the ball in a position that is as far away from the opponents as possible.

PassBall(t): Kick the ball directly towards teammate t .

GoToBall(): Intercept a moving ball or move directly towards a stationary ball.

In each of the keepaway programs described here, each offensive player is always in one of three modes: “with-ball”, “going-to-ball”, or “supporting-ball”. The player is in with-ball mode if it is within kicking distance of the ball. If no offensive player is within kicking distance then the offensive player that can reach the ball the soonest (as determined by a CMUnited-99 primitive) is put in going-to-ball mode. Since each player is actually in a separate process, each player must decide separately what mode it is in. Because of sensing errors, occasionally two players will both think they can each reach the ball soonest and both go into going-to-ball mode. But this is rare and one can generally think of mode assignment as being centrally determined.

In all of the keepaway teams described here, the with-ball player either executes HoldBall() or PassBall(). When a pass is kicked, the receiver generally becomes the player which can reach the ball the soonest and automatically goes into going-to-ball mode. The player in going-to-ball mode executes GoToBall(): its behavior is identical to that of the CMUnited-99 players in this mode.

In the experiments presented in section 4 the with-ball player is controlled with a somewhat elaborate heuristic. However, based on our experience with controlling the with-ball player, we believe that this elaborate heuristic achieves roughly the same performance as always passing the ball immediately and selecting the receiver that maximizes the minimum angle between the pass and a defensive player no further from the ball than the intended receiver. In the experiments described here we hold the with-ball behavior fixed so that all of the performance differences we observe are a result of differing behaviors of the players in supporting-ball mode.

In all versions of the program described here, the movements of the supporting-ball players are controlled by force fields — each supporting-ball player moves in the direction of a sum of vector fields. Players are kept in bounds with a field that repels the players from the out of bounds lines. This bounds-repellent fields

becomes infinitely strong as a player approaches an out-of-bounds line. More specifically, the bounds-repellent field is defined as follows where B_x and B_y are the x and y coordinates of the field, x and y are the player's current x and y coordinates, and x_{\min} , x_{\max} , y_{\min} and y_{\max} define the in-bounds region.

$$B_x = 5/(x - x_{\min}) - 5/(x_{\max} - x)$$

$$B_y = 5/(y - y_{\min}) - 5/(y_{\max} - y)$$

In general we arrange that a given field will tend to dominate other fields if it has a magnitude large compared to 1. The constant 5 in the above equation causes the out-of-bounds field to become strong if a player is within five meters of the edge of the playing field. At ten meters or further from any edge the bounds-repellent field is weak.

There is also an off-sides-repellent field that operates much like the bounds-repellent field to keep players on-sides. This off-sides-repellent field acts only on the x coordinate of the player and is defined as follows where O_x is the x coordinate of the force field and x_{off} is the x coordinate of the off-sides line.

$$O_x \equiv \mathbf{if}^*((x_{\text{off}} - x) \leq^1 5, -5, 0)$$

$$\mathbf{if}^*(p, x, y) \equiv p * x + (1 - p) * y$$

$$x \leq^\delta y \equiv s((y - x)/\delta)$$

$$s(x) \equiv 1/(1 + e^{-x})$$

This fairly complex formula expresses a rather simple idea. If the player is significantly less than five meters from the off-sides line then the force field pushes the player away with a force of five. If the player is significantly more than five meters from the off-sides line then the force field is negligible. The field varies continuously from a negligible value to a value near 5 as the player crosses a line five meters from the off-sides line.

In addition to the bounds-repellent and off-sides-repellent force fields, there are force fields between players. For a given offensive player there is a strategic inter-player force due to teammate i , denoted S_i , and defined as follows where d_i is the distance (in meters) to teammate i and U_i is the unit vector pointing in the direction to teammate i (all from the perspective of a player calculating forces on itself due to its teammates).

$$S_i \equiv [(d_i =^{10} 20) - 2(d_i \leq^{10} 20)]U_i$$

$$(x =^\delta y) \equiv e^{-(x-y)^2/\delta^2}$$

This is a limited range force field — the strategic force is negligible when significantly further away than 20 meters. The basic idea is that players should be

within passing distance of each other but far enough apart so that a pass between them would move the ball a significant distance. Note that S_i is a continuous function of d_i and U_i .

Players near the ball are influenced by two tactical inter-player force fields. The first, T_i , is a purely repulsive force between the offensive players. The second tactical force field, the get-clear force, denoted C , pushes a potential receiver away from defenders. The force T_i is defined as follows where again d_i is the distance to teammate i and U_i is the unit vector in the direction of teammate i .

$$T_i \equiv \text{if}^*(d_i \leq^3 8, -5, 0) U_i$$

Note that T_i is again a continuous function of d_i and U_i . The precise magnitude and direction for the get-clear force is somewhat complex and could probably be simplified without influencing the performance of the program. We do not present it here.

Intuitively, the strategic forces apply to players far from the ball and the tactical forces apply to players near the ball. The shift from “near” to “far” is done smoothly. The overall force on a supporting-ball player, denoted F is defined as follows where S is the sum over teammates i of S_i , T is the sum over teammates i of T_i , and d_b is the distance of the player from the ball.

$$F \equiv B + O + \text{if}^*(d_b \leq^{10} 20, T + C, S)$$

A supporting-ball player always tries to run, as fast as possible, in the direction of the combined force F .

4 Variations on the Basic Program

Here we consider two additional strategic force fields for controlling the supporting-ball players. The toward-ball strategic force S^b is a force of unit magnitude directly toward the ball. This force pushes supporting-ball players that are far from the ball toward the ball. The forces repelling players from each other, S and T , keep them from bunching up around the ball. The down-field strategic force, S^d , is a force of unit magnitude directly toward the opponents end of the field. In all of the variations of the program considered here, the total field controlling a supporting-ball player has the following form where the strategic field S^* is one of the fields, S , $S + S^b$, $S + S^d$ or $S + S^b + S^d$.

$$F \equiv B + O + \text{if}^*(d_b \leq^{10} 20, T + C, S^*)$$

The possession time and average x position of the ball for CMUnited and the four variations of the basic program are shown in table 1. The possession time is given as a “95% confidence interval” defined by the mean possession time over a sequence of trials plus or minus $2\sigma/\sqrt{n}$ where σ is the observed standard deviation of the possession time of a trial and n is the number of trials in the run.

Program	Possession Time	Mean Ball x Position
CMUnited	5.7-6.6	-19.5
S	16.9-18.7	-33.6
$S + S^b$	24.8-27.9	-35.9
$S + S^d$	22.2-25.2	25.7
$S + S^b + S^d$	23.7-26.8	26.6

Table 1. Offensive possession time and average x position of the ball when the offensive team is CMUnited and four variations of the basic program. The defensive team is CMUnited in all cases.

Of the four versions of the basic program, all except the basic program have essentially equivalent possession times (the differences in possession times are not statistically significant). It seems that the basic version of the program gets stuck in an unusually cramped position on the left end of the field. This cramped configuration can be “broken” in a variety of ways. e.g., by moving players nearer to the ball or moving players downfield. Once the cramped configuration is broken, a variety of behaviors have equivalent possession times. In particular, a much better ball position without significantly changing the possession time.

In summary, we believe that keepaway is a good development task because holding time, as opposed to goal rate, can be meaningfully measured, and because being able to hold the ball for longer periods should lead to better scoring performance. Our results indicate that good performance on the keepaway task can be achieved with a force field approach to action control.

References

1. H.-D. Burkhard, M. Hannebauer, and J. Wendler. AT Humboldt — development, practice and theory. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 357–372. Springer Verlag, Berlin, 1998.
2. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
3. J. Riekki and J. Roening. Playing soccer by modifying and combining primitive reactions. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 74–87. Springer Verlag, Berlin, 1998.
4. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Berlin, 2000. Springer Verlag.
5. P. Stone, M. Veloso, and P. Riley. The CMUnited-98 champion simulator team. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999.
6. M. Veloso, P. Stone, and M. Bowling. Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, Boston, September 1999.

Potential Tasks and Research Issues for Mobile Robots in RoboCup Rescue

Robin R. Murphy, Jenn Casper, Mark Micire¹

Computer Science and Engineering
University of South Florida, Tampa FL 33620, USA

Abstract. Previous work[5] has summarized our experiences working with the Hillsborough Fire Rescue Department and FEMA documents pertaining to Urban Search and Rescue. This paper discusses the lessons learned and casts them into four main categories of tasks for the physical agent portion of RoboCup-Rescue: 1) reconnaissance and site assessment, 2) rescuer safety, 3) victim detection, and 4) mapping and characterizing the structure.

1 Introduction

As chronicled in [6], we have been researching mobile robots for Urban Search and Rescue (USAR) since 1996. Our hands-on experience includes having a graduate student and thesis based on the Oklahoma City bombing [4] and attending training classes in USAR with the Hillsborough County Fire Rescue department. Previous work [5] has investigated the US FEMA (Federal Emergency Management Agency) documents concerning USAR in order to generate an understanding of the potential roles of intelligent mobile robots. This understanding, or domain knowledge, should help RoboCup Rescue provide a meaningful competition and to stimulate research which will have practical applications. This paper reviews and duplicates salient portions of [5].

Fire rescue departments in the United States are responsible for responding to four different USAR scenarios, each with its own requirements: *HAZMAT*, *bomb threats*, *collapsed buildings*, and *trench rescue*. Collapsed buildings and trench rescue share many similarities (see Fig. 1). In both cases, victims may be buried and require *search and detection*. When found, most often the victim cannot accurately describe where his/her limbs are located due to dis-orientation, stress, etc., interfering with the *extrication* effort. The victims may provide incorrect information that leads to further injury as rescuers remove rubble. In the case of a completely buried victim, rescuers depend upon witnesses and communicating with other victims in order to determine the location of other victims. Equipment used in these types of USAR efforts must be weather proof and intrinsically safe, in order to both operate under all conditions in the rubble and to withstand being washed down thoroughly if any hazardous materials or body fluids are encountered.

Collapsed buildings need small, highly mobile platforms. Current technical rescue operations can mechanically penetrate 18 feet into a collapsed structure

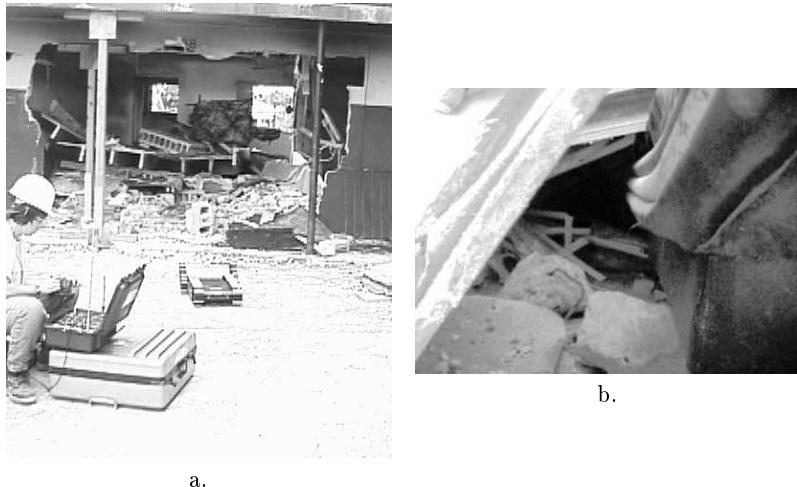


Fig. 1. a.) Testing a mobile robot at the SRDR USAR test site, and b.) a typical void formed by rubble falling against furniture.

that is un-enterable by humans. This is accomplished using a borescope to provide a video feed to the rescuers. The designer must also make the robot rugged, as it might fall or become trapped in such a way that a weaker robot would be crushed or damaged. The robot must also be self-rightable in that it will not be immobilized by being turned over. It is likely to encounter ground which will dramatically alter the robot's orientation and this must be accounted for.

2 Phases in Collapsed Building Rescues

In order to understand how the physical agent component of RoboCup-Rescue can be of use to rescue workers, it is helpful to review a rescue scenario. Collapsed building rescue efforts follow five phases. First, the incident command system for coordinating the rescuers and controlling the media and family is established and an initial reconnaissance is performed. Though the actual time depends on the magnitude of the incident, the average set up time is around three hours.

The second phase is to rescue and recover surface victims. Surface victims are people who have been hit with flying debris, or injured during a fall. They are the victims that have the best chance of surviving because they are easily reached and removed from further harm. Often surface victims are assisted by other witnesses or civilians before the rescue teams arrive on the scene. *Surface victims account for about 50% of the total number of victims recovered.* Despite the likelihood of surface victims, it is important not to approach the incident area before the site is assessed for safe entry, as secondary collapses and aftershocks immediately threaten impatient rescuers and trapped victims.

The third phase is the location and extrication of lightly trapped victims. Lightly trapped victims are those that can be removed by one or two rescuers, usually by removing a piece of furniture or light rubble. *Lightly trapped victims account for approximately 30% of the total number of victims recovered.* Victims are currently located using search dogs, listening devices, thermal cameras, etc.

The fourth phase, search of void spaces, requires trained and specially equipped rescuers working in conjunction with structural engineers. Approximately *15% of the total number of victims are found in void spaces.* These victims are trapped in a void and are unable to move. The average removal time is 4 hours with 10 highly trained and equipped rescuers. Approximately *5% of the total number of victims are entombed.* Entombed means to be trapped by main building components, such as a wall or support beam. The delicacy with which a support beam or wall needs to be removed results in an approximate victim rescue time of 10 hours. Sensors currently used to locate victims in void spaces include thermal imaging cameras, directional listening devices, fiber optic cameras (borescopes), etc.

The fifth phase is recovery and cleaning up the incident area. Recovery is different from rescue in that it is assumed that no one is left living and the rescuers are now removing bodies. Heavy equipment is used to move remove rubble. However, there is a small chance that a live victim may be found, so removal teams must be cautious.

3 Tasks for RoboCup-Rescue Physical Agents

Earlier discussions of the physical agent league for RoboCup have discussed both victim detection and extrication tasks. This poster discusses only the subtasks in victim detection. Victim detection is the first step before a victim can be extricated, also extrication is hardware intensive while much of detection requires advances in artificial intelligence and software.

3.1 Reconnaissance and Site Assessment Subtask

One potential role of mobile robots is to provide an initial survey of the site, without risking rescuers or triggering a further collapse. The building, or *hot zone*, must be assumed to be structurally unsound until a qualified civil engineer can inspect it. The weight and impact of humans walking around, climbing, or setting up equipment could trigger a further collapse.

Generally, it takes three hours for the rescue team to set up its incident command structure and to do an outer circle check. Lightweight mobile robots could be deployed immediately to do an outer circle check while the human team was setting up its incident command system and establishing the hot, warm, and cold zones. This would allow the incident commander and relevant personnel to have an accurate, metric layout of the structure. The robot(s) and supporting AI may be able to identify and rank possible voids for entry into the structure.

30 minutes	91% survive
1 day	81% survive
2 days	36.7% survive
3 days	33.7% survive
4 days	19% survive
5 days	7.4% survive

Table 1. Trapped victim survival rate.

3.2 Rescuer Safety Subtask

Intelligent robots are also needed for the humanitarian reason of saving the lives of victims and rescuers. In the 1985 Mexico City earthquake, 135 rescuers died. 65 of these deaths were due to drowning when ground water flooded the confined space area where they were searching for earthquake victims. In order to minimize potential injuries and problems, hazards need to be recognized and eliminated. There are seven major hazard categories: *structural instability, overhead, surface, and below-grade hazards, leaking utilities, hazardous materials, and incidental hazards*. A structural instability hazard could be a weakened floor. Ceiling wires threaten entanglement and are an example of overhead hazards. Surface hazards include debris laying on the ground, such as glass or nails. This threatens the rescuers as well as the rescue dogs. An example of a below-grade hazard is a broken gas pipe in a basement. Rescuers could potentially fall into the basement which might lack oxygen. Utilities running to the site could cause an explosion. Incident hazards include fire and smoke.

A robot could potentially provide the capability to recognize hazards, going into voids before humans. The robot would have to carry a sensor suite suitable for monitoring, which is quite different from navigation and victim detection.

3.3 Victim Detection Subtask

The aim of a rescue is to reduce the amount of time a victim is entrapped. Table 1 displays results of research conducted after the 1988 Armenian Earthquake[1]. The percentage of survivors is directly related to the entrapment time.

There are several ways of detecting victims used by rescuers today. The most common sensors are TV cameras, digital thermal cameras, and acoustic probes. Human rescuers most commonly use vision. While this might seem intuitively obvious, there are several facets that make this particularly difficult for robotics. First, traditional color segmentation and region detection will most likely fail because when victims are trapped within a collapsed building, they are typically covered with dust and dirt, camouflaging them within their environment. Movement would be another good visual detector to augment the color deficiencies, but because the victims are very often unconscious, this does not completely resolve the vision issues. Texture recognition may be another possible solution, but

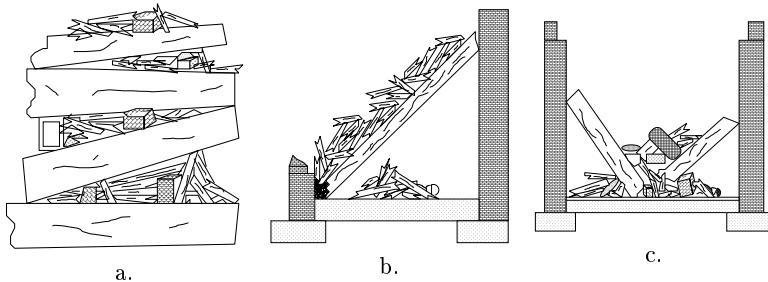


Fig. 2. Three types of collapsed building structures: a. pancake, b. lean-to, c. v-shape.

clearly much more research must be made in the area of visual victim recognition if it is to be feasible in this domain. Thermal imaging cameras have recently increased in accuracy and decreased in price as to make this an extremely useful tool for locating victims. In smoke or dust filled rooms, these temperature sensitive cameras can provide almost x-ray vision for a robot completely blind within the visible spectrum. Even lightly buried victims have a heat signature that stands out from the environment. Using adapted versions of classical image processing algorithms, false color pictures could quickly be analyzed for victim identification.

3.4 Mapping and Characterizing Structure Subtask

It should be clear from the previous tasks that any information about the layout of the structure and the nature of the collapse will help rescuers determine what resources should be deployed. The type of voids in the structure can also help determine where to look to survivors and how to plan to extricate them.

Different types of voids are created from the collapse mechanism. The three types of collapses are: pancake, lean-to, and v shape. A *pancake collapse* is prevalent with multi-story buildings and results in a cement slab stack. Entire building floors are reduced to inches in most cases. Often these voids are large enough to serve as a survivable space for victims, but are confined and difficult to access. In a pancake structure, as shown in Figure 2a, the spaces between the cement slabs needs to be searched for victims. A *lean-to collapse* is typically the result of a wall failure causing the floor or ceiling to fall on one end while being supported at the other end by the standing wall. The triangular shaped void serves as a survivable void for victims. A *v shape collapse* occurs when the center weight of a floor or ceiling is too much that the floor collapses in the middle.

4 Conclusions

Physical agents should be an important part of RoboCup-Rescue because of the potential impact of robots, especially in victim detection types of tasks. Vic-

tim detection in USAR poses four major subtasks for intelligent mobile robots: reconnaissance and assessment, rescuer safety, victim detection, and mapping. Victim detection and mapping are two tasks which are more amenable to being simulated in a competition venue. The tasks are on the “inside” of the hot zone, encouraging the development of platforms and navigation algorithms suiting for operating in confined spaces. The tasks are also amenable to automation, since it currently takes ten specially trained researchers four hours to find and remove one victim trapped in a void space and ten hours to find and remove one entombed victim. These two categories of victims account for 20% of the total victims in a building collapse. Given that the survival rate of victims drops to 33.7% after 48 hours, anything that speeds up the rescue of trapped victims will be helpful.

A testbed for physical agents should encourage the development and meaningful evaluation of the intelligence of the agents, as well as platform suitability. In particular, the league should assume that rescue workers are likely to have heterogeneous agents, with differing mobility, sensing, computation, and physical abilities. The testbed should provide opportunities to exercise *organic* sensing and behaviors needed for navigation in confined spaces as well as for *mission* sensing needed for victim detection. Also, the testbed should reflect the same percentage distribution of victims in different parts of the collapsed structure (on the surface, lightly trapped, entombed). Finally, the competition should reward the integration of software with hardware, rather than just hardware alone due to the difficulties of teleoperating robots.

Acknowledgments

The authors would like to thank Special Operations Chief, Ronald Rogers, and Jeff Hewitt of the Hillsborough County Fire Rescue, and LTC John Blitch for their helpful discussions. Some photographs were collected during constructive experiments funded under the DARPA TMR program at the SRDR USAR training site in Miami Beach.

References

1. *Rescue Systems 1*. National Fire Academy, 1993.
2. *Technical Rescue Program Development Manual*. United States Fire Administration, 1996.
3. *Standard on Operations and Training for Technical Rescue Incidents*. National Fire Protection Association, 1999.
4. J.G. Blitch. Knobsar: An expert system prototype for robot assisted urban search and rescue. *Masters Thesis*, 1996.
5. J. Casper and R.R. Murphy. Issues in intelligent robots for search and rescue. In *SPIE Ground Vehicle Technology II*, 2000.
6. R.R. Murphy. Marsupial robots for urban search and rescue. *IEEE Intelligent Systems*, 15(2):14–19, 2000.

Potential Field Approach to Short Term Action Planning in RoboCup F180 League

Yasunori NAGASAKA¹, Kazuhito MURAKAMI², Tadashi NARUSE²,
Tomoichi TAKAHASHI¹ and Yasuo MORI³

¹ Chubu University, Kasugai, Aichi, 487-8501 JAPAN

² Aichi Prefectural University, Nagakute-cho, Aichi, 480-1198 JAPAN

³ Mechatrosystems Co.Ltd., Nagoya, Aichi, 450-0003 JAPAN

Abstract. We propose a potential field approach to represent a game situation. In a potential field, a ball should be moved according to the gradient of the potential field. There are three kinds of potential fields. One is defined for a game field, and another is defined for each robot. The third field is defined as the combination of these two. The combined field is used for evaluation of a situation. We applied this method to our robot control program. Potential values are used to determine the direction in which a robot kicks a ball. We compared the potential field based strategy and an usual “if then” type rule based strategy. The potential field based strategy makes better decisions in several cases and no worse decisions than the rule based strategy.

1 Introduction

Planning actions in dynamically changing game situations is difficult. Game situations have wide variety, so writing down the actions for each situation is unrealistic. As an alternative method, action planning based on the parametric representation which is generated from some calculation is necessary.

In robotics, a potential field method has been used for obstacle avoidance and path planning [1][2]. Effect of the potential is similar to the potential energy in physics. In robotic soccer, the situation of a game changes quickly, so fast calculation is more important than precise representation. SPAR [3] obtained similar effectiveness as the potential field approach. It does not calculate a potential field directly, but executes other parametric calculations and decides the position of each robot.

We propose a potential field approach to represent a game situation. Since the potential model is simple, the calculation speed is fast enough to deal with the changes in the real world. In the potential field, a ball should be moved from its present position to the points with lower potential.

We integrated the potential field into our robot control program. In our current strategy, the direction in which a robot kicks a ball is determined from the potential field. It was only used for the local and short term action. We compared the potential field based strategy and our usual “if then” type rule based strategy. While the potential field based strategy showed as good competence as the rule based strategy as a whole, it showed good decisions in several cases.

2 Potential Field

We define a basic potential field P formed by an object in the soccer field. It is calculated by the next equation.

$$P = \frac{e}{r^n} \quad (1)$$

e indicates the “energy” of the object. The energy is a peculiar constant value defined for various objects, like an electrical charge in an electric field. r indicates the distance between the object and the point in the field where the potential is calculated. This equation is similar to the calculation of an electric field except the lack of constant term and the difference of value n . We determined n to 1/2, considering the expanse of the potential field.

We define three kinds of potential fields. One is defined for whole field, and we call this the “Static potential field”. Another is defined for each robot, and we call this the “Local potential field”. The third field is defined as the sum of these two, and we call this the “Dynamic potential field”. We indicate these three potential fields as P_S , P_L and P_D , respectively.

2.1 Static Potential Field

Static potential field P_S gives the potential of both our team’s goal and the opponent’s goal. Our goal has energy $+g$ which is the highest value in the field, while the opponent goal has energy $-g$ which is the lowest. We determined g to be 5, considering the expanse of the potential field around the goals. The static potential field is sum of the potential about our goal and the potential about the opponent’s goal. Fig. 1(left) shows the static potential field. The x and y axes display the size of the game field, while the z axis displays a potential value. The sizes 280cm and 160cm are almost equal to the sizes of actual game field. The potential is varying from our goal to the opponent’s goal gradually. The contours of the potential are drawn under the potential surface. In front of each goal, the contours spread in a radial pattern. Potential which is more than 1.0 and less than -1.0 are rounded to 1.0 and -1.0, respectively. This aims to give the same potential level to the whole goal area. The static potential field does not vary throughout the game.

2.2 Local Potential Field

Local potential field P_L gives the potential of each robot. It has a special form as shown in Fig. 1(right), that is, it has an e/r^n form for our goal side and a $-e/r^n$ form for the opponent’s goal side. Energy e is set to 1. Both teams’ robots have the same energy model. Calculation of the local potential is the same as the static potential field. The effective area of potential is limited to the inside of a certain circle. The radius of the circle is 10cm. The local potential field is given around each robot. The form of it does not vary, but it moves in the soccer field according to the movement of the robot.

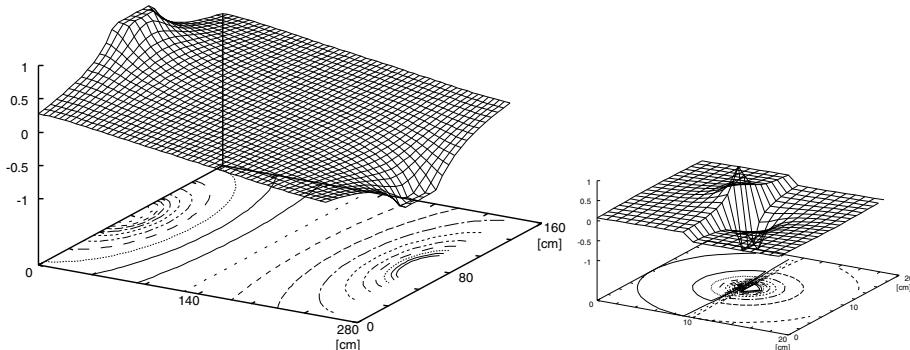


Fig. 1. Static potential field (left) and local potential field (right)

The reason why a robot has the special potential form is as follows. When a ball is placed at back side of our robot (Fig. 2 (a)), it can not kick the ball forward unless it moves to the back side of the ball. But when the ball is in front of the robot (Fig. 2 (b)), it can kick the ball immediately. Therefore the position (b) is preferable than the position (a) for our robot. When the ball is in front of an opponent's robot (Fig. 2 (c)), it can kick the ball immediately. This case is undesirable for our team. Case (d) is little better than case (c). Unifying these results, when the ball is placed at our goal side ((a), (c)), it is undesirable. When the ball is placed at the opponent's goal side ((b), (d)), it is desirable. The shape of local potential field reflects these facts.

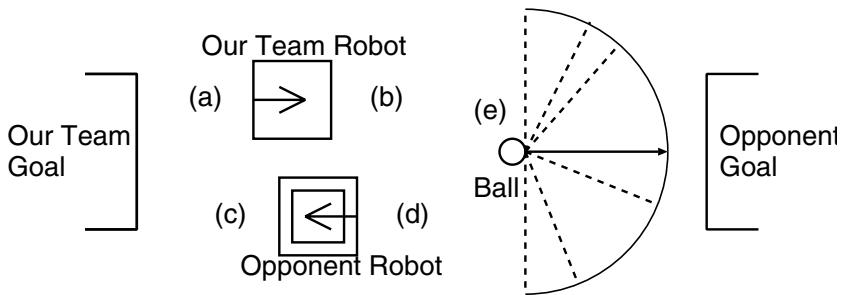


Fig. 2. Front side and back side of robots. (a) to (e) display different ball positions.

2.3 Dynamic Potential Field

Dynamic potential field P_D is given as the combination of P_S and P_L .

$$P_D = P_S + \sum P_L \quad (2)$$

Fig. 3 shows an example of the dynamic potential field. Some peak points of the local potential fields are not drawn correctly, because of the rough sampling of drawing points. All robots are moving in real time, so that the dynamic potential field should be frequently recalculated to reflect the latest situation.

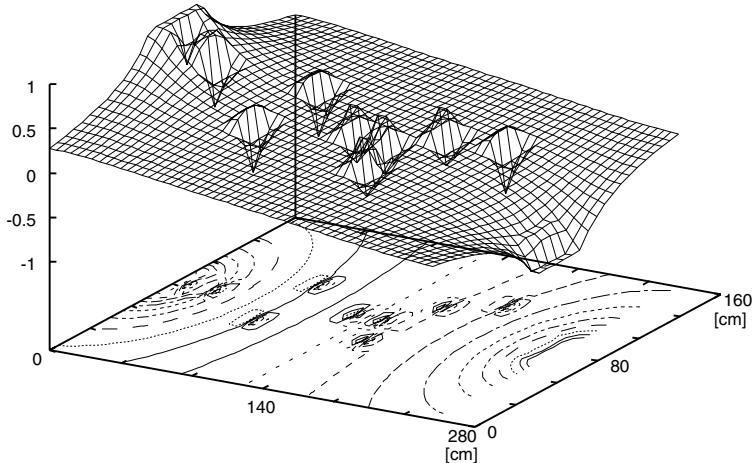


Fig. 3. Dynamic potential field

3 Strategy

3.1 Usual Rule Based Strategy

We are developing the robot control system in team Owaribito [4], which participates in RoboCup F180 League. Our team has used a “if then” type rule based strategy so far. Examples of the rules are as follows.

```
if (NEARMYGOAL == BallArea()) Clear();
if (NEARENEMYGOAL == BallArea()) Shoot();
if (SIDE == BallArea() && NEARENEMYGOAL == BallArea()) Centering();
if (YES == NearestBallPlayer()) PressBall();
```

These are the rules for the robots which approach the ball. The rule set has a hierarchy, like global strategy, an action of a robot, and a low level movement description. The robot control program analyzes the present situation and decides the next action of each robot. Our strategy program also realized a dynamic role change mechanism. It dynamically switches the role of each robot while considering the game situation. For example, two robots which are close to the ball play as midfielders(MF) try to control the ball everytime. Other robots become a forward player(FW) and a defender(DF), and help the MFs. The roles like MF, FW, DF are determined only from the positions of robots.

3.2 Integration of Potential Field into Strategy

We integrated the potential field into our robot control program. Dynamic potential field is used to determine the direction of kick. The potential field is not used for the dynamic role changing at present. The gradient of the potential field is examined around the ball every 10 degrees from -90 degree to 90 degree. The searching area is a half circular shape as shown in Fig. 2(e). The most descending direction is selected.

When a robot stands in front of the ball, the gradient value increases there. In such a case, the direction is not selected, and the most descending direction is selected instead. If some robots are surrounding the ball, the area in which no robot stands will be selected. As a result, the ball is carried to the open space where the opponent robots can not get the ball easily.

Most of strategy is the same as the usual rule based strategy including dynamic role changing, except for the decision of the kick direction using the potential field. Kicking a ball appears quite many times in soccer games, and it is a main part of all plays. The control of kicks is the basis of the strategy.

4 Evaluation

We compared the potential field based strategy and our usual rule based strategy in a soccer simulator. This simulator was developed for verification of strategies.

Although the scores of the games did not show a clear difference, the potential field based strategy showed good decisions in several cases. Fig. 4 shows snapshots of the game. In this figure, the single line boxes indicate our team's robots and the double line boxes indicate opponent robots as shown in Fig.2. Small black boxes indicate the tracks of ball movements. Fig. 4(left) shows the example of play. In this case, three opponent robots were surrounding the ball. Our robot 3 reached and kicked the ball upward. At next moment other robot kicked the ball to the right, and robot 4 could receive the ball. Fig. 4(right) shows another example. In this case, three opponent robots were surrounding the ball. Opponent robot 3 was little behind, so that our robot 4 could kick the ball downward and overtake the opponent robots. In both cases, the rule based strategy could not select the adequate kick direction due to the complexity of situation.

We sometimes observed these plays in the games. This result shows that the potential field based strategy can select the kick direction adequately. To obtain the same level of adaptivity by the rule based strategy, a complex rule set will be needed.

We measured the calculation speed of the potential field based strategy. It was carried out on a PC which has a Pentium II 300MHz processor. A calculation rate of over 200 cycles/sec was achieved. One cycle includes all of the potential field recalculation and all of the strategy decision. The potential field is currently calculated as a fine mesh with intervals of 1cm. In actual robot control, image processing is also needed. But it is confirmed that the calculation of the strategy is sufficiently fast.

In the team Owaribito, robot hardware and radio communication have almost been accomplished. But image processing of global vision is not accomplished yet. Therefore evaluation of the strategy on real robots is in progress.

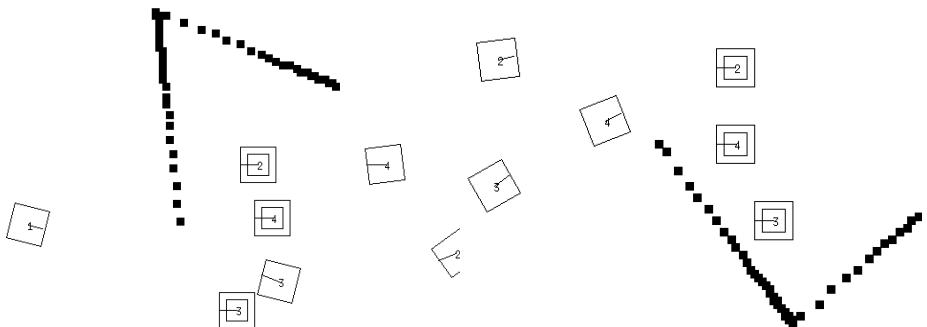


Fig. 4. Examples of play derived from potential field

5 Conclusion

We proposed a potential field approach to represent a game situation. There are three kinds of potential fields, the static potential field, the local potential field and the dynamic potential field.

We applied this method to our robot control program. We compared the potential field based strategy and an usual “if then” type rule based strategy. The potential field based strategy made better decisions in several cases and no worse decisions than the rule based strategy. We confirmed that the potential field based strategy can select the adequate kick direction. It is the merit of this method that the good adaptivity is obtained without making a complex rule set. We measured the calculation speed of the strategy, and it was sufficiently fast.

At the present time, the potential field is not applied for the team play like robot position control or passing a ball to a team mate. Applying the potential field to the team play and the long term global strategy is the future work.

References

1. Jean-Claude Latombe: Robot Motion Planning. Kluwer, (1991)
2. Yong k. Hwang, Narendra Ahuja: Gross Motion Planning – A Survey. ACM Computing Surveys, Vol.24, No.3, September, (1992) 219–291
3. Manuela Veloso, et al.: Anticipation: A Key for Collaboration in a Team of Agents. Proc. of Third Int. Conf. on Autonomous Agents, October, (1999)
4. Tadashi Naruse, et al.: OWARI-BITO Team description. RoboCup-99 Team Description, Accessible from <http://www.ida.liu.se/ext/RoboCup-99>, (1999)

RoboCup-Rescue Simulation: in case of Fire Fighting Planning

Masayuki Ohta^{1,2}, Tomoichi Takahashi³ and Hiroaki Kitano²

¹ Tokyo Institute of Technology

² Kitano Symbiotic System Project, ERATO,
Japan Science and Technology Corporation

³ Chubu University

Abstract. RoboCup-Rescue project was proposed, to examine disaster prevention and mitigation using technology from RoboCup. We have implemented a disaster simulator for RoboCup-Rescue, and use it to select the optimal distribution of fire brigades. We found the "Concentrate Strategy" is the best in this case.

1 Introduction

Disaster mitigation is attracting attention as a new domain for Multi-Agent System research, and RoboCup-Rescue[1] is proposed. We have designed and implemented a simulator for the RoboCup-Rescue. SoccerServer[2] has been successful as a test-bed Multi-Agent environment with a few dozen agents. The RoboCup-Rescue Simulation System differs from SoccerServer in that it has to be able to handle much larger number of heterogeneous agents. These differences provide a lot of new research issues for Multi-Agent systems.

In this paper, we show an example of process to select the optimal agent distribution, in this simulation system.

2 RoboCup-Rescue Simulation System

2.1 Structure of the RoboCup-Rescue Simulation System

As shown in Figure 1, the RoboCup-Rescue Simulation System consists of a core module of the system called Kernel and a number of modules which are plugged into the Kernel. It can simulate many combinations of phenomena when we plug in the necessary disaster simulators. For the sake of modularity, these plug-in components communicate with each other only via the kernel using protocols based upon UDP/IP. These protocols are formed to make it possible to pass only the necessary properties of objects, where the simulation world consists of various kinds of objects, each of which has a certain properties. This protocol does not depend on any particular simulation model and algorithm; this design allows modules to be added or removed easily.

More details about the simulation system are shown in the "RoboCup-Rescue Simulator Manual"[4].

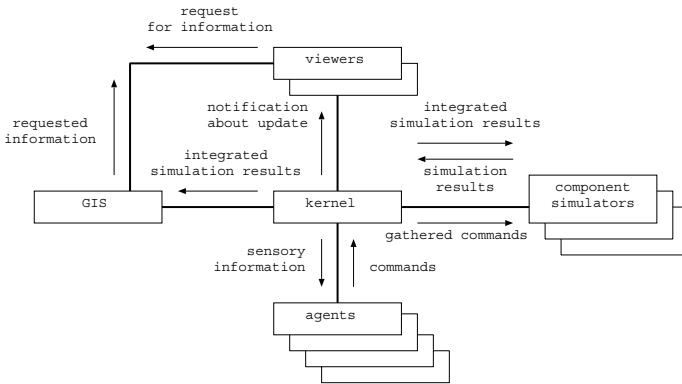


Fig. 1. Structure of RoboCup-Rescue Simulation System

2.2 RoboCup-Rescue Agent

One of the main tasks to consider in rescue activities is developing agents. Agent modules decide the action which intelligent individual is going to take. In every simulation cycle, agents receive sensory information and send back control commands. The perception information is extremely limited, and with this information, agents have to decide how they will act. Civilian, fire brigade, ambulance team and police force are now implemented.

2.3 Rescue vs. Soccer

The SoccerServer is a famous testbed for Multi-Agent research. Because of the features of the problem, rescue and soccer have many of differences as shown in Table 1. These differences provide new challenging problems.

- Agents may need to model other agents, because they can not assume their partners have the same abilities.
- Because sensory information is severely limited, agents need to look for necessary information actively.
- Because agents have to decrease the final damage of disaster, long term strategy must be considered.

3 Simulation Example

3.1 Simulation World

Using the RoboCup-Rescue Simulation System, we simulated a part of the fire fighting activities after the Hanshin-Awaji Earthquake⁴ on January 17th 1995. In this experiment we use the following components.

⁴ More than 6,500 people were killed and about 80,000 houses were collapsed by this earthquake.

Table 1. Difference between RoboCup-Rescue Simulation System and SoccerServer

	Rescue	Soccer
Number of Agents	More than 100	11
Agents in the team	Heterogeneous	Homogeneous
Information	Severely Limited	Partial
Strategy	Long Term	Short or Middle Term
Goal	Many	Only one

- RoboCup-Rescue Kernel version 0.21
- The real-world GIS data of Kobe-city (about 500m square)
- Fire simulator using “Kobe City Fire Bureau - Applied Technology - Takai’s Model”[5]
- Sample Traffic Simulator
- Sample Viewer (2D)
- Fire Brigade Agents

The simulation treats the case that, there is a big earthquake and fires have ignited at three locations (X:right, Y:left, Z:center). In the initial state, fires spread according to the fire simulator. There are nine fire brigade agents in this area. They go to the fire sites and start fire fighting. The simulation will continue until all the fires are extinguished.

The purpose of this simulation is to find the optimal distribution of fire brigade agents to fire sites. The optimal distribution minimizes the damage of the disaster. We define the number of buildings, which are damaged by fire more than 1%, as the amount of damage.

3.2 Fire Brigade Agents

In this simulation, fire brigade agents have the following features.

- It can see fire from any place in the map.
- It can see other objects (like other agents) within 30m.
- It can send a control command either “move” or “extinguish”.
- It cannot communicate with other fire brigades directly.

In this simulation, a fire brigade agent goes to the nearest burning building for which no other fire brigade is extinguishing the fire, and continues fire fighting until the fire is extinguished or the building is burned out. The agent which is assigned to an ignition point gives priority to extinguishing fires near the ignition point.

3.3 Find the Optimal Distribution of Rescue Agents

To minimize the damage of the disaster, the efficiency of the rescue agent needs to be maximized. We investigate agents' efficiency with different distributions by simulation. In this experiment, the optimal distribution is decided with the following three steps.

Acquire the “assigned number and damage” function

First, we investigate the function of “assigned number and damage” for each fire site around the ignition point, by simulation. To check each point separately, only one ignition point is activated in each simulation. We simulated the fire spread with one to nine fire brigade agents and captured the damage data.

Calculate the function of “assigned number and efficiency”

In this step, the function of “assigned number and damage” is converted into the function of “assigned number and efficiency”. We use the following equation to calculate the efficiency of agents assigned to the fire site i : E_{i,N_i}

$$E_{(i,N_i)} = \frac{Const}{D_i * N_i} \quad (1)$$

where $Const$ is the constant value, D_i is the amount of damage caused by fire site j and N_i is the number of agent engaged in the work at the fire site i . This means that D_i and N_i are inversely proportional to E_i .

Select the Best Strategy

The optimal distribution (x,y,z) is decided as the distribution that maximizes the total efficiency E_T calculated by the following equation:

$$E_T = E_{(X,N_x)} * N_x + E_{(Y,N_y)} * N_y + E_{(Z,N_z)} * N_z \quad (2)$$

3.4 Results

First, we simulated with only one ignition point to acquire the “assigned number and damage” function. Figure 2 shows the damage for each number of agents for the ignition point Z. The higher line indicates the total damage, and lower line indicates the extinguished number out of this damage.

Then, the function of “assigned number and efficiency” is calculated using equation(1) as shown in Figure 3. The constant value for the equation is 1,000,000. This curve means that, with five or six fire brigades, they can give full play to their ability, but four fire brigades are not enough to prevent the spread of fire. The graph shows that, this domain has a feature that cooperation can make the efficiency per agent be much improved[3]. So this result gives meaning to the distribution of rescue agents.

The function of “assigned number and efficiency” for the ignition point X and Y can be calculated by the same simulation. Considering these functions,

“Concentrate Strategy” X:7, Y:1, Z:1 maximizes E_T in equation(2), when the total number of agent is 9. We compare the following two strategies to confirm the result.

- “Balancing Strategy” : The same number of fire brigade agents are dispatched for each fire site. (X:3, Y:3, Z:3)
- “Concentrate Strategy” : The calculated optimal strategy. (X:7, Y:1, Z:1)

Figure 4 shows the result with “Balancing Strategy” and Figure 5 shows the result with “Concentrate Strategy”. “Balancing Strategy” could prevent the fire spread around Y, but enlarged the damage around X and Z. Final damage with each distribution was 501:405. “Concentrate Strategy” reduces the damage by about 20% compared to the “Balancing Strategy”.

4 Conclusion

The RoboCup-Rescue Simulation System provides a Multi-Agent environment to test cooperative behaviors of agents. In case of simulation of fire fighting activities on this system, cooperation can make the efficiency per agent be much improved. We presented a simulation example to select the optimal distribution of agents, and found that “Concentrate Strategy” is the best in this case.

This paper treated only the fire brigade agent. We may have different results, when using the road blockade simulator, building collapse simulator, and other kinds of agents. These topics will be addressed in future work.

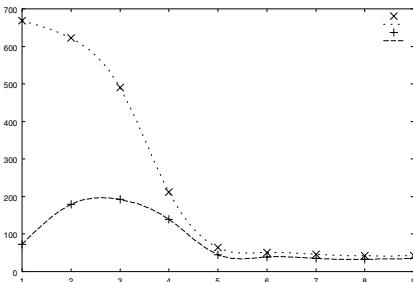


Fig. 2. Change of the Damage

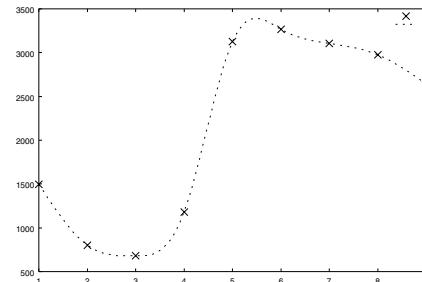


Fig. 3. Efficiency per an Agent

References

1. Hiroaki Kitano, Satoshi Tadokoro et al. RoboCup-Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research, Proc. of IEEE SMC, 1999.



Fig. 4. Simulation Result with Balancing Strategy (X:3, Y:3, Z:3)



Fig. 5. Simulation Result with Concentrate Strategy (X:7, Y:1, Z:1)

2. Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki and Ian Frank. Soccer Server: A Tool for Research on Multiagent Systems. *Applied Artificial Intelligence*, Vol.12, pages 233-250, 1998.
3. Masayuki Ohta, Tetsuhiko Koto, Ikuo Takeuchi, Tomoichi Takahashi and Hiroaki Kitano. Design and Implementation of the Kernel and Agents for the RoboCup-Rescue Proc. of The Fourth International Conference on MultiAgent Systems pages 423-424, July 2000.
4. <http://kiyosu.isc.chubu.ac.jp/robocup/Rescue/manual-English-v0r3/manual-v0r3.ps>
5. Takeshi Matsui. A Fire Propagation Simulation Including Fire Fighting Activities. Proc. of the Eighth Meeting of Special Interest Group on AI Challenges. 1999.

On Emergence of Scalable Tactical and Strategic Behaviour^{*}

Mikhail Prokopenko, Marc Butler, Thomas Howard

Artificial Intelligence in e-Business Group
CSIRO Mathematical and Information Sciences
Locked Bag 17, North Ryde, NSW 1670, Australia

Abstract. The principle of behavioral programming [1] suggests to derive low-level controllers from symbolic high-level task descriptions in a predictable way. This paper presents an extension of the principle of behavioral programming — by identifying a feedback link between emergent behaviour and a scalable Deep Behaviour Projection (DBP) agent architecture. In addition, we introduce a new variant of the RoboCup Synthetic Soccer, called *Circular Soccer*. This variant simulates matches among multiple teams on a circular field, and extends the RoboCup Simulation towards strategic game-theoretic issues. Importantly, the *Circular Soccer* world provides a basis for an architecture scale-ability evaluation, and brings us closer to the idea of meta-game simulation.

1 Introduction

Behavioural approaches to artificial intelligence often feature situatedness of agents reacting to changes in environment and exhibiting emergent behaviour, instead of reliance on abstract representation and inferential reasoning [2, 5]. Tactical and strategic reasoning, however, would seem to require domain knowledge and a degree of multi-agent cooperation beyond the reach of situated behaviour-based agents. Over the last few years, it has become apparent that a unifying architecture, combining cognitive (top-down) and reactive (bottom-up) approaches, cannot be achieved by simply connecting higher and lower layers. It has been suggested in recent literature that a “middle-out” architecture [1] is required. The approach adopted in [1] follows the *behavioural programming* principle: “taking symbolic descriptions of tasks and predictably translating them into dynamic descriptions that can be composed out of lower-level controllers”. The idea that reactive behaviours can be derived from (and importantly, can be proved to be correct with respect to) a higher-level theory follows an earlier approach — the situated automata framework [5] — in relating declarative agent representations and their provably correct situated behaviours.

While research and development work has progressed in both top-down and bottom-up directions, a systematic methodology for provably correct hierarchical architectures remains an important and open challenge. The view taken in this paper is that, rather than defining situated or tactical reasoning *ad hoc*, it is

* This paper subsumes the team description paper “Cyberoons2000: Experiments with Emergent Tactical Behaviour”.

desirable to categorise agents according to their functionality and reactions to the environment, and identify corresponding classes of agent architectures. In other words, the principal target is a systematic description of increasing levels of agent reasoning abilities, where a given behaviour can “grow” into an expanded level, while every new level can be projected onto a deeper (more basic) behaviour. More precisely, our framework (referred to as **Deep Behaviour Projection** — DBP) supports two parallel and interdependent streams:

- a hierarchical behaviour-based agent architecture, and
- a corresponding hierarchy of logic-based action theories, each of which declaratively describes an agent type and provides a basis to prove certain agent properties.

These components complement each other in the following way:

- given a formal translation procedure, agent behaviours can be automatically derived from an action theory, and proven correct with respect to it;
- an observed emergent behaviour can be formally captured and retained by an action theory of a higher level, with subsequent translation into an embedded behaviour.

The derivability/provability link from an action theory to agent behaviours fully complies with the principle of behavioral programming. On the other hand, the feedback connection from emergent behaviour to a (meta-)action theory, and then to a (provably correct) derived behaviour on a higher level, extends this principle. Thus, a successful agent behaviour can be present in the architecture in two forms: implicit (emergent) and explicit (embedded). We believe that this duplication (or depth) provides necessary functional interchangeability, and allows the agent to “mediate” among related behaviours. The results reported in [11, 12, 13] formalise the DBP approach at the situated and basic tactical levels. This paper does not introduce new systematic models and formal correctness results, but rather presents the DBP approach, highlighting its biological motivation.

2 “Deep Behaviour Projection” Agent Architecture

While designing architectures for artificially intelligent agents, it is quite normal to draw parallels with the natural world — after all, human (and animal) intelligence is so far the only available example. For instance, a canonical problem of finding resources in unknown environment and collecting them at a specified location can be simulated as the foraging problem in ants’ colonies [3] — where ants look for food and carry it towards their nest, laying down pheromones to indicate good paths to food locations. In this instance, the motivating biological example provided good intuition for experimental framework relating behavioural and genetic programming.

Motivated by biology, we intend to study increasing levels of complexity in agent behaviours and correlate these with enhanced architecture and reasoning abilities, sometimes in teamwork context. We begin at the *reactive* level — observing that certain sub-types of reactive behaviour are quite often neglected

in the AI literature and clumped indiscriminately under that generic label. The previous work [11, 12] has formally identified and analysed the types of reactive behaviours that we believe correspond to very basic animal behaviors. In particular, *Clockwork*, *Tropistic* and *Hysteretic* agent classes were studied as examples of *situated* agents. Another series of agent classes has been grouped according to *tactical* abilities: *Task-Oriented* and *Process-Oriented* types. The developed hierarchy can be briefly summarised as follows:

$$\langle C, S, E, \begin{array}{ll} sense : C \rightarrow S, & response : E \rightarrow C, \\ timer : C \rightarrow S, & command : S \rightarrow E, \\ tropistic_behaviour : S \rightarrow E, & \\ I, & hysteretic_behaviour : I \times S \rightarrow E, \\ T, & decision : I \times S \times T \rightarrow T, \\ P, & engage : I \times S \times T \times P \rightarrow P, \end{array} \begin{array}{l} update : I \times S \rightarrow I, \\ combination : T \rightarrow 2^H, \\ tactics : P \rightarrow 2^T \end{array} \rangle,$$

where C is a communication channel type, S is a set of agent sensory states, E is a set of agent effectors, I is a set of internal agent states, T is a set of agent task states, P is a set of agent process states, and H denotes the set of *hysteretic_behaviour* instantiations $\{(i, s, e) : e = \text{hysteretic_behaviour}(i, s)\}$.

The resulting architecture draws its expressive power from the situated automata and subsumption-style architectures, while retaining the rigour and clarity of logic-based representation. The Deep Behaviour Projection framework underlies this hierarchy and ensures that more advanced levels capture relevant behaviour more concisely than their deeper projections. Moreover, the depth in behaviour representation provides functional interchangeability among levels, and enables architecture scale-ability across domains. Our primary application domain is RoboCup Simulation League — an artificial multi-agent world [6], where the DBP framework provided a systematic support for design and full implementation of Cyberoos [11, 13]. Previous generations of Cyberoos developed under the DBP approach, captured certain types of situated behaviour (Cyberoos'98) and some basic classes of tactical behaviour (Cyberoos'99). Cyberoos2000 is the third “generation” designed in line with this framework. In particular, Cyberoos2000 focuses on exploring emergent tactical teamwork.

2.1 Situated Agents

A simplest perception-action feedback is implemented by the *Clockwork* agent, which is able to distinguish only between sensory states that have different time values, having no other sensors apart from a *timer*. Moreover, the *Clockwork* agent behaviour is predefined and is totally driven by time values. Like a clock-work mechanism, the *Clockwork* agent executes its fixed behaviour as a sequence of commands sent to the simulator at regular time points. Despite its almost *mechanical* simplicity, this agent type can be associated with very basic forms of cellular life driven by periodic biological cycles. In context of RoboCup, we found a pure *Clockwork* agent useful only in testing scenarios, where a player is expected to execute a given sequence of commands without synchronisation clashes (more than 1 command per simulation cycle) or stalls (no commands per simulation cycle). The activity in the *Tropistic* agent is characterised by a

broader perception-action feedback represented by the *tropistic behaviour* function. This agent reacts to its sensory inputs in a purely reflexive fashion. This kind of behaviour can be easily identified with plants, but sometimes even humans follow tropistic reflexes (eg., a hand reflexively retracts from a hot surface).

After many experiments with Cyberoos goalkeepers, we observed that a tropistic catch was the most effective behaviour in dangerous situations. However, the *Tropicstic* agent may have only a partial capacity to distinguish degrees of risk (a sensory state may, for example, omit information on play mode at a given cycle — own “free_kick” or “play_on”). Therefore, a tropistic goalkeeper will try to catch a close ball all the time — unless this behaviour is subsumed by higher levels. Other important examples of tropistic behaviour exhibited by a Cyberoos agent are obstacle avoidance and ball chasing. It is worth pointing out that regardless of how instantiations of tropistic behaviour (tropistic rules) are developed — by elaborate programming and fine-tuning, genetic evolution, or reinforcement learning — their semantics remains simple and is captured by direct mapping from sensors to effectors.

However, such a direct mapping becomes conceptually and computationally cumbersome if the number of tropistic rules grows significantly — it becomes increasingly difficult to represent and encode each behaviour instantiation, and it takes a long time to match partial sensory states in a strictly sequential computational environment.

A *Hysteretic* agent is defined as a reactive agent maintaining internal state I and using it as well as sensory states S in activating effectors E , i.e. its activity is characterised by *hysteretic behaviour*. An *update* function maps an internal state and an observation into the next internal state. Some animals seem to be proficient almost exclusively at this situated level, and yet may exhibit interesting behaviours. For example, the flocking behaviour of birds can be simulated totally at the hysteretic level with three simple rules: (i) maintain a minimum distance from other birds or other objects; (ii) match the velocity of birds in the neighbourhood; (iii) move towards the perceived centre of mass of the nearby birds [7]. Importantly, in order to behave hysteretically, an agent must maintain an internal state (containing, in the flocking example, variables for minimum distance, average velocity, distance to neighbourhood centre of mass, etc.). Faced with an obstacle, the simulated flock splits around and reunites past it. This is a classic example of *emergent*, rather than hard-coded, behaviour².

A Cyberoos2000 agent exhibits quite a few interesting examples of emergent hysteretic behaviour, eg., dribbling around opponents toward a target; intercepting a fast moving ball; resultant-vector passing; shooting at goal along a non-blocked path; etc. The *hysteretic behaviour* is implemented as a (temporal) production system (TPS). Whenever the TPS fires a hysteretic rule, an atomic commands sequence is inserted into a queue for timely execution, inherited from the *Clockwork* level. In addition, the TPS monitors currently progressing actions, thus providing an explicit account of temporal continuity for actions with

² In fact, such flocking behaviour has been proven successful to some degree in the RoboCup Simulation domain — by YowAI team from Japan in 1999.

duration [14, 13], and allowing us to embed actions ramifications and interactions [12, 13]. For example, a dribbling action is suppressed while shooting or passing.

It is worth noting that this architecture allows us to easily express desired subsumption dependencies [2] between the *Hysteretic* and *Tropicstic* levels, by an inhibition of the lower level behaviour if necessary. For instance, tropistic chase is suppressed if a teammate has possession of the ball.

2.2 Tactical Agents

The *behaviour* functions of these situated agents are not constrained. Sometimes however, it is desirable to disable all but a subset of behaviour instantiations (rules) — for example, when a tactical task requires concentration on a specific activity. The *Task-Oriented* agent type is intended to capture this feature — it incorporates a set of task states, and uses the *decision* function in selecting a subset of behaviour instantiations (a task) most appropriate at a particular internal state, given sensory inputs. A task activates only a subset of all possible rules by invoking the *combination* function. Furthermore, a *Process_Oriented* agent maintains a process state and is able to select an ordered subset of related tasks — *tactics*. Implementation of task-orientation requires some adjustments to the TPS. The TPS traces action rules whose actions may be in progress, and checks, in addition, whether a rule is valid with respect to a current task. The rules producing hysteretic behaviour mentioned in the previous section (dribbling, intercepting, etc.) are combined in corresponding tasks and can be selected by a Cyberoos2000 agent in real-time. For example, *zone playing* is implemented as a task, enabling relevant (hysteretic) rules for offside trap, making defensive blocks, cover zone, etc.

Task-orientation appears to be not only useful conceptually, but is also a practical functional element. Although it is clear that an elaborate *hysteretic behaviour* can achieve the same results as any given task-orientation, the latter captures patterns of emergent behaviour more concisely. We believe that appropriate task-orientation evolved in animals as well — to support and strengthen specialisation. One impressive example, directly related to tactical teamwork, is hunting behaviour of lions. It was found [9] that the Serengeti (Tanzania) lions most often work together when tackling difficult prey such as buffalo and zebra, but hunt alone in taking down easy prey. In Etosha (Namibia), however, lions specialise in catching the springbok — one of the fastest antelopes of all — in flat and open terrain. The research [9] has shown that “a single lion could never capture a springbok, and so the Etosha lions are persistently cooperative”. Interestingly, an analogy was drawn between Etosha lions hunting behaviour and a rugby team’s tactics, in which wings and centers move in at once to circle the ball, or prey. This “highly developed teamwork stands in sharp contrast to the disorganized hunting style of the Serengeti lions” [9].

First important lesson that can be drawn from this analogy is that **stable patterns of emergent behaviour are worth retaining** — in this instance, via suitable task-orientation. Secondly, the observed tendency in emergence of more complex tactical behaviours (eg., bird flocking to prey intercepting to prey

circling) can be correlated with **appearance of new elements in agent architecture**. Put simply, when emergent behaviour struggles to fit into the existing scheme, extension of the framework is warranted! This was obvious in the introduction of internal state that allowed the hysteretic agent to situate itself better than tropistic one in relation to other agents and environment objects. The following observation exemplifies this tendency on a tactical teamwork level.

The easily recognised pattern of “kiddie soccer”, where everyone on the team would chase the ball, could emerge as sub-optimal tactics on tropistic level — and could genetically evolve in the RoboCup Simulation domain as well [8]. We observed that *Hysteretic Cyberoos* agents exhibited another sub-optimal behaviour — solo dribbling towards the enemy goal. This kind of behaviour could emerge if there are no visible teammates and internal agent state does not keep track of them. Solo dribbling can be observed in “teenage soccer”, and arguably is more efficient than the “kiddie” one, while being more demanding and challenging in terms of the individual skills required. It may also lead to a devastating consumption of the player stamina.

A simple tactic complementing the “solo dribbling” is the so-called “backing up” — the following closely on the teammate with the ball “to assist him, if required, or to take on the ball in case of him being attacked, or otherwise prevented from continuing his onward course” [4]. It is not surprising that this simple behaviour was considered a tactical triumph in the 1870s, when the football sport was often called “the dribbling game” and forward passes were disallowed. This behaviour can be programmed on the hysteretic level as well, if desired — by appropriately inhibiting tropistic chase and elaborating (disqualifying) other competing hysteretic rules like *cover zone*. However, with teamwork tactics progressing, it becomes inconvenient and computationally expensive to keep elaborating hysteretic rules. The lessons of emergent behaviour suggest again that a new concept is required (provided in this case by the agent task state).

One particular instance of tactical behaviour, emerging at the hysteretic level, is making “defensive blocks” against an opponent dribbling towards the team goal — one defender chases the ball and tries to kick it away, while another runs towards some point on the line between the opponent and the goal. Arguably, making defensive blocks is similar to basic hunting tactics of the Serengeti lions. This tactical behaviour emerges at the hysteretic level when the second defender (lion) recognises that there is no need to directly attack the opponent (prey) — more precisely, when agent’s internal state (containing the fact that a teammate presses the opponent with the ball) resolves to subsume the tropistic chase. Again, a more efficient implementation of this tactics can be achieved with appropriate task-orientation, enabling only certain hysteretic rules.

In short, task-orientation makes tactical teamwork more explicit — when agents’ task states are complimentary, collaboration becomes more coherent. Importantly, a task does not fix agent behaviour, but rather constrains it within a set of relevant behaviour instantiations (rules). The task-orientation of the Serengeti lions would make them less successful hunters of the springbok: although their running and catching skills are probably as good as Etosha lions’ ones, the latter kind packaged the skills tactically differently.

2.3 Towards Emergence of Domain Models

The *Task-Oriented* agent is capable of performing certain tactical elements in real-time by activating only a subset of its behaviour instantiations, and thus concentrating only on a specified task, possibly with some assistance from other agent(s). Upon making a new *decision*, the agent switches to another task. In general, there is no dependency or continuity between consecutive tasks. This is quite suitable in dynamic situations requiring a swift reaction. However, in some cases it is desirable to exhibit a coherent behaviour during longer intervals.

We are currently pursuing several complementary directions potentially leading to emergence of such a deliberative behaviour (when an agent is engaged in an activity requiring several tasks).

One direction centers on the notion of process — a set of possible tasks without a precise sequential or tree-like ordering. In other words, process-orientation is not restricted to be a result of pure deliberation. An appropriate tactical scheme comprising a few tactical elements may simply suggest for an agent a possible subset of decisions, leaving some of them optional. For example, a penetration through centre of an opponent penalty area may require from agent(s) to employ a certain tactics — a certain set of elementary tactical tasks (dummy-run, wall-pass, short-range dribbling) — and disregard *for a while* another set of tasks. It is worth noting that whereas a team's tactical formation is typically a static view of responsibilities and relationships, process-orientation is a dynamic view of how this formation delivers tactical solutions. A *Process-Oriented* agent maintains a process state and is able to select an ordered subset of tasks — *tactics* — given a particular internal state, sensory inputs, current task and *engaged* process. Ideally, a *Process-Oriented* agent should be capable of consolidating related tasks into coherent processes.

Another promising direction towards deliberation introduces a domain model into the architecture. The idea of having a “world model” directly represented in the architecture is intuitively very appealing. However, we believe that “world model” should grow incrementally instead of being inserted and glued to other elements. In other words, our preference is to observe and analyse examples of emergent behaviours which potentially make use of the domain model. At the moment, Cyberoos2000 agents do not use world models and inter-agent communication, relying entirely on deep reactive behaviour and emergent tactics.

3 Evaluating Architecture Scale-ability

In order to comprehensively evaluate an intelligent agent architecture, it is desirable to compare produced behaviours under different circumstances, and in various domains. While RoboCup Simulation creates quite a challenging synthetic soccer world, current research may still be subject to a potential methodological bias. It is conceivable that designers introduce results of their own understanding of the domain *directly* into the agent architecture. Consequently, it may become rather unclear how a given architecture would scale to a reasonably different domain.

To alleviate this situation, we introduce here an extension of the RoboCup Synthetic Soccer, called a *Circular Soccer*. This variant is very similar in terms of rules to the one currently in use in the RoboCup Simulation League, with some exceptions: the number of teams competing in a single match may be greater than 2, and the stadium field is circular rather than rectangular (Figure 1 depicts the case of three teams). Other differences include a modified implementation of the off-side rules, and corner kicks. Another sub-variant, a *Closed Circular Soccer*, simulates the field boundary as a solid circular wall, based on elastic wall-ball collisions.

In our view, the *Circular Soccer* may provide a domain, where scale-ability of agent behaviours and tactical teamwork can be verified. Ideally, a team of intelligent agents should be capable of adapting to the *Circular Soccer* world, with minimal design modifications. We would argue that allowed alterations on situated level may include, for example, visual information parsing routines and triangulation algorithm(s). Tactical scale-ability can be really tested by an amount of changes required to make a team operational, and ultimately successful. Arguably, if no tactical behaviour emerges after the deployment in the new world, the architecture fails to deliver the required flexibility. Of course, we do not intend to restrict any modifications. However, the main question translates into how easy it is to re-combine tactical behaviour instantiations in order to achieve coherent tactics.

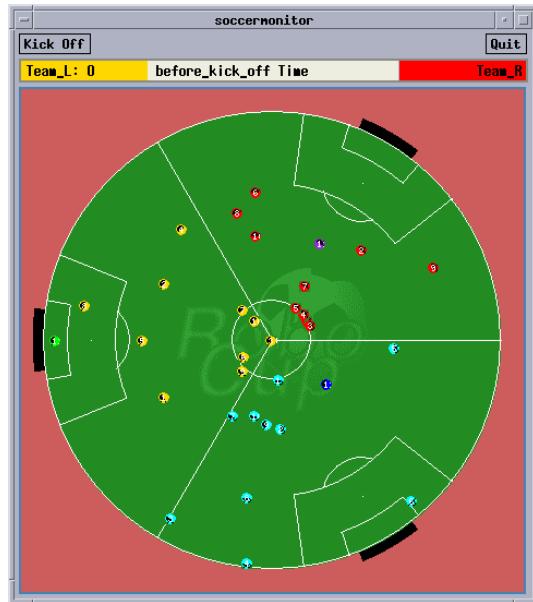


Fig. 1. *Circular Soccer*: a three teams case

Moreover, the *Circular Soccer* world provides a very interesting platform for testing *strategic* behaviour with a game-theoretic flavour. Even three competing teams, for example, bring cooperation and competition to a higher level. Let us assume the following simple zero-sum score assignment mechanism for the teams

A, *B* and *C*. If the team *A* scores against *B*, then the score of *A* is incremented by 1 and the score of *B* is decremented by 1, while the score of *C* is unchanged. At the end, teams are ranked by a single-number score, and the winner is the team with the most positive score (a team-to-team score is important only as a tie-breaker). For example, after a game where *A* scored once against *B* and once against *C*, and *C* scored four times against *B*, the overall score would be *A* : +2, *B* : -5 and *C* : +3. The team *C* is the winner, despite losing to *A* in a team-to-team contest.

Strategic behaviour becomes evident when players of one team, let us say team *A*, reason about cooperating *with* other team *against* the third. For example, in the beginning of the example match it made no difference if team *A* cooperated in attack with team *C* against *B*. However, at the end, team *A* should have cooperated in defense with team *B*, trying to prevent *C* from scoring the winning fourth goal.

In short, the *Circular Soccer* world extends the RoboCup Simulation towards strategic game-theoretic issues, and provides a basis for an architecture scale-ability evaluation. Moreover, it brings us closer to the idea of meta-game simulation in RoboCup domain. Meta-game programming is, in general, a task of writing a program that plays a game of a certain domain class, provided only with the rules of a game [10]. Rather than designing programs to play an existing game known in advance, the meta-game approach suggests to design programs to play new games, from a well-defined class, taking as input only the rules of the games. As only the class of games is known in advance, a degree of designers bias is eliminated, and meta-game playing programs are required to perform game-specific optimisations without designers assistance [10].

The concept of meta-game simulation sounds very appealing to us, as it makes the architecture scale-ability evaluation almost explicit. Put simply, a single-game behaviour will perform more strongly if it is well-tuned to the game, while a meta-game behaviour will be stronger if it is based on a more scalable architecture.

4 Conclusions

In this paper we attempted to illustrate emergence of new behavioural patterns as a good indication for inclusion of new elements in our agent architecture. This tendency has been observed while moving from *Clockwork* to *Tropistic* to *Hysteretic* to *Task-Oriented* to *Process-Oriented* agents.

We maintain that a complexity of an agent architecture is relative: for any elaborate agent type, it is possible to define more concisely another agent type with the same external behaviour. Hence, an agent has an embedded choice as to which one of related hierarchical levels should assume control to better suit external environment. If successful, such interchangeability among levels offers useful (and potentially vital) duplication and deep functional flexibility. In summary, the “middle-out” layer is required between any two levels of an intelligent agent architecture — and animals (including humans) seem to maintain an expertise and abilities providing just that. Given a formal framework (such as

DBP), we can attempt to logically prove that two agent types produce identical emergent behaviour — i.e., the *interchangeability* can be proven correct in terms of external dynamics. Therefore, rather than searching for a mysterious hub connecting “reactive behaviour” and “cognitive skills”, we should identify and study dialectic relations between emergent behaviour and elements of agent architectures. This might allow us to link architecture design and behavioural programming in a more systematic, concise and predictable way.

References

1. Michael S. Branicky. Behavioural Programming. In AAAI Technical Report SS-99-05, the AAAI-99 Spring Symposium on Hybrid Systems and AI, 29–34, Stanford, 1999.
2. Rodney A. Brooks. Intelligence Without Reason. In Proceedings of the 12th Int'l Joint Conference on Artificial Intelligence, 569–595 Morgan Kaufmann, 1991.
3. Stephane Calderoni and Pierre Marcenac. Genetic Programming for Automatic Design of Self-Adaptive Robots. Genetic Programming, Springer-Verlag Lecture Notes in Computer Science, Vol. 1391.
4. Paul Gardner. The Simplest Game. The Intelligent Fan's Guide to the World of Soccer. MacMillan USA, 178–179, 1996.
5. Leslie P. Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. In Maes, P. (ed) Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, 35–48, MIT/Elsevier, 1990.
6. Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela M. Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda and Minoru Asada. The RoboCup Synthetic Agent Challenge. In Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997.
7. Stephen Levy. Artificial Life. The Quest for a New Creation. New York, 76–80, 1992.
8. Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson and James Hendler. Co-evolving Soccer Softbot Team Coordination with Genetic Programming. In RoboCup-97: Robot Soccer World Cup I (Lecture Notes in Artificial Intelligence No. 1395), H. Kitano, ed. Berlin: Springer-Verlag, 398–411, 1998.
9. Craig Packer and Anne E. Pusey. Divided We Fall: Cooperation among Lions. Scientific American magazine, May, 1997.
10. Barney Pell. Metagame: A New Challenge for Games and Learning. In H.J. van den Herik and L.V. Allis, (ed), Heuristic Programming in Artificial Intelligence 3, 1992.
11. Mikhail Prokopenko, Ryszard Kowalczyk, Maria Lee and Wai-Yat Wong. Designing and Modelling Situated Agents Systematically: Cyberoos'98. In Proceedings of the PRICAI-98 Workshop on RoboCup, 75–89, Singapore, 1998.
12. Mikhail Prokopenko. Situated Reasoning in Multi-Agent Systems. In AAAI Tech. Report SS-99-05, the AAAI-99 Spring Symp. on Hybrid Systems and AI, 158–163, Stanford, 1999.
13. Mikhail Prokopenko and Marc Butler. Tactical Reasoning in Synthetic Multi-Agent Systems: a Case Study. In Proceedings of the IJCAI-99 Workshop on Non-monotonic Reasoning, Action and Change, 57–64, Stockholm, 1999.
14. Erik Sandewall. Towards the Validation of High-level Action Descriptions from their Low-level Definitions. Linköping electronic articles in Computer and Information science, (1):4 1996.

Karlsruhe Brainstormers - A Reinforcement Learning approach to robotic soccer

M. Riedmiller, A. Merke, D. Meier,
A. Hoffmann, A. Sinner, O. Thaté, and R. Ehrmann

Institut für Logik, Komplexität und Deduktionssysteme
University of Karlsruhe, D-76128 Karlsruhe, FRG

Abstract. Our long-term goal is to build a robot soccer team where the decision making part is based completely on Reinforcement Learning (RL) methods. The paper describes the overall approach pursued by the Karlsruhe Brainstormers simulator league team. Main parts of basic decision making are meanwhile solved using RL techniques. On the tactical level, first empirical results are presented for 2 against 2 attack situations.

1 Introduction

The main motivation behind the Karlsruhe Brainstormer's effort in the robocup soccer domain of the simulator league is to develop and to apply Reinforcement Learning (RL) techniques in complex domains. Our long term goal is a learning system, where we only plug in 'Win the match' - and our agents learn to generate the appropriate behaviour. The soccer domain allows more than $(108 \times 50)^{23}$ different positionings of the 22 players and the ball - the complete state space considering object velocities and player's stamina is magnitudes larger. In every cycle, even a non-ball-holding agent can choose between more than 300 basic commands (parametrized turns and dashes), which makes a choice of 300^{11} joint actions for the team *per cycle*. This complexity is a big challenge for today's RL methods; in the Brainstormer's project we are investigating methods to practically handle learning problems of such size.

2 Robotic Soccer as a Reinforcement Learning Problem

The problem that we face in a robotic soccer game can be formulated as finding an optimal policy π^* in a Markov Decision Problem $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p(\cdot|\cdot), c(\cdot)\}$. A state s of the state space \mathcal{S} consists of position and velocity of the 22 players and the ball, an action $a \in \mathcal{A}$ is a 22-dimensional vector of basic commands kick/ turn/ dash (one for each player), the world model $p(s_{t+1}|s_t, a_t)$ gives the transition probability that the successor state s_{t+1} is reached when action vector a_t is applied in state s_t . Finally, each transition causes costs that occur, when action a is applied in situation s . Since we are generally interested in controlling one team, we will assume that we can only choose the eleven entries in the action

vector that correspond to our team. For the rest of the paper, we consider the remaining eleven components of the opponent team to be chosen by an arbitrary and unknown, but fixed policy. The task in an MDP is to find an optimal policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$, that minimizes the expected costs, accumulated during a trajectory: $J^*(s) = \min_{\pi} E\{\sum_{t=0} c(s_t, \pi(s_t)) | s_0 = s\}$. A central team policy π would have to assign each state an eleven-dimensional command vector.

Reinforcement Learning in principle is designed to find optimal or approximately optimal policies in MDPs where the state space is too large to be tackled by conventional methods. The basic idea is to approximate the optimal value function by iteratively refining estimates for J^* : $J_{k+1}(s_t) \leftarrow \{c(s_t, a_t) + J_k(s_{t+1})\}$. However, the robotic soccer domain is such complex, that currently known RL algorithms would have no chance to find a solution in acceptable time. Therefore, in the following we try to give an analysis of the occurring problems and discuss both solved and open questions of the solution approach pursued by the (human) Karlsruhe Brainstormers. Two main aspects have to be solved: one is the reduction of the complexity, the other is the question of Reinforcement Learning in distributed systems.

2.1 Reduction of the complexity

In principle, there are three sources of complexity: the number of states, the number of actions and the number of decisions that have to be taken until a trajectory is (successfully) finished. One idea behind RL is to handle complex state spaces by learning on experiences occurring during interaction with the world, thereby considering only the relevant parts of state space. A further reduction might be obtained by learning on features rather than on the state representation directly. This might turn the MDP into a POMDP (partially observable MDP) and is a current research topic. The number of actions can be reduced by arbitrarily constraining the number of commands that the agent can choose. On one extreme, we might allow only a single action, and decision making becomes trivial, but we obviously pay for it by loosing quality of the solution. The trick therefore is to provide as many actions as might be useful for a good policy, but not more. On the level of basic commands it is very difficult to decide, which actions are actually required in a certain situation. However, if we allow whole sequences of actions, the job becomes easier: A player needs a sequence to intercept the ball, to dribble the ball and so on. Such a sequence is called a 'move' here. Each move has a defined subgoal. The concept of moves has the following two important aspects: It considerably reduces the number of actions available to the agent, and being itself a sequence of basic commands, it reduces the number of decisions made by the agent. As we will see below, a move itself can be learned by RL methods.

2.2 Reinforcement Learning in distributed systems

Applying the move-concept, the policy of an agent has to select between a couple of moves instead of basic commands - this can be interpreted as a 'tactical'

decision. In the soccer domain, the tactical decisions have to be done by each agent individually. This raises the problem of learning in distributed scenarios. Since all the players have a common goal, this is a cooperative multi-agent system. In the MDP-framework, the cooperation aspect can be modelled by giving each agent the same transition costs $c()$. The restricted communication ability causes a further difficulty: No agent knows what action is taken by its teammate. It can only observe the resulting state of the joint action. This scenario is sometimes called an 'independent learners' (IL) -scenario. In case of a deterministic environment, we recently proposed a distributed learning algorithm for a team of IL-agents [2]. An extension to stochastic domains is currently under development. This will provide the theoretical foundations for learning in a team of individually acting soccer agents. Some empirical results based on heuristic modifications of single-agent Q-learning have already been carried out (section 4.1).

3 Moves

A move is a sequence of basic actions, that transforms a current situation $s(0)$ into a new situation $s(t)$ some time steps later. The resulting situation is one of a set of terminal states \mathcal{S}^f , which might be either positive/ desired situations \mathcal{S}^+ or negative/ undesired situations \mathcal{S}^- . The move ends, if either a terminal state is reached $s(t) \in \mathcal{S}^f$, or the time exceeds a certain limit $t > t_{max}$.

For example, the move *intercept-ball* terminates if either the ball is within the player's kick range (\mathcal{S}^+) or if the agent encounters a situation, where it is no more possible for the player to reach the ball (\mathcal{S}^-).

3.1 Reinforcement Learning of Moves

Since each move has a clearly defined goal, the task is now to find a sequences of basic commands that does the job. This can be done either by conventional programming, or, as it is the case in our approach, by reinforcement learning methods. The general idea of RL is that the agent is only told, what the eventual goal of its acting is and which actions it might use (here: turn/ kick/ dash). Per time step, one command is selected depending on the situation. Learning here means to incrementally improve its decision policy such that the learning goal is fulfilled better and better. Here we apply Real-Time Dynamic Programming methods [1], that solve the learning problem by incrementally approximating an optimal value function by repeated control trials. Since the state space is continuos, a feedforward neural network is used to approximate the value function. For a detailed description of RL using neural networks, the reader is referred to [3].

Example: Learning to kick Finding a good kick routine is a very tedious job. In the RL framework, this job is done by the agent itself. It is provided with 500 parametrized instances of the kick command (direction is discretized in 100

steps, power is discretized in 5 steps) plus 36 instances of the turn command. This makes an overall of 536 actions, from which the agent can choose one per cycle. The learning goal is to find a sequence of this kicks and turns, such that eventually the ball leaves the player in a certain target direction with a certain target velocity. This target defines the 'positive' States, \mathcal{S}^+ . If such a state is reached 0 costs occur and the sequence terminates. A negative situation \mathcal{S}^- occurs, if the player loses the ball during a sequence. This results in maximum costs of 1. Since a reasonable optimization goal is to use as few commands as possible for a successful sequence, each intermediate transition causes low constant costs $c(s, u) = 0.002$ [3]. After about 2 hours of learning, the resulting policies

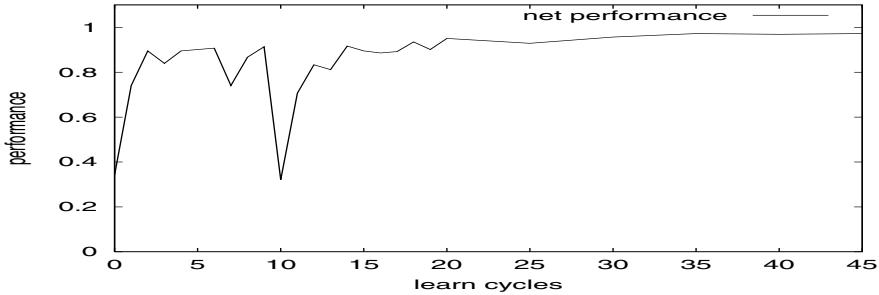


Fig. 1. An example of the performance during learning to kick the ball with maximal velocity 2.5 m/s. Each learning cycle comprises 2000 start states with different ball positions and velocities and the corresponding traversed trajectories. The net stabilizes after approximately 150000 individual updates and reaches a performance of approx. 95%.

were quite sophisticated - much more efficiently than in hand-coded heuristic kick routines, the agent learned to pull the ball back, turn itself (if necessary) and to accelerate the ball several times in order to yield high speeds (see also figure 1). It was able to learn to accelerate the ball to speeds up to 2.5 m/s. Altogether, three different neural nets were trained, one for target velocities up to 1.0 m/s, one for target velocities up to 2.0 and one for target velocities up to 2.5 m/s (each of them using 4 inputs, 20 hidden and 1 output neuron).

4 The tactical level

On the tactical level, each agent must decide among the following moves *intercept-ball*, *go in one of 8 directions*, *wait at position*, *pass ball to teammate* (10 choices), *shoot to goal in three variants*, *dribble in one of 8 directions*. This makes an overall of 10 actions in situations, where the agent has no ball and 22 actions in situations where the agent possesses the ball. No agent can win the game on its own; therefore, the agents must learn to cooperate and coordinate their behaviour. A

team strategy can only work, if every agent acts reliably and contributes to the common goal.

4.1 Reinforcement Learning of team strategies

Our initial experiments are carried out with 2 attackers against 1 or 2 defenders. The defenders have a fixed policy: they run to the ball and if they get it, they are directly moved to the attackers goal, where they score. The task for the attackers is to score as fast as possible. If they do so, they are 'rewarded' by final costs of 0; if they loose the ball, they are punished by costs of 1. Every time step the attackers need until they score, they get a small constant cost of 0.002 (meaning that they can need up to 500 steps until their current policy looks as bad as a policy that loses the ball). At every time a new decision is required, the agent can choose between 13 moves if he has the ball (he has only one teammate for passing) and 10 moves if he is without the ball. In contrast to the approach proposed in [4], we used the complete continuous state information as input. Since this state information is continuous, we use neural networks to represent the value function. In the implementation finally used, each move uses an individual net to represent the costs that would occur if this move was actually applied in the respective situation. This allows to use individual features for every move, hence supporting generalization abilities. The following table shows the improvement achieved by a learned policy over an alternative greedy policy, which gets the ball and try to score then.

	greedy	RL trained
2 against 1	35 %	85 %
2 against 2	10 %	55 %

The greedy policy scores against 1 defender in 35% of the situations and against 2 defenders in only 10% of the situations. The 2 learning agents successfully learned to cooperate to score in most of the situations. In case of 1 defender, they score in 85% of the situations, and against 2 defenders they score in 55% of the situations. This is a fairly high rate of success, since in a real soccer game not to score does not mean to get a goal on your own, but just that your team has to regain the ball. We even observed such complex cooperation patterns as 'double passes'!

5 Summary

The soccer domain can be modelled as a complex MDP. Its main properties are the high dimensional, continuous state space, the huge number of basic commands, huge number of policies the requirement for cooperative independent Reinforcement Learning and the problem of the partial observability of the state information.

The current Karlsruhe Brainstormers approach tries to tackle the complexity by using moves, which themselves are learned by RL methods and define the skills of an agent. Feedforward neural networks are used for function approximation

in continuous state spaces. To deal with the Multi-agent aspects, we pursue both the investigation of theoretically founded distributed RL algorithms plus the empirical/ heuristically motivated way of modified single-agent Q-learning. Still a lot of research questions are open, for example the dealing with partial observability of state information, the definition of theoretically founded and efficient distributed learning algorithms, the ideal representation of the value function, the search for appropriate features and the theoretical justification to use them, the automatic finding of appropriate moves. However, RL methods have already proven to be very useful in our competition agent.

5.1 Reinforcement Learning in the competition team

In the current competition version of our Brainstormer's agent all basic moves are learned by Reinforcement Learning, i.e. 1. a *kick* move which can accelerate the ball to put it with arbitrary velocity (0 to 2.5 m/s) in a desired direction 2. an *intercept-ball* move that effectively intercepts a rolling ball, taking the stochastic nature of the domain into account 3. an *dribble* move that allows to run without losing control over the ball, 4. a *positioning* move which reaches a particular position while avoiding collisions with other players, 5. a *stop-ball* move specialised in stopping high speed balls, 6. a *hold-ball* move which keeps the ball away from an attacker. Nearly all of the fundamental command decisions are therefore done by neural network based decision making.

On the tactics level, the very promising results for the 2 against 2 attack play are currently not directly applicable in the competition agent. For the tactic level, currently a method is used that we consider to be an intermediate step to Reinforcement Learning: Each possible move is judged by both its usefulness (quality) and probability of success. The quality of a move is given by a simple priority ranking (PPQ-approach).

5.2 Acknowledgements

We would like to thank the CMU-Team for providing parts of the source code of their competition team. In our current agent, we make use of their world model.

References

1. A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, (72):81–138, 1995.
2. M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of International Conference on Machine Learning, ICML '00*, pages 535–542, Stanford, CA, 2000.
3. M. Riedmiller. Concepts and facilities of a neural reinforcement learning control architecture for technical process control. *Journal of Neural Computing and Application*, 8:323–338, 2000.
4. Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Buch Verlag, 1998.

Preliminary Studies of Dynamics of Physical Agents Ecosystems

Josep Lluís de la Rosa, Israel Muñoz, Bianca Innocenti, Albert Figueras, Miquel Montaner, Josep Antoni Ramon

Institut d'Informàtica i Aplicacions, Universitat de Girona & LEA-SICA
Lluis Santaló s/n, E-17071 Girona, Catalonia
{peplluis, imunoz, bianca, figueras, montaner, jar}@eia.udg.es

Abstract. This paper is a step forward from the agent ecosystems that Hogg studied [6]. We plan to extend these agents ecosystems to physical agents that interact with the physical world. The aim is to conceive algorithms for the choice of resources and to expand this work. Dynamics of choice in such ecosystems depends on pay-off functions that contain information about the real physical world. One contribution here is to formalise the choice of knowledge resources by including a consensus technique. The second contribution is to include diversity by means of physical agents and to analyse the emergent impact in terms of diversity and performance. Simulated soccer robots exemplifies all this.

1 Introduction

1.1 Consensus

There is extensive research in consensus and sensor fusion techniques, in which normally consensus is more focused on knowledge and sensor fusion in the acquisition of data, [1] and [3]. The idea is simple, when there are multiple sources of information or knowledge, there will possibly be contradictions that will require additional knowledge and techniques to obtain useful knowledge out of the contradictory data.

1.2 Physical Agents

According to Asada [2], physical bodies play an important role, when emerging complex behaviours in order to achieve goals in dynamic real world. The RoboCup Physical Agent Challenge provides a good testbed to see how physical bodies play a significant role in realising intelligent behaviour using RoboCup framework [7]. Our research focuses in dynamical physical agents, in which the term *dynamics* is a way to model the transient and steady behaviours of the physical bodies of agents [8].

1.3 Universal Information Ecosystems

Hogg and Hubermann [6] studied the dynamics of choice in communities of agents, with different delays to access information. This work defined the resource choice as a preference function of resource 1 vs. resource 2. It is defined in terms of difference of the performance obtained by applying the information from resource 1 and 2. The uncertainty (σ) is modelled as a typical deviation on performance to decide when clearly resource 1/2 leads the agent to a better performance.

$$\rho = \frac{1}{2} \cdot \left(1 + \operatorname{erf} \left(\frac{G_1(F(f_1)) - G_2(F(f_2))}{2\sigma} \right) \right) \quad (1)$$

$$\frac{df_{rs}}{dt} = \alpha (f_s^{\text{type}} \rho_{rs} - f_{rs}) \quad (2)$$

$F(f_i)$ is a reward function related to the resource usage. G has access to $F(f_i)$ with different delays. α is the rate at which agents re-evaluate their resource choice. f_{rs} is the rate of agents of type s that at t prefer resource r , also called *population* for r,s

1.4 Work to be Developed

All these concepts have been applied to a team of agents, implemented in simulation using JavaSoccer. A set of agents, with different physical features, will play against a benchmark. Each player at any time takes a decision based on the rules, coded in 5 rule sets or roles, that goes through a consensus process, to reduce conflicts. Consensus parameters are related to the evolution of the different populations shown in [6].

2 Description of the Benchmark

Implicit opponent is a generalisation of benchmarks that were proposed by Jeffrey Johnson [5]. Its features allow having a good reference benchmark so that all existing teams can be evaluated/compared in a robust and general way. This benchmark has two main features. The first feature is an angled field, this causes the ball to fall down, in case it is not properly controlled, this is a challenging difficulty. The second feature is the presence of static obstacles. The robot can not easily move the ball to the opponent goal and it hinders the player to regain the ball [9].

3 Realisation of Decision Making

Five roles have been coded using fuzzy sets. These roles propose actions with a certainty associated, considering the position of the ball and the position of the player,

which is making the decision. These roles are: goalie (G), defender (D), right/left mid-fielders (RM/LM) and forward (F). Each role has a number of actions associated, and some roles share one or more actions. Actions have to go through a consensus process, in order to reduce conflicts, in case two agents decide to do the same. The procedure is the following: initially each robot takes the decision with the highest certainty associated after the revision process done by the consensus technique. In case two agents want to take the same decision, the one with the highest certainty wins. The revision process is based on two parameters, *Prestige* and *Necessity* [4], that modify the certainty (φ).

Prestige (P) performs a linear transformation over φ

$$\varphi' = P(\varphi) = P \cdot \varphi \quad (3)$$

Then, *Necessity* (N) performs a non-linear transformation over φ'

$$\varphi'' = N(P(\varphi)) \quad (4)$$

4 A Consensus Based Choice Function in Ecosystems

In this paper Hogg's work is used to increase or diminish the preference of every agent for each of the resources. In this case resources (r) are roles and agent diversity (s) is created through different physical features for each robot, instead of using delays to evaluate pay-off. Equation (1) has been modified in order to deal with 5 resources. This equation is a first approximation and it will be improved in the future.

$$\rho_{rs} = \frac{G_{rs}}{\sum_{r=1}^R G_{rs}} \quad (5)$$

The agent s has a performance G_r for each resource r , which is calculated in a temporal window of T_w units of time. This index is calculated taking into account several aspects: set points accomplishment, player situation on the field depending on the role, goals received/scores, dribbling ability, ability to regain the ball. Each rewarding action has a value associated depending on the position on the field, where the action has taken place. For example, it is far more important to regain the ball at the penalty area than recovering it at midfield. f_s changes according to this equation:

$$\frac{df_s^{type}}{dt} = \gamma(\eta_s - f_s^{type}) \quad (6)$$

$$\eta_s = \frac{\sum_{r=1}^R G_{rs}}{\sum_{s=1}^S \sum_{r=1}^R G_{rs}} \quad (7)$$

These equations were also proposed by Hogg [5], but also they have been modified. η_s increases population values to agents that perform better globally.

In order to combine the idea of preference for one role and the population values, the value of f_{rs} is used to assign the parameter *Prestige* (P) for revision in consensus. (k is a constant value).

$$P_{rs} = \frac{1}{1 + \exp\left(\frac{\frac{1}{N_players \cdot N_roles} - f_{rs}}{k}\right)} \quad (8)$$

The values of *Necessity* are computed depending on the resources/roles use. It aims at having a team with balanced resource usage, this introduces some kind of continuous excitement in the system. The way the value of *Necessity* is computed as follows: (p_r is the fraction of resource r usage and δ is a constant value)

$$N_r = \exp(-\delta \cdot \frac{p_r}{f_{rs}}) \quad (9)$$

5 Experiments

Using the ideas presented so far, the experiments performed aim at showing how a system, made up of different types of physical agents, can self-organise/stabilise (stable means show persistence on one role) depending on their fitness of their physical features for each role. Also the experiments try to show how a diversity of physical agents help to stabilise the system vs. a team of identical agents under some circumstances

5.1 Experimental Conditions

The experiments were performed under the same conditions:

Time	Tw	Δt	k	γ	δ	Field angle
12 h	100 sec	0.01-0.05	0.02	1	1	8°

Table 1. Parameter Values

Time is JavaSoccer simulation time (sample time 100 ms). The differential equations 4 and 7 have been approximated to a discrete formulation .Initially Δt is a very small (0.01), after some time this value increases progressively until 0.05. This change allows the agents to go through many different situations initially, until they “decide” to settle in one role. The used agents have the following features:

Agent	Radius (cm)	Translation Velocity (m/s)	Rotational Velocity (rad/s)	Kick Speed (m/s)
0	0.08	0.15	3.28	0.85
1	0.04	0.40	8.28	0.2
2	0.07	0.20	4.28	0.65
3	0.06	0.30	6.28	0.5
4	0.05	0.35	7.28	0.3

Table 2. Robot Features

5.2 Experimental Results

Agent	Final Role	Dominant Population	% Final Role Usage	% Dominant Population	Stable Roles ?
0	D/G	D/G	50/100	75/98	Y/Y
1	F/No Role	F/LM	95/--	58/60	Y/N
2	G/LM	G/LM	100/90	98/63	Y/Y
3	LM/RM	LM/RM	95/90	66/63	Y/Y
4	RM/No Role	RM/D	90/--	48/36	Y/N

Table 3. Results for heterogeneous/homogeneous agents

These results are specific for one run, although these results can be obtained in many runs. As it can be seen from Table 3 heterogeneous agents tend to self-organise taking over each one a role. There is a very interesting aspect to notice: in logged data agents 1 and 4 seem to take a role, but after some time they finally settle in a different role. The reason for this sudden change is the better fitness of agent 1 for forward role. Agent 1 presents an excellent dribbling ability. Agents 3 and 4 have a quite good dribbling ability but they tend to regain very well balls at mid-field. For defensive roles it can be seen that agent 2 is the best for goalie, this is because agent 2 is big but at the same time is faster than 1. Agent 0 has a relatively small role usage, the reason for that problem is the small number of actions available for all roles. This problem leads to many conflicts among agents and it is manifested in defensive roles. For the experiments performed over the identical agents (agent type 3), results show that some agents tend to take a role, but there are other agents that do not seem to find a role, taking continuously rules from different roles. The reason for this problem seems to be the difficulty to find an agent fitted for defensive and forward roles, under the extreme circumstances that the experiments have been performed. In this case seems that diversity helps to stabilise the system, although this is not always true. Some experiments performed with a smaller angle show that both multi-agent systems tend to self-organised, taking each agent a different role.

6 Conclusions

We have been able to expand Hogg's work [6] to physical agents, although some changes were needed to adapt these ideas to physical agents. Also we have been able to link the dynamics of choice and the evolution of populations to a set of parameters that features the consensus technique here presented in a successfully way and finally we have introduced a new type of diversity based on physical features.

7 Future Work

All these ideas will be implemented in the future using real robots and the benchmark proposed. These robots, although physically identical, will have a different dynamical behaviour, achieved using different types of control. Reward assignment, equations, and so on and forth will be improved in the future.

8 Acknowledgement

This work is partially funded by projects TAP98-0955-C03-02, TAP99-1354-E, of the Spanish Research Foundation CICYT, and the special action 2000ACES00018.

References

1. Abidi M.A and R.C. González, Data Fusion in Robotics &Machine Intelligence, Academic Press, pp:7-135, 1992.
2. Asada M., Kuniyoshi Y., et al. The RoboCup Physical Agent Challenge, First RoboCup Workshop in the XV IJCAI-97 International Joint Conference on Artificial Intelligence, pp.51-56, 1997.
3. K. Chi Ng and B. Abramson, Consensus Diagnosis: A Simulation Study, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, n° 5, September/October 1992.
4. de la Rosa J. Ll. and Delgado A. Fuzzy Evaluation of Information Received from other Systems, Proceedings of the IPMU'92, pp. 769-778, Palma de Mallorca, July 6-10, 1992.
5. Johnson J., de la Rosa J.Ll. and Kim J.H., "Benchmark Tests in the Science of Robot Football" Proceedings IEEE of Mirosoft-98, Paris 1998.
6. Hogg T. and Huberman B. A., Controlling Chaos in Distributed Systems, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 6, November/December 1991.
7. Kitano H., Veloso M., et al., The RoboCup Synthetic Agent Challenge 97, XV IJCAI-97 Int. Joint Conference on Artificial Intelligence, Vol 1, pp.24-29, Nagoya, August 1997.
8. Oller A, de la Rosa J.Ll., and del Acebo E., DPA2: Architecture for Co-operative Dynamical Physical Agents, MAMAAW '99, Valencia June 1999.
9. de la Rosa J. Ll., Muñoz I., Duhaut D., Johnson J., Experimental Basis for Benchmarking RoboCup, Proceedings of the first spanish-portugese Workshop on Physical Agents, pp87-101, Tarragona, September 2000.

RoboCup-Rescue Disaster Simulator Architecture

Tomoichi Takahashi¹ Ikuo Takeuchi² Tetsuhiko Koto²
Satoshi Tadokoro³ Itsuki Noda⁴

¹ Chubu University, Matsumoto, Kasugai, 487-8501, JAPAN ttaka@isc.chubu.ac.jp

² University of Electro Communication, Chougaoka 1-5-1, Chofu, 182-8585, JAPAN
{nue@nue.org, koto@takopen.cs.uec.ac.jp}

³ Kobe University, Rokkodai-cho, Nada-ku, Kobe, 657-8501, JAPAN
tadokoro@octopus.cs.kobe-u.ac.jp

⁴ Electrotechnical Laboratory, Tukuba 305-8568, Japan noda@etl.go.jp

Abstract. *RoboCup-Rescue* project aims to simulate large urban disasters and rescue agents' activities. The simulator must support both simulation of heterogeneous agents such as fire fighters, victims' behaviors and interface to disaster's environments in the real world. *RoboCup-Rescue* simulator is a comprehensive urban disaster simulator into which a new disaster simulator or rescue agents can be easily plugged. In this paper, the simulator's specification, based on the Hanshin-Awaji Earthquake, and the simulator's architecture are described.

1 Introduction

RoboCup-Rescue addresses the problem of the simulation of a large-scale urban disaster and the rescue operation that ensues [1]. The initial stages of the *RoboCup-Rescue* initiative are concerned with building a simulation environment chiefly for research and competition, while the eventual goal is to support disaster management before and after a major catastrophe.

The characteristics of the RoboCup disaster simulation are a comprehensive disaster simulator by distributed computation, and a large-scale heterogeneous agent system. In this paper, the simulator's specification, based on the Hanshin-Awaji Earthquake, and the simulator's architecture are described.

2 Design from Nagata Ward Case Study

The *RoboCup-Rescue* project will support the functions for the disaster and rescue simulations. Our scenario of rescue operations and requirements for the simulator are as follows:

1. A large urban disaster is composed of various kinds of disasters such as fire, building blockage, road blockade, etc. The simulators for these disasters have been developed independently in each field. *RoboCup-Rescue* simulator can be plugged into these component disaster simulators.

2. At a large disaster, fires break out at the same time. Traffic jams caused by civilian's behavior may have an effect on rescue operations [2]. *RoboCup-Rescue* simulator must simulate the rescue operations at the civilians' action level.
3. Large earthquakes take place all over the world every few years on average. In the last five years, Kobe-Awaji, Los Angeles, Turkey and Taiwan suffered from large earthquakes. *RoboCup-Rescue* simulator will simulate each disaster by replacing geographic data and disaster models.

Necessary conditions for *RoboCup-Rescue* project are considered by investigating disasters in Nagata Ward. The area was 11.47 km^2 and one of most damaged areas of the Hanshin-Awaji Earthquake. On October 1, 1994, 130,466 people (53,284 households) lived there.

simulation period : Rescue activities start when earthquakes occur, and are to be continued for years. The purposes of the activities change as time passes. The purpose of rescue activities at the first stage is saving victims without any aid from outside. The period to be simulated is set to the first 72 hours, considering that after that period the survival rate decreases rapidly.

rescue agents : When earthquakes occur, only local rescue agents will perform the first rescue actions. There were 7 rescue agents at Nagata fire offices. The order of rescue agents is set in this order.

space resolution : Representing disaster situations or rescue activities requires the display of items the size of cars. In order to do that, GIS (Geographic Information System) data is maintained with a resolution of 5 m mesh. The area of 1.5 km^2 centered on JR Nagata railway station is selected to be the target for the prototype system from the amount of GIS data.

3 Prototype system

3.1 Architecture of prototype system

Fig. 1 shows the overview of the *RoboCup-Rescue* prototype simulator. It is composed of a number of modules which communicate with each other using a protocol based upon UDP/IP. The modules are one kernel, one GIS, agents, component simulators, and 2D/3D (dimensional) viewers.

Kernel controls the simulation by receiving and sending messages between other modules. At the beginning of the simulation, the kernel receives the initial configuration of the simulated world from the GIS module. At every simulation step, the kernel sends to agents sensory information, and receives their action commands. The kernel checks the validity of actions. For example, when a fire brigade agent extinguishes a fire, it cannot use water beyond the amount provided by fireplugs. To disaster simulators, the kernel sends changes in the world, and receives from them the result of simulations.

Agent module controls an intelligent agent that decides its own actions according to situations. Civilians, fire fighters, ambulance teams and so on are agents. The agents are virtual entities in the simulated world.

Component simulators simulate the effect of disasters by their own methodologies. Building and road blockage simulators calculate how the buildings are broken, and the roads are covered with the broken buildings. A fire simulator calculates how the fire spreads, and how fires are extinguished by fire fighting.

GIS module provides the configuration of the world, where roads run and where buildings and individuals are located at the beginning of the simulation ⁵. The other role is to manage the spatial temporal data representing transition of the disaster and the rescue operations during simulation steps.

Viewers visualize the result of simulation. Global information is displayed in a form of 2D scenes, and local scenes are displayed as 3D images.

3.2 World Model

The world model of prototype system is consisted of the following objects:

Building : a building which may collapse, obstruct a road, or bury persons.

Civilian : a family or a person which has encountered the disaster.

Car : a car which a civilian agent drives.

Fire brigade : a team that extinguishes fires.

Fire station : a station that can order fire brigade agents.

Ambulance team : a team that rescues persons buried alive and carries injured persons to hospitals.

Ambulance center : a center of ambulance teams that controls the team.

Police force : a team that opens roads obstructed by collapsed buildings.

Police office : a center of police forces that orders the police.

3.3 Protocols between modules

Table 1 shows protocols that specify communications among modules. In the prototype system, one simulation cycle is one second of computer time and it corresponds to one minute in the real world time. The simulation cycle follows the following steps:

1. At each cycle, the kernel sends **sensory information** to each agent module. This information consists of objects that the agent can see at that time. They are within a certain radius around the agent.
2. Each agent module decides what actions the individual should take, and send it to the kernel as **command**.

⁵ The prototype system uses a GIS data format based on KIWI format that is discussed in Intelligent Transport Systems Working Group of ISO as a standard format for car navigation systems.

3. The kernel gathers all messages sent from agent modules, and broadcasts them to the component simulators.
4. The component simulators individually compute how the world changes its internal status. These results are sent back to the kernel.
5. The kernel receives the simulation results of the component simulators within a certain time, and integrates them. The kernel broadcasts the integrated result and forward simulation clock.
6. The viewers request the GIS to send updated information of the world, and display visually the information with statistical data.
7. The GIS keeps track of the simulation results, and the changes in the simulated world.

At step 2, the agent modules decide the agent's behavior according to their objectives, such as to search for the victims, to rescue them, to evacuate them to safe places, to provide them with some food, etc. A local language is defined for the agent-agent communication. Agent act command includes the followings: *extinguish Target*, *rescue Target*, *load Target*, *unload Target*, *open Target*.

Table 1. commands in agent's protocol.

Command	Information Transfer	Function
Initialization		
init	agent → kernel	Initialization of agents
Action Commands		
move	agent → kernel	Motion of agent body
act	agent → kernel	General term for disaster mitigation action, implemented ones are extinguish, rescue, load, unload, open.
say	agent → kernel (agent)	Auditory information transmission
tell	agent → kernel (agent)	Via transmission line
Sensory information		
see	agent ← kernel	Visual information acquisition
hear	agent ← kernel (agent)	Auditory information acquisition
listen	agent ← kernel (agent)	Via transmission line

4 Evaluation of prototype system

The prototype system is tested under the following environments.

machine environments : Modules in prototype system are distributed throughout seven computers. Their processing power is at the Pentium III 700 Mhz level, and they are connected within a 100 M sub-network.

GIS test worlds : Four different sizes - 1/1000, 1/100, 1/10 and 1/1 scale - are tested. 1/1 scale model is a square area 2,217 m on a side. There are 9,357 houses and the road network consists of 9,776 links and 9,143 nodes.

Fig. 2 shows a snapshot of a 2D viewer at the 1/10 scale and the 1/1 scale model. The spots are buildings and the lines are roads. The color of buildings is green, and buildings turn red when they burn. The points on the lines represent agents. The number of civilians is 934 at 1/10 scale. The real number, which is 100 times this number, is too large to use at this time.

The prototype shows our formulation manages a comprehensive disasters simulation. However, it becomes clear that there are following problems:

agent How much of the world should an agent know ? A civilian agent knows his/her neighborhood better than the area far away, while a fire brigade agent knows the simulated area equally with a rough resolution.

component simulator How does the component simulator finish its simulation within a simulation step for large cities ? Especially, a newly plugged simulator does not know how much area it should calculate or how much time it takes to calculate.

kernel All data is communicated among modules via kernel. The amount of data that the kernel must handle increases as the number of modules or the size of area becomes larger. One kernel becomes inevitably unable to handle the data as the size becomes larger or the simulation level becomes finer.

5 Discussion concerning a new *RoboCup-Rescue* simulator

RoboCup-Rescue simulator provides not only a multi-agent system but also a challenge problem for distributed interactive simulations. Taking into account other simulator systems [3] or standardizations [4], topics such as data distribution for a large city, evaluation standard for rescue competition, are being discussed for the next simulation system.

The authors would like to thank other members of the *RoboCup-Rescue* Technical Committee, and three universities (Tokyo Institute of Technology, Nagoya Institute of Technology, University of Southern California), which demonstrated their rescue teams At RoboCup 2000.

References

1. S. Tadokoro, et al., "The RoboCup-Rescue Project: A Robotic Approach to the Disaster Mitigation Problem," *Proc. of SMC*, pp. 24-29, 1999.
2. J. L. Casti, "Would-Be Worlds," John Wiley & Sons, Inc. 1996
3. I. Noda, "Modular Simulator: draft of new Simulator for RoboCup," *WorkShop (ABS-4) Notes, ijcai 99*, pp. 154-159, 1999.
4. J. S. Dahmann, et al., "A Reusable Architecture for Simulations," *COMMUNICATIONS of the ACM*, Vol. 42, No. 9, pp. 79 - 84, 1999.

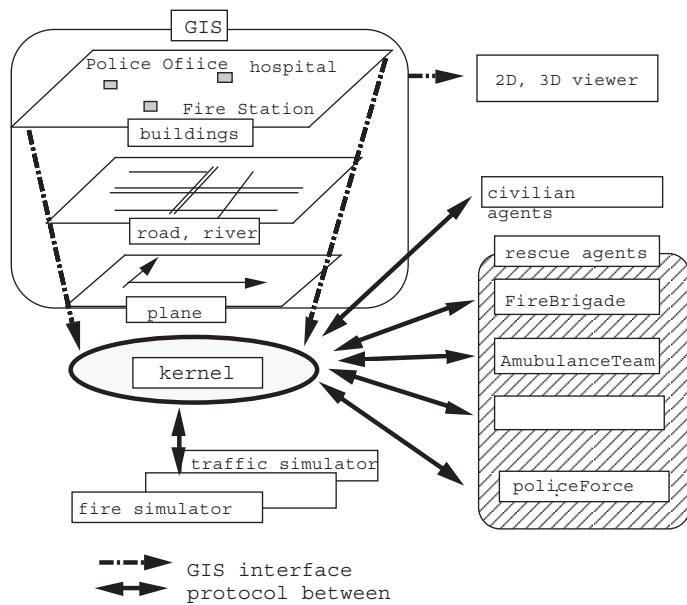


Fig. 1. Overview of prototype rescue simulator

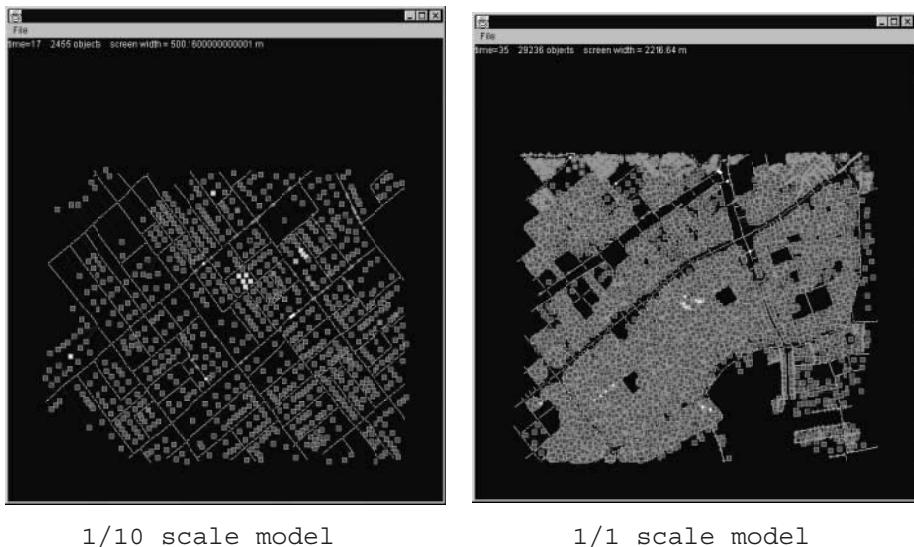


Fig. 2. 2D viewer of 1/10 scale and 1/1 scale.

Improvement Continuous Valued Q-learning and its Application to Vision Guided Behavior Acquisition

Yasutake Takahashi¹, Masanori Takeda³, and Minoru Asada¹

¹ Dept. of Adaptive Machine Systems,
Graduate School of Engineering Osaka University, Suita, Osaka 565-0871, Japan
² Nara Institute of Science and Technology, Japan

Abstract. Q-learning, a most widely used reinforcement learning method, normally needs well-defined quantized state and action spaces to converge. This makes it difficult to be applied to real robot tasks because of poor performance of learned behavior and further a new problem of state space construction. We have proposed Continuous Valued Q-learning for real robot applications, which calculates contribution values to estimate a continuous action value in order to make motion smooth and effective [1].

This paper proposes an improvement of the previous work, which shows a better performance of desired behavior than the previous one, with roughly quantized state and action. To show the validity of the method, we applied the method to a vision-guided mobile robot of which task is to chase a ball.

1 Introduction

Reinforcement learning has been receiving increased attention as a method with little or no a priori knowledge and higher capability of reactive and adaptive behaviors for robots through the interactions with their environment[2]. However, Q-learning, a most widely used reinforcement learning method, normally needs well-defined quantized state and action spaces to converge. This makes it difficult to be applied to real robot tasks because of poor performance of learned behavior and further a new problem of state space construction.

We have proposed Continuous Valued Q-learning for real robot applications[1]. The proposed method interpolates continuous values between roughly quantized states and actions. This contributes to realize smooth motions with much less computational resources. This paper proposes an improvement of the previous work through the discussions what are problems in the previous one and how they are solved. The proposed method obtained the better performance of desired behavior than the conventional real-valued Q-learning method, with roughly quantized state and action. To show the validity of the method, we applied the method for a vision-guided mobile robot of which task is to chase the ball.

2 Continuous Valued Q-Learning

2.1 State Representation and Action Interpolation

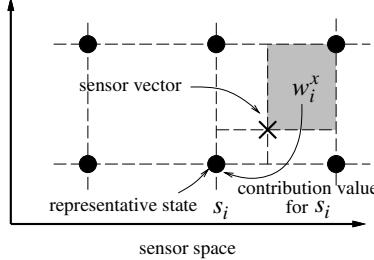


Fig. 1. Calculation of contribution value w_i for the representative state s_i in the case of two-dimensional state space

A basic idea for continuous value representation of the state, the action, and the reward in Q-learning is to describe them as contribution vectors of the representative states, actions, and rewards, respectively. First, we quantize the state/action space adequately. Each quantized state and action can be the representative state s_1, \dots, s_n and the representative action a_1, \dots, a_m , respectively. The state (action) representation is given by a contribution vector of the representative state $\mathbf{s} = (w_1^s, \dots, w_n^s)$ ($\mathbf{a} = (w_1^a, \dots, w_m^a)$). A contribution value indicates the closeness to the neighbor representative state (action). The summation of contribution values is one.

There are several methods to calculate a contribution value, and we select a following one because of the simplicity of the calculation. For the readers' understanding, here we assume that the N -dimensional sensory information directly construct the N -dimensional state space, and the motor space has M -dimensions. The robot perceives the current sensory information as a state vector $\mathbf{x} = (x_1, x_2, \dots, x_N)$, and executes motor command $\mathbf{u} = (u_1, u_2, \dots, u_M)$.

First, we tessellate the state space into N -dimensional hyper cubes¹. The vertices of all hyper cubes correspond to the representative state vectors $\mathbf{x}^i = (x_1^i, x_2^i, \dots, x_N^i)$ $i = 1, \dots, n$ (here, n denotes the number of the vertices), and we call each vertex the representative state s_i . The contribution value w_i^s for each representative state s_i when the robot perceives the input $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is defined as the volume w_i^s of the box diagonal to the state s_i . Mathematical formulation is given by

$$w_i^s = \prod_{k=1}^N l_i(x_k) \quad , \text{ where } l_i(x_k) = \begin{cases} 1 - |x_k^i - x_k| & \text{if } |x_k^i - x_k| \leq 1 \\ 0 & \text{else} \end{cases} . \quad (1)$$

¹ the unit length is determined by normalizing the length of each axis appropriately

Fig.1 shows a case of two-dimensional sensor space. The area w_i^x is assigned as a contribution value for state s_i . Thus, the state representation corresponding to the input \mathbf{x} is given by a state contribution vector $\mathbf{w}^s = (w_1^s, \dots, w_n^s)$.

Similarly, the representative action vectors $\mathbf{u}^j = (u_1^j, u_2^j, \dots, u_M^j)$ $j = 1, \dots, m$, and the representative action a_j are given. When an action contribution vector $\mathbf{w}^a = (w_1^a, \dots, w_m^a)$ is given from the current policy π of the agent, the mapping to the motor command is defined as follows:

$$\mathbf{u} = \sum_{j=1}^m w_j^a \mathbf{u}^j. \quad (2)$$

2.2 Extension to the Continuous Space

According to the state and the action definition mentioned above, we modify the standard Q-learning algorithm as follows:

When the agent goes around its environment, it has a certain policy π , which returns a representative action $a \in A = \{a_1, a_2, \dots, a_m\}$, for each each representative state $s \in S = \{s_1, s_2, \dots, s_n\}$. That is:

$$a = \pi(s). \quad (3)$$

Using this policy π , the agent calculates the contribution vector of the representative action as follows:

$$w_j^a = \sum_{i=1}^n w_i^s f(s_i, a_j, \pi), \text{ where } f(s_i, a_j, \pi) = \begin{cases} 1 & \text{if } \pi(s_i) = a_j \\ 0 & \text{else} \end{cases}. \quad (4)$$

The Q -value when executing the representative action a_j at the representative state s_i is denoted by $Q(s_i, a_j)$. A Q -value at any state with policy $\pi(\mathbf{x}, \pi)$ is given by:

$$Q(\mathbf{x}, \pi) = \sum_{i=1}^n w_i^s Q(s_i, a), \quad \text{where } a = \pi(s_i). \quad (5)$$

Given the representative state s_i , the representative action based on the optimal policy π^* is calculated by:

$$a_k = \pi^*(s_i), \quad \text{where } k = \arg \max_j Q(s_i, a_j) \quad (6)$$

The state value function $V(\mathbf{x})$ is calculated by:

$$V(\mathbf{x}) = \sum_{i=1}^N w_i^s V_i, \quad \text{where } V_i = \max_j Q_{i,j}. \quad (7)$$

Then, the Q value when choosing an policy π at the current state \mathbf{x} , and transiting the next state \mathbf{x}' given reward r is updated by:

$$Q(s_i, a_j) \leftarrow Q(s_i, a_j) + \alpha w_i^s f(s_i, a_j, \pi)(r + \gamma V(\mathbf{x}') - Q(\mathbf{x}, \pi)) \quad (8)$$

where α is a learning rate and γ is a fixed discount factor between 0 and 1.

3 Improvements of the learning algorithm

3.1 Action Contribution Vector

The calculation of the contribution vectors of representative actions from the actual motor command may not work well as expected. We show a simple example. There are two representative states, i.e. “left” and “right”, and 3 representative actions, i.e. “fast”, “slow”, and “stop”. The goal state is the “right” one. The “fast” action is optimal at the “left” representative state, and the “stop” action is optimal at the “right” representative state. When the agent is at the middle position between two representative state, the agent may calculate the “slow action” by the interpolation between the optimal actions. Even if the agent execute the “slow” as the result of action interpolation, the Q-values for “slow action” will be reinforced on the both representative states. Since the “slow” action is not the optimal action at the both state, this reinforcement causes a wrong effect.

To avoid this situation, we use the contribution vectors of representative actions not calculated from the actual motor command, but based on the agent’s current policy π (Equations (5) and (8)). In the last example, if the agent execute the “slow” as the result of action interpolation of “fast” and “stop”, the “fast” and “stop” action will be reinforced, and “slow” action will not.

3.2 State Value Function

The calculation of a state value function $V(\mathbf{s})$ in the previous version[1] was:

$$V(\mathbf{x}) = \sum_{i=1}^N \sum_{j=1}^M w_i^s w_j^{a*} Q_{i,j}. \quad (9)$$

This means that a state value function at the state \mathbf{s} should be the Q value at the state taking the action a^* based on the optimal policy. However, this is not a good idea because the state values except for the representative states are not properly estimated unless the optimal action at all states are the same one.

Let us consider a case in which the optimal representative action at a representative state differs from one at the other representative state. The contribution of the each representative action w_j^{a*} has a mean value between the 0 to 1. The estimated state value at each representative state $w_j^{a*} Q_{i,j}$ is always less than the state value at the representative state $V_i = \max_j Q_{i,j}$. Then the estimated state value at the middle state is underestimated. This phenomena is not appropriate for the updating the Q-table. We should use the equation(7).

3.3 Q value Estimation

The Q value interpolation in the previous version was:

$$Q(\mathbf{s}, \mathbf{a}) = \sum_{i=1}^n \sum_{j=1}^m w_i^s w_j^a Q_{i,j} \quad (10)$$

As we mentioned in previous subsections, it is not a good idea to use the action contribution vector in order to calculate or update the Q -table. Therefor, we changed the definition of the state value function from equation (10) to equation (7), and the equation of Q value estimation like (5), or the agent may underes-timate the Q value than the state value $V(x)$ if it takes the optimal policy.

3.4 Update Equation

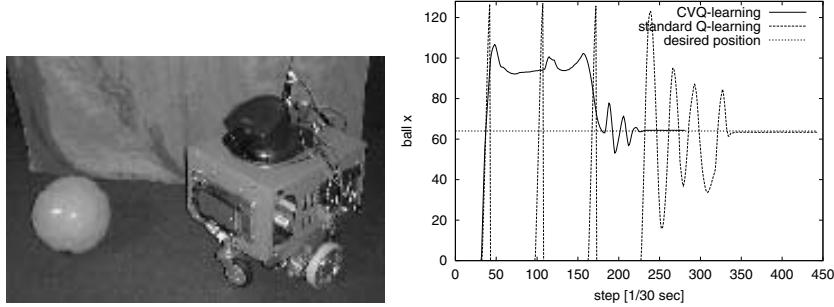
The equation of update Q value in the previous version was:

$$Q_{i,j} \leftarrow Q_{i,j} + \alpha w_i^s w_j^a (r + \gamma V(s') - Q_{i,j}). \quad (11)$$

This updates $Q_{i,j}$ even if the estimation is perfect. This phenomena does not lead to the convergence, but causes the divergence of learning. We should use the equation(8).

4 Experimental

In order to show the validity of the proposed method, we apply the method to a mobile robot of which task is to chase a ball, one of the vision-guided behavior acquisition. Fig.2(a) shows a picture in which the mobile robot we have designed and built and the ball to chase are shown. A simple color image processing (Hitachi IP5000) is applied to detect the ball area in the image in real-time (every 33ms).



(a) A mobile robot and a ball

(b) Step responses with/without our method

Fig. 2. Experiment

The robot has a TV camera of which visual angles are 35 degs and 30 degs in horizontal and vertical directions, respectively. The camera is tilted down 23.5

degs to capture the ball image as large as possible. The image area is 128 by 110 pixels, and the state space is constructed in terms of the centroid of the ball image. The driving mechanism is PWS (Power Wheeled System), and the action space is constructed in terms of two torque values to be sent to two motors corresponding to two wheels. These parameters of the robot system are unknown to the robot, and it tries to estimate the mapping from sensory information to appropriate motor commands by the method.

The state and action spaces are two-dimensional ones, and tessellated into 5 by 5 and 3 by 3 grids, respectively. The learning rate α and the discount factor γ are 0.05 and 0.7, respectively. The parameters are constant during the learning.

Action selection during the learning was random and the goal state is a situation that the robot captures the ball region at the center of the image and its size is pre-specified value so that the ball is located just in front of the robot (about 50cm apart). In other words, the goal state is that the coordinates of the centroid of the ball region is (64,55).

To show the efficiency of the exploration and smoothness of the motion, we compare the results with standard Q-learning based on the quantized state and action spaces. We show the difference in the step responses of the robot behavior after the learning. The ball is positioned 3m far from the robot. Fig.2(b) shows the step responses, where the behavior based on the proposed method successfully reaches the goal state with smaller error and smoother motion than the standard Q-learning method. The standard Q-learning method sometimes lost the ball and oscillated left and right before reaching the goal state.

Acknowledgments

We would like to thank Shoichi Noda (Displays, Hitachi Ltd., Japan) for fruitful discussions about modification of the CVQ-learning.

References

- [1] Y. Takahashi, M. Takeda, and M. Asada. Continuous Valued Q-learning for Vision-Guided Behavior Acquisition. *Proceeding of the 1999 IEEE International Conference on Multi-sensor Fusion and Integration for Intelligent Systems*, 255–260, 1999.
- [2] J. H. Connell and S. Mahadevan, editors. *Robot Learning*. Kluwer Academic Publishers, 1993.

Recognizing Formations in Opponent Teams

Ubbo Visser, Christian Drücker, Sebastian Hübner,
Esko Schmidt, Hans-Georg Weland

TZI - Center for Computing Technologies
University of Bremen
D-28334 Bremen, Germany
visser@informatik.uni-bremen.de
WWW home page: <http://www.virtualwerder.de>

Abstract. The online coach within the simulation league has become more powerful over the last few years. Therefore, new options with regard to the recognition of the opponents strategy are possible. For example, the online coach is the only player who gets the information of all the objects on the field. This leads to the idea determine the opponents play system by the online coach and then choose an effective counter-strategy. This has been done with the help of an artificial neural network and will be discussed in this paper. All soccer-clients are initialized with a specific behavior and can change their behavior to an appropriate mode depending on the coach's commands. The result is a flexible and effective game played by the eleven soccer-clients.

1 Introduction

Our team is based on the sources that were released by the CMUnited99 team [10]. We decided to do so because it would have been to time consuming to reinvent all basic skills.¹ Instead, we focus on research w.r.t. high level functions which will hopefully lead to new ideas and results for the RoboCup community. Our long run plan is to use part of the provided functions from the CMU-client to construct a more sophisticated team with individual players.

Over the last few years several attempts have been made in learning of team behaviour. Similar approaches have been developed from numerous research groups. These studies have the focus on learning team behavior within the simulation and middle size league (see [11], [12], [13], [9]). Raines et al. [7], e.g., describe a new approach to automate assistants to aid humans in understanding team behaviours for the simulation league. This approaches are designed for the analysis of games, off-line after playing, to gain new experiences for the next games. Frank et al. ([4]) present a real time approach which is based on statistical methods. A team will be evaluated statistically but there is no recognition of team strategies.

While conducting our research for this project we obtained support from real-life soccer experts. In an interview, Thomas Schaaf, the manager of SV Werder Bremen pointed out the importance of the strategy recognition of the opponent team. While the

¹ We would like to give special thanks to the original authors

coach-client has been able to participate through analysis and control in real matches since 1998 [3], the idea of general strategic planning becomes possible. Like in real life matches, the coach is able to give strategic commands depending on the opponent's system and the current score. We presume that the performance of our team can be improved by analyzing the opponent's strategy.

2 Agents

The Virtual Werder team consists of individual players which have different behaviors.

Players: 22 types of players have been developed with a variety of characteristics to ensure the flexibility and variability of actions and reactions within a game. There are different types of forwards, defenders, mid-fielders and goal keepers. Therefore, our clients have the option to change their behavior from one style to another at any stage of the game. This concept is an important key feature to carry out changes in a formation while playing against teams that are switching between different play systems. We plan to use the online-coach, so that our clients have the ability to receive the messages, parse them, and change their behavior accordingly. On the other hand, the low-level skills of the agents are based on the sources provided by CMUnited99 [2]. These skills include functions to locate the ball, the other team members and the opponent players. In addition, methods to communicate via UDP-sockets, a parser for soccer server messages and other utilities such as the memory structures, have been used. Our clients are initialized with a certain behavior and the desired formation when connecting to the soccer server. Furthermore, they have the ability to choose other behaviors and switch to them immediately. These high-level functions include methods to carry out different defense systems such as man-to-man marking and zone defense. We also included new mid-level functions, e. g. finding a teammate able to catch a pass. Another new skill includes the player moving to a certain point on the field, while keeping track of the ball. This is accomplished through the *turn_neck* command. The characteristics of these 22 types of players are described in the Virtual Werder team description.

Coach: The coach observes the game continually, analyzes the formation of the opponent team at given points in time with an artificial neural network (ANN) ([5], [8]) and broadcasts an adequate counter formation to the players during the next interruption. The current evaluation takes place twice per second. The positions of the players serve as inputs for an ANN, that is trained with the formations most commonly played in our test games and the log-files (see section 3). In order to get a reliable impression only outputs with high ratings are used. Whenever the play mode switches to another state than *PLAY_ON*, the coach generates a message for his team. It instructs the players which formation the opponent is currently using and gives information about the appropriate counter attack.

3 Approach

We observed that teams in the last RoboCup-tournaments typically relied on their strategy and team formation and often didn't change it within a game. When changes were

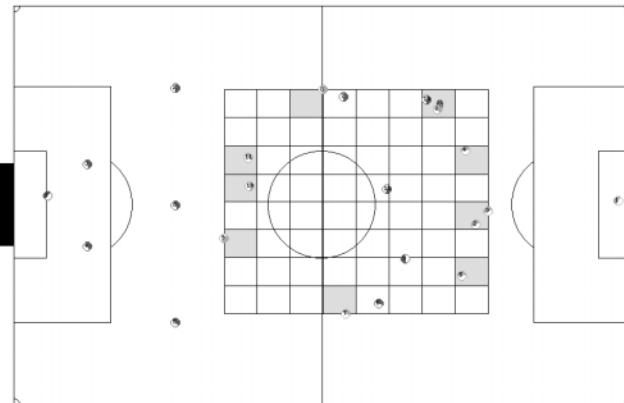


Fig. 1. Positions of opponent players and bounding box; marked cells define the input vector for the neural network.

made they depended on the score. A common practice was to switch from a offensive to a defensive formation if the team lead with more than n goals. On the other hand some teams remained on the same system regardless of the opponent's strategy. This is the point where the online-coach diagnoses what to do depending on the opponent's formation. The coach-client had to be fed with formations and information on how to analyze them. Part of the integrated knowledge was obtained from an interview with a real expert, the Werder Bremen's head coach Thomas Schaaf, other parts from literature, e. g. [1] and from games played in the last RoboCup tournaments. This knowledge can be used by the online-coach within the decision process.

The observation and analysis of the opponents team is processed in several steps which are described below. The CMUnited99 sources provide the communication with the soccer server and the parsing of messages. Furthermore, it supplies the coach with collected information in data structures which are easily accessible. This information is then prepared and will act as an input vector for an ANN. In order to prepare the data we first have to decide which variables should be used. Our model consists of 64 binary input variables. For this purpose a bounding box is placed around the positions of the opponent players. The box is currently divided into a grid of eight by eight cells which leads to an arrangement of 64 fields (Fig. 1). There must be at least one player inside a specific field with the value of this field set to 1. Otherwise, the value must be set to 0. The sum of the fields defines the input vector for the ANN. The network classifies the vector in one of 16 output classes, each representing a specific play system.

The network is implemented as a C-function² and is called to calculate the ratings at 16 possible output neurons which represent the opponent's formations (section 2). If the output neuron with the highest value exceeds a demanded threshold, this class will be chosen as the result of this function. This result and other information in addition, e.g. the size and position of the bounding box or the current score of the game, deter-

² snns2c by Bernward Kett was used to transform the trained network from an internal SNNS representation to a usable C-function [14].

mines the appropriate counter formation. During the next interruption of the game, this information is broadcasted to the team.

Training results: Our coach uses an ANN to analyze the formation of the opponent's team. The network itself was trained to recognize 16 different formations. The coach-client uses this knowledge about the classified formation to evaluate a proper counter-attack. Soccer formations are typically noted as a combination of defense, mid-field and forward players, e.g. a 5-3-2 represents a team with five defense players (and a goalie), three players in the middle field and two forward players. However, there are some special systems that do not fit into this pattern, these are referred to by name, e.g. the Catenaccio system [1].

We developed a tool called "ExportPlayer" to obtain formation examples for our network. This program is based on the log-player and takes automatic snapshots of existing log-files. It extracts the positions of the players (not including the goal keeper) and normalizes the coordinates to a grid of eight by eight cells. Cells labeled '0' do not contain players, cells labeled '1' contain at least one player. The ExportPlayer returns a pattern file which contains the values of all input and output neurons to serve the ANN (see also team description). We used the Stuttgart Neural Network Simulator (SNNSv4.1) [14] to train the examples and to create the code for a feedforward-network. We used standard backpropagation as learning method with the learning parameter $\eta = 1.0$ and the maximum difference $d_{max} = 0.3$ between the teaching value and the output.

The choice of the threshold mentioned in section 3 is a very important factor for the efficiency of the online-coach. Table 1 shows the correlation between different thresholds which results in the amount of permitted input patterns and their correctness. The inquiry is based on 680 patterns obtained from log-files and test games. These have been previously classified by us. On ten separate occasions, these patterns were randomly divided into 612 training and 68 test sets ($\approx 10\%$) and were processed by the net. The average of these ten different results have been calculated for validity.

Due to the large quantity of test patterns (currently, a snapshot is made twice per second), the relatively high amount of rejections is not problematic in this environment. Furthermore, it is similar to a real soccer game where distinct formation occurs infrequently.

threshold	classified	classified correctly
0%	100%	48.37%
80%	49.67%	65.53%
85%	41.34%	67.88%
90%	32.52%	69.30%
95%	20.10%	72.27%

Table 1. Relation between output threshold and correctness

4 Results

Our hypothesis was that we improve the performance of our team by detecting opponents strategies and obtaining the appropriate counter formation. Our test environment consists of our team and the teams of CMUnited99 and last years' Mainz Rolling Brains. We carried out ten games against each team with two different play systems. We decided that the formations 5-4-1 (defensive) and 3-4-3 (offensive) were the most promising for our demonstration.

Against CMUnited99 Virtual Werder performs better with a defensive formation. The average loss against the CMU-team was 0:14 with the 3-4-3 and 0.1:9 with the defensive strategy (we might add that this is an unacceptable situation in total). Table 2 shows the average results. Against Mainz Rolling Brains on the other hand we can see that Virtual Werder performs better with the offensive formation. The average score of 3.1:0.7 was better than the 0.5:0.9 score with the defensive formation. We believe that the understaffed mid-feld caused this situation (see table 2).

We come to the conclusion that the online-coach can help to detect the opponent's strategy. Once a team knows the play system of the opponent, appropriate counter actions can be carried out. However, later experiments have shown that we cannot exclude that the score is caused by other skills such as the individual play style. We think that further investigations with a 'standard team' would be helpful to make a clear point on this issue. In summary, we have seen that the Virtual Werder team performs better with this new information. The average score depends upon the chosen play system and whether the team can change their system online.

	Mainz RB	CMU-99
VW Def. 5-4-1	0.5:0.9	0.1:9
VW Off. 3-4-3	3.1:0.7	0:14

Table 2. Relation between formation and score

The technology of strategy detection could be useful for other application areas. Firstly, the quality of action predictions of physical agents can be improved which plays an important role within the control mechanisms of autonomous agents. Secondly, it is important to improve the robustness and security issues of electronic markets within the area of electronic commerce.

5 Future Work

Further work can be done in the following areas:

Keeping track of changes: This means that the coach-client consists of internal states. With internal states a list of 'scenes' describing the current play system of the opponent's can be stored. The next step is to detect changes in the opponent strategy. A low pass filter can then be used to determine whether the play system changed temporary or for a longer period.

Evaluation of counter attacks: The evaluation of a fitting counter-attack is another issue in our research. Therefore, we will focus on new criteria, such as play cycle and score. The idea is to change formations in situations that do not depend on the opponent's play.

Captain: Looking at formations is a first step to a more strategic play. The next step to improve the team performance will be the transfer of the coach knowledge to a key-player, which can give commands to the team members during the game, not only within a break. This "captain"-concept could also be extended with the concept of a key defense player, which is responsible for the guidance of the defense.

References

1. Christoph Biermann and Ulrich Fuchs. *Der Ball ist rund, damit das Spiel die Richtung ändern kann*. Kiepenheuer & Wisch, 1999.
2. CMU. Sources of low-level skills. <http://www.cs.cmu.edu/afs/cs/usr/pstone/mosaic/RoboCup/CMUnited99-sim.html>.
3. Emiel Corten, Klaus Dorer, Fredrik Heintz, Kosta Kostiadis, Johan Kummeneje, Helmut Myritz, Itsuki Noda, Jukka Riekki, Ratrick Riley, Peter Stone, and Tralvey Yeap. Soccerserver manual ver5.1 release. Manual, 1999.
4. Ian Frank, Kumiko Tanaka-Ishi, Katsuto Arai, and Hitoshi Matsubara. The statistics proxy server. In Tucker Balch, Peter Stone, and Gerhard Kraetschmar, editors, *4th International Workshop on RoboCup*, pages 199–204, Melbourne, Australia, 2000. Carnegie Mellon University Press.
5. J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*, volume 1. Addison-Wesley Publishing Company, Redwood City, California, 1991.
6. H. Kitano, Y. Kuniyoshi, I. Noda, M. Asada, H. Matsubara, and E. Osawa. Robocup: A challenge problem for ai. *Artificial Intelligence Magazine*, 18(1):73–85, 1997.
7. Taylor Raines, Millind Tambe, and Stacy Marsella. Automated assistants to aid humans in understanding team behaviors. In *Fourth International Conference on Autonomous Agents (Agents 2000)*, Barcelona, Spain, 2000.
8. Raul Rojas. *Neural Networks - A Systematic Introduction*. Springer-Verlag, Berlin, 1996.
9. Peter Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. Intelligent Robotics and Autonomous Agents. MIT Press, Cambridge, Massachusetts, 2000.
10. Peter Stone, Patrick Riley, and Manuela Veloso. The CMUnited-99 champion simulator team. In Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Berlin, 2000. Springer-Verlag.
11. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, 12(3):165–188, 1998.
12. Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48, 1998.
13. Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In M. Asada and H. Kitano, editors, *Robot Soccer World Cup II*, Berlin, 1999. Springer-Verlag.
14. A. Zell, G. Mamier, M. Vogt, N. Mache, R. Hubner, K.U. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, A. Hatzigeorgiou, S. Döring, and D. Posselt. Snns: Stuttgart neural network simulator, user manual version 3.2. Technical Report 3/94, Universität Stuttgart, 1994.

SBC++ Simulator Team

Amin Abbaspour, Mahmood Rahmani, Bahman Radjabalipour, and
Eslam Nazemi

{a-abbaspour, m-rahmani, b-radjabalipour, nazemi}@ce.sbu.ac.ir

Electrical and Computer Engineering Faculty
Shahid Beheshti University

1 Introduction

In this article the SBC++ simulator team and its overall design and scientific goals are described. Our main goal in this project is to achieve new methods of machine learning and deciding; also multi-agent behaviors such as strategy setting and cooperative learning with insistence on behavior networks. To achieve this, SBC++ uses MASIM[1] model, designed by Maes, as its fundamental base.

Our team uses the CMUnited-99[4] low level code with little modification. Major modification and improvements are in the high level code.

2 Team Development

Team Leader:

- Eslam Nazemi
– Iran
– Faculty member
– Attended the competition

Team Members:

- Amin Abbaspour, Mahmood Rahmani, Bahman Radjabalipour
– Iran
– Undergraduate student
– Attended the competition

Web page

<http://www.sbu.ac.ir/sbc/>

3 REASM behavior network model

MASIM(Maes Action Select Mechanism) was first introduced in 1989 by Maes[1]. in 1999, Dorer[2] introduced a revised and expanded version of it called REASM. Ten years of research on MASIM resulted in REASM, which showed its abilities in RoboCup 99[3].

Here we have successfully added a new learning method to REASM. To observe this, a brief of what REASM is and how it works might be helpful. More detailed information about REASM and its principles can be found in [2].

3.1 REASM in brief

There are three main layers in this model:

- Goal layer
- Competence layer
- Perception layer

Each layer contains several modules. Functions related to these modules are executed in each step. Perception layer gives information about the world to the competence layer through variable ' e ', which shows the executability of choosing each competence module. All competence modules have a related behavior for themselves. Each time a competence module is chosen, its behavior is executed. Competence modules also have a set of parameters as follow:

- | | |
|----------------|--|
| γ | Activation of modules |
| δ | Inhibitions of modules |
| β | Inertia of modules |
| θ | Activation threshold, $\theta \in (0, a]$, where a is threshold upper limit |
| $\Delta\theta$ | Threshold decay |

Competence modules are activated and inhibited by goal modules or other competence modules. The following formulas calculate this activation and inhibition for competence module k :

$$a_{k_{g_i}}^{t'} = \gamma \cdot f(i_{g_i}, r_{g_i}^t) \cdot ex_j \quad (1)$$

$$a_{k_{g_i}}^{t''} = -\delta \cdot f(i_{g_i}, r_{g_i}^t) \cdot ex_j \quad (2)$$

$$a_{k_{g_i}}^{t'''} = \gamma \cdot \sigma(a_{succ g_i}^{t-1}) \cdot ex_j \cdot (1 - \tau(p_{succ}, s)) \quad (3)$$

$$a_{k_{g_i}}^{t''''} = -\delta \cdot \sigma(a_{conf g_i}^{t-1}) \cdot ex_j \cdot \tau(p_{succ}, s) \quad (4)$$

$$a_{k_{g_i}}^t = abs \max(a_{k_{g_i}}^{t'}, a_{k_{g_i}}^{t''}, a_{k_{g_i}}^{t'''}, a_{k_{g_i}}^{t''''}) \quad (5)$$

$$a_k^t = \beta a_k^{t-1} + \sum_{i=1}^n a_{k_{g_i}}^t \quad (6)$$

where:

- n Number of goal modules linked to this competence module
- ex_j Expectation of goal module linked to the competence module
- $\tau(p, s)$ A triangular function of perceptron p and world-state s
- $f(i, r)$ A triangular function of importance i and relevance r
- $\sigma(x) = 1/(1 + \exp(k(\mu - x)))$ Goetz[5] formula

After calculating all competence modules' activation, it is combined with executability of them using a non-decreasing function $h : R \times [0, 1] \rightarrow R$, $h(x, 0) = 0$. If the highest value of $h(a_k^t, e)$ lies above θ then its related competence module is chosen and its action is executed; otherwise θ is reduced as much as $\Delta\theta$. Any time the highest activation reaches this reduced threshold, θ is set to its base value again.

3.2 Toward Dynamic Activation Setting in REASM

To add learning property, we tried to use the same way cerebellum uses for its learning processes. Cerebellum compares what is supposed to be done and what is actually done; then by changing the activation of the neurons, tries to reach the ideal state.

In REASM there are several goal modules in goal layer; this gives you the opportunity of goal-oriented learning. We had to have a function considering all goals but tending to those goals which have higher importance value i . So this simple rule saying, "Do a work which tries to satisfy the most important goals" was used. Hence changing activation γ and inhibition δ of competence modules simulates learning.

γ and δ are changed according to the following rules(competence module k):

$$\gamma_k^t = \gamma_k^{t-T_k} \cdot \psi\left(\frac{\sum_{k=1}^n i_{g_i} \cdot \Delta f(i_{g_i}, r_{g_i}^t)}{n}\right) \quad (7)$$

$$\delta_k^t = \delta_k^{t-T_k} \cdot \psi\left(\frac{-\sum_{k=1}^n i_{g_i} \cdot \Delta f(i_{g_i}, r_{g_i}^t)}{n}\right) \quad (8)$$

$$\psi(x) = \frac{2}{1 + e^{-\eta x}} \quad (9)$$

Here each behavior has an estimated time for its effect T_k . Actions performed by the competence layer changes the world-state, therefore output of goal functions is different between t and $t - T_k$. We take advantage of this difference. An average is calculated by adding difference of goal modules' output at time t and $t - T_k$ times goal's importance (through which important goals obtain higher effect in final average). If this average lies above zero, function ψ increases γ and decreases δ , otherwise decreases γ and increases δ .

ψ is a simple sigmoid function. The only extra parameter in ψ is η which is the learning rate and in this model is set manually. Experiences show that $\eta = 0.10$ gives better results in RoboCup domain.(this value is strongly related to your perception layer design and its output).

Some of this world-state changing between $t - T_k$ and t is caused by other agents, so it would be much effective if we take into account only those changes which are the result of this agent's activation. Overcoming these deficiencies are our future work and research topics.

It is clear that changing the activation of competence modules leads to separate activation values for each module. (previously, γ was common among all modules.)

4 Conclusion and future goals

The SBC++ simulator team is at its early stages of development and wishes to introduce new methods of learning and strategy setting.

What is said above is a part of our attempts. In future we will try to use multi-layer-input system in REASM to increase its learning capacity and adjusting network by using on-line coach.

References

- [1] Maes, P. (1989). The Dynamics of Action Selection. In Proceedings of the International Joint Conference on Artificial Intelligence-'89 Morgan Kaufmann, Detroit.
- [2] Dorer, K. (1999). Behavior Network for Continuous Domains Using Situation-Dependent Motivation. To appear in : Proceedings of the 16th International Joint Conference of Artificial Intelligence Morgan Kaufmann, Stockholm.
- [3] Dorer, K. (1999). Extended Behavior Networks for the magmaFreiburg Soccer Team, Center for Cognition Science, Institute of Computer Science and Social Research Albert-Ludwig-University, Freiburg.
- [4] Stone, P. et al (1999). The CMUnited-99 Champion Simulator Team AT&T Labs-Research,180 Park Ave.,room A273,Florham Park,NJ 07932.
- [5] Goetz, Ph (1997). Attractors in Recurrent Behavior Networks, Ph.D thesis. State University of New York, Buffalo.

A Team

Hidehisa Akiyama

Department of Information Processing, Tokyo Institute of Technology

1 Introduction

Soccer has the problem of the extremely high complexity. This complexity can be considered a problem of the search space. When the player decides the action from the situation of the world, quite many choices exist and it is impossible to search for all the choices. This search space problem is essence of the problem in soccer.

Genetic Programming(GP) is an effective technique for such a search space problem (e.g. Traveling Salesman Problem). We applied GP to the Soccer Server System, and tried to acquire the effective action patterns and its parameters. It is assumed that GP is applied to the low-level skills and the high-level behavior separately. This paper describes only the application of GP to the low-level skill part.

2 Genetic Programming(GP)

GP is a kind of Genetic Algorithm(GA), which is the evolutionary computing method[3]. GA is an optimization method that imitate the process of the evolution of the living thing. The population of the virtual living thing is generated in the computer. The selection and the reproduction are repeated until the best solution is obtained.

In GA, the chromosome of individual is expressed by the bit string. On the other hand, in GP, the chromosome of individual is expressed by the tree structure. This tree structure is equivalent to S-expression in LISP. Therefore, GP is suitable for the function identification, the acquisition of best IF-THEN rule, and so on.

GP generate the optimized solution automatically by repeating the generation change. However, when GP is applied to the Soccer Server System, it is difficult to optimize all control rule of the player by the tree structure of one individual. The high-level behavior for soccer player requires the low-level skills for soccer player. If we optimize the low-level skills and the high-level behavior simultaneously, many same structural sub-trees are needed in the tree structure and uselessness is caused in the optimization process.

Therefore, according to the action level of the player, it is necessary to apply GP to each level individually.

3 Application of GP to low-level skills

We applied GP to low-level skills, ‘kick’ and ‘dribble’.

At first, we tried to optimize dribble by using only basic commands(kick, turn, dash). However, even if hundreds of generations passed, the improvement of the fitness of individual was not seen at all. By this experiment, it turned out that it was difficult to optimize the processing structure and the parameter simultaneously even if it was basic action like the dribble. Then, first of all, the problem was simplified to the optimization of the kick, which is more basic action than dribble.

In the optimization of the kick action, the necessary is only optimization of the parameter of the kick command. This can be considered a problem of the function identification. The parameters used as an environmental input are distance to the ball, direction to the ball, velocity of the ball, distance to the target and direction to the target. The number of types of the value that the node returns is four(power, angle, Boolean value and real number). A final distance difference between the ball and the target position was used for the criterion of the fitness of the individual.

The optimization experiment was done by the above-mentioned setting. The transition of the fitness of the result is shown in Figure 1. Consequently, an excellent function to calculate the best parameter from an environmental input was acquired.

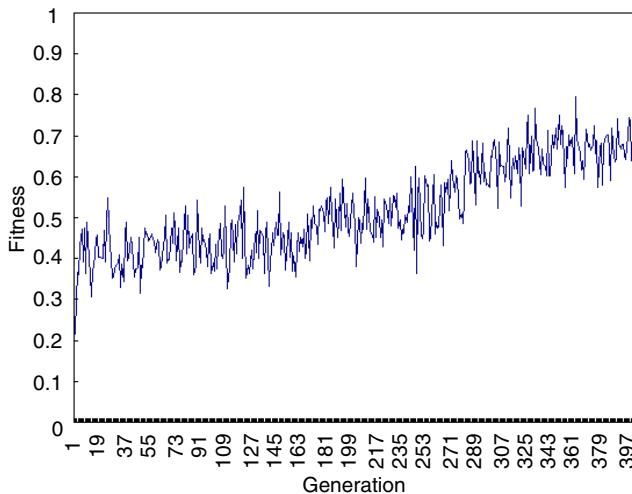


Fig. 1. Transition of Fitness : Kick

Next, the optimization experiment of the dribble was done by using the result of optimized kick parameter function. In this experiment, if the ball is within

kickable area, the player kicks ball to a temporary target. If not, the player dashes to ball or turns to ball(if necessary). Because the parameter of the kick has already been optimized, the necessary for dribble is optimization of dash power and temporary dribble target. The time took to reach the target was newly set as a criterion of the fitness of the individual. Other setting is the same as the case of the kick. After several hours of experiment, a roughly excellent result was obtained. The transition of the fitness of the result is shown in Figure 2. However, because this experiment is done in the environment where obstacle player does not exist, its effectiveness might be low.

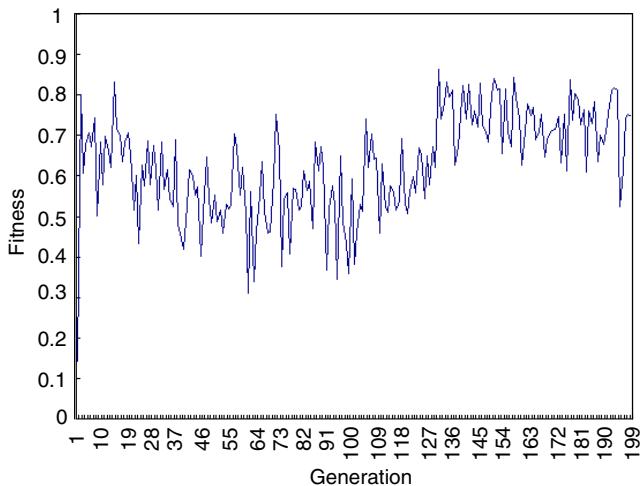


Fig. 2. Transition of Fitness : Dribble

4 Application of GP to high-level behavior

In high-level behavior, the action of a soccer player is classified. These classified actions are basic actions as a soccer player. For Example, pass, shoot, dribble, and so on. After acquiring the low-level skills, basic actions can be easily composed by combining them. Basic actions here include the passing to specific teammate, the shot, and the dribble to a specified position, etc.

Now, it is possible only to experiment in very limited the game situation(e.g. 3 vs. 3), and the result is not enough. Therefore, the result of the experiment has not been used in the actual competition yet.

5 Conclusion

The application of GP to the Soccer Server System was effective to develop low-level skills. However, in high-level behavior, that effectiveness cannot be shown yet. In competition of RoboCup2000, because making a high-level behavior was insufficient, the result was a total defeat. In the future, it is scheduled to acquire a high-level behavior with GP, and to verify its effectiveness.

6 Team Development

Team Leader: Hidehisa Akiyama

Team Members:

Hidehisa Akiyama

- Tokyo Institute of Technology
- Japan
- graduate student
- did attend the competition

Web page No Web page

References

1. Sean Luke at el. Co-Evolving Soccer Softbot Team Coordination with Genetic Programming. In Proceedings of the RoboCup-97 Workshop at the 15th International Joint Conference on Artificial Intelligence(IJCAI97). H Kitano, ed. IJCAI. 115-118. 1997.
2. David Andre, Astro Teller. Evolving Team Darwin United. RoboCup-98: Robot Soccer World Cup II, Springer Verlag, 1999
3. John R. Koza. Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge MA, 1992
4. Hitoshi Iba. Genetic Programming. Tokyo Denki University Press.

AT Humboldt 2000 (Team Description)

Hans-Dieter Burkhard, Joscha Bach, Kay Schröter, Jan Wendler*,
Michael Gollin, Thomas Meinert, and Gerd Sander

Humboldt University Berlin, Department of Computer Science,
Artificial Intelligence Laboratory, D-10099 Berlin, Germany

{hdb,bach,kschroet,wendler,gollin,meinert,sander}@informatik.hu-berlin.de
<http://www.ki.informatik.hu-berlin.de>

1 Introduction

Our agent team *AT Humboldt 2000* is partly an extension of our former team *AT Humboldt 99*[2, 3]. Again we used a BDI architecture. Especially the world model and some skills were revised. A new timing concept and a completely different architecture for the deliberation component were developed. The actual development was subject of an undergraduate course. Because of problems with the integration of components developed by different work groups, we were forced to start in RoboCup 2000 with a mixed team, consisting mainly of an extended version of the players used in EuRoboCup 2000. Only the goalie used all our new concepts.

2 Special Team Features

We are interested in virtual soccer for the development and the evaluation of our research topics in artificial intelligence, which concern the fields of Agent oriented techniques, Multi-Agent Systems and Case Based Reasoning. Thus many aspects of our soccer program are heavily influenced by these fields. Key feature of our actual agent concept is a deliberation structure that allows for reasoning and evaluations persisting over several simulation steps, while low-level behaviours (skills) are continuously executed (in the form of plans), as long as neither the deliberator nor conditions in the environment of the agent call for a change of action. The deliberator is going to be extended to allow for high-level descriptions of situations and orchestrated action.

3 World Model

AT Humboldt 2000 uses the same world model concept as in our previous teams[1–4] (plus several serious bugs that impair the performance of the basic

* This work has been partially supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316).

skills). For any given time a situation object will be generated which consists of object representations for teammates, opponent players, the ball and the agent itself. Flags are only used to determine the own absolute object representation which consists of the absolute player position, speed, body direction and of the relative face direction. With this data the absolute objects, representations of the other players and the ball are calculated. The agent can get a new situation-object from sensory information. Another way to get a new situation-object is the simulation of the actions of the agent, like it is done in the *SoccerServer*. So after every sensor information we have two concurrent Situation-objects which are merged together by finding corresponding players and using the best information of both situations for the new one. Reliability of information is currently only assessed by time-stamps.

4 Communication

Though rarely used, the agents are able to communicate by means of compressed strings of data, consisting of world-model and status/intention information. To ensure the integrity of the data, the string is time-stamped and check-summed.

5 Skills

AT Humboldt 2000 makes use of a new kick skill with adjustable start and target speeds and avoidance of nearby enemy players. Kick actions consist of one to four kick commands and are calculated, not learned. Tests using a correct world model have shown that accurate kicks of 2.4 m/s were consistently achieved. Because of our malfunctioning world model, the new kicking skill turned out to be of very little use and did not show any practical improvement over the old one during RoboCup2000. Consequently, dribbling and passing that should capitalize on better kicks, did not improve, while goal kicking got more powerful.

6 Strategy

Our agent architecture uses a mental deliberation structure which is best described by a belief-desire-intention architecture (BDI) [5]. Distinct from other (e.g. logically motivated) approaches our approach is closely related to procedural thinking, and we use object oriented programming for the implementation.

To cope with the increasing number of different intentions, a concept of nested intentions was chosen[6]. The design allows for high-level descriptions of plans and strategies (i.e. double passes, clique playing etc.) to combine pre-adjusted with emerging behaviour, a process we consider as necessary to adapt to strategic features and weaknesses of opponent teams. *AT Humboldt 2000*, however, failed to pose any real threat for advanced opponents, mainly because of its poor basic skills.

7 Team Development

The virtual soccer team *AT Humboldt* is developed by our AI group. Most of the work is done by students. As in the previous years some tasks are practical exercises for an AI course during summer semester. Other aspects are examined in diploma theses. A small core development group maintains the code and coordinates the work. Besides the experiments with AI methods, the project is also a challenge for software development: The sources of *AT Humboldt* contain over 40.000 lines of code. It is a non trivial task to integrate new ideas during extremely short intervals.

To support the concurrent development we use the freely available source code management system CVS[7] and the documentation system doc++[8]. We have developed an extensive testing and debugging tool – the *logbrowser*. It controls the log player and synchronously displays information on the agents. Information from different components of our agents can be selected or deselected for displaying, as desired. More efficient logging mechanisms and a compressed log format make it possible to write detailed logs of all agents at real time conditions, which turned out to be very helpful in the cases where it was consequently put to use.

Team Leader: Prof. Hans-Dieter Burkhard

Team Members:

Prof. Hans-Dieter Burkhard (professor, leader of the AI group)

Joscha Bach, Kay Schröter (research assistants)

Jan Wendler (PhD student)

Michael Gollin (undergraduate student)

- Humboldt-University Berlin, Germany

- did attend the competition

Thomas Meinert, Gerd Sander (undergraduate students)

- Humboldt-University Berlin, Germany

- did not attend the competition

Web page <http://www.ki.informatik.hu-berlin.de/>

8 Conclusion

The main goals of our further work are the extension and the refinement of the new deliberation concept, the development of a new world model, the usage of learning and CBR techniques to evaluate situations and preselect useful intentions, and the modelling of opponents in the frame of the BDI architecture to anticipate their behaviour.

Furthermore we want to support our sony legged robot team, the *Humboldt Hereos*, with techniques already used by our simulation team.

References

1. H.-D. Burkhard, M. Hannebauer, and J. Wendler. AT Humboldt - Development, Practice and Theory. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *LNAI*, pages 357–372. Springer Verlag, 1998.
2. H.-D. Burkhard, M. Hannebauer, J. Wendler, H. Myritz, G. Sander, and T. Meinert. BDI Design Principles and Cooperative Implementation — A Report on RoboCup Agents. In *Proceedings of the IJCAI-99 Third International Workshop on RoboCup*, Stockholm, Schweden, Aug. 1999.
3. H.-D. Burkhard, J. Wendler, T. Meinert, H. Myritz, and G. Sander. AT Humboldt in RoboCup-99. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, LNAI. Springer Verlag, 2000.
4. P. Gugenberger, J. Wendler, K. Schröter, and H.-D. Burkhard. AT Humboldt in RoboCup-98. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, volume 1604 of *LNAI*, pages 358–363. Springer Verlag, 1999.
5. A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In V. Lesser, editor, *Proceedings of the ICMAS-95*, pages 312–319. MIT Press, 1995.
6. G. Sander. The Intention-Agent: a new approach for a solution of longterm cooperation in RoboCup. In H.-D. Burkhard, L. Czaja, A. Skowron, P. Starke, editors, *Proceedings of the Workshop on Concurrency, Specification and Programming 2000*, Informatik-Bericht 140(2), pages 235–239. Humboldt-University, 1995.
7. CVS: <http://www.loria.fr/molli/cvs-index.html>
8. DOC++: <http://www.zib.de/Visual/software/doc++/index.html>

AIRG Sibiu

Ciprian Candea and Marius Staicu

Artificial Intelligence Research Group Sibiu
St. Reconstructie Nr. 2A, 2400, Sibiu, Romania

1 Introduction

In this year the AIRG group prepares a new team for the RoboCup competitions based on new ideas and concepts. Based on the MAS(Multi Agent Systems) architecture implemented by us in 1999 for the Sibiu Team that took part at RoboCup 1999 competition hold in Stockholm, the paper describes a generic architecture for a RoboCup team with four holon types situated on three levels. This architecure tries to synergistically combine the characteristics of both domains: MAS and HMS(Holonic Manufacturing Systems).

2 Special Team Features

Our main research focus in this year was to promote a new approach able to increases the flexibility of decisional system. Accordingly with this new model we modified the decisional architecture used by us in 1999 keeping at the same time the functionality. First of all, our idea has many connections with the PROSA architecture developed at PMA/KLUEven as a reference model for Holonic Manufacturing Systems. On the other hand, some concepts regarding the decision control and structure in team-based settings has been inspired from the studies that are usually approached in the field of GDSS (Group Decision Support Systems) research. The acronym PROSA came from Product-Resource-Order-Staff Architecture, the holon types used. The resource holon contains a physical part namely a production resource of the manufacturing system, and an information processing part that controls the resource. The product holon holds the process and production knowledge to assure the correct making of the product with sufficient quality. The order holon represents a task in the manufacturing system. It is responsible for performing the assigned work correctly and on time. The staff holon is implemented in the idea to assist the rest of three holons in performing their work. A holon is the fundamental part of any holarchy, and it can be referred to as an agent who has outspoken cooperatively and autonomy characteristics.

3 World Model

The World Model is one of the most important parts of any Robosoccer agent. The entire agent behavior is build around it and that is why the model must

be as close as possible to the reality. The world is represented using an object-oriented approach. Each thing on the field is an object. The hierarchy is modeled subsequent to the object hierarchy of the robocup simulator. The data in the objects are the position, velocity, acceleration and other characteristics (shirt number, team side for players). The interesting part of our world model is that we have two copies of it. One is used by the input thread (the one that receives the simulator messages), and the other by the other holons. The idea behind this architecture is that while some holons access the world model data we cannot update it (if a SEE message is received). Using two copies we can process a SEE message as soon as possible. After the message is processed, on the next cycle, the other copy is updated and the holons will have the correct information. If after a cycle there is no see message, the world model will be updated using the current object position, their velocity and acceleration (if available) and the agent position, velocity and acceleration. Using this method the world model is kept as close to the reality as possible and it's updated without interfering with the agent operations. Even if the agent misses to receive messages for one or two cycles (because of a lengthy operation or bad skills), with this architecture the model will still be updated in background and will be available at the end of the operation. If some information for a player isn't known, depending on what information is missing (shirt number, team) the player will be used as a teammate or as an obstacle. The world model keeps a history for the two last see messages. Using this history it will update the data even for objects that weren't seen in the last 5 - 10 cycles. After 10 cycles, if the object isn't seen its position is marked as unknown. For the position of the objects that were computed using the history in the model there is a confidence factor that decrease in time. When the object is seen this factor is 95 and will decrease until 0 after 10 cycles. Further at each 100 cycles (mean value), each agent try to send to the teammates the data for the closest objects they see. This way the agents can update/improve their model.

4 Coach

Last year we used an online coach for next things. First of all we decided to implement inside an algorithm for identification in the opponent team of the correspondent player from our team i.e. if our defender from left has number 3 what is the corresponding opponent? This type of information is necessary because our neural network was trained using the correspondent opponent position and number (practically the opponent role). In the same time when it is possible our coach send to all team some information about all players position and the ball. All this function was kept in the current version of coach and now with our new architecture we have possibility to extend with other capabilities like player change or assign to a player a temporary role, really easy. To determine the pairs (our player - opponent player) we use a ray trace algorithm.

5 Skills

Because we try to propose a new architecture we implement our skills using mathematical analysis. A part from skills idea used in our implementation was taken from Magma Freiburg code. The positioning problems it is one of the most important problems in life generally. For RoboCup teams we resolved this task using a decision tree. In this tree we have a lot of possibilities and each possibilities is very easy defined based on predicates. The number of predicates is not so big (i.e. we use 15 predicates) and finally we can create in this way a simple methods to move our players on field. To pass we use a special predicate who try to predict based on the current field situation what teammate is in the best position to receive a pass. We used a probability algorithm for each teammate and this probability is calculated based on current position and how many opponent players are in area and what intention they have. The rest of skills implemented by us I think that are classical in this moment.

6 Strategy

As we already said for strategy we used the method described in [1] where each team member tries to learn from the corresponding human(or opponent team) player in a real game. Following a unified approach, strategic and tactical behavior is learned by training a feed-forward neural network (ANN) with a modified back-propagation algorithm. We implemented a feed-forward neural network made up by three layers. The ANN has 20 inputs (two for each of the 19 players and the ball - because the goalkeepers have no role to play in this simplified strategy). The player inputs are composed of two bits enabling to represent four different vectors describing the location of the players or ball, related to each agent (that means, four distinct "tactical" components of the overall picture). In this way our players can learn a strategy at the team level practically we clone the behavior from other opponent players. For example if we want to play with a CMU base strategy we have to take a game with CMU vs any other team and to train off-line our ANN. After the training period we have 10 strategy behaviors (one for each agent) based on CMU strategy. More we observe that in same situations our player apply some tactics unimplemented by us in the corresponding decision tree. The ideal model for our approach is to change the tactics for each opponent team in part but this think it is not possible in competition time, is necessary some time to train the ANN.

7 Team Development

Our team is part from a larger private research group situated in Sibiu, Romania. Last year we develop our team like a student program where we study some interesting idea about behavioral learning and we used the skills from HCII and the implementation was made in java. Because we had an interesting and new idea about how it is possible to put together two different domains like MAS

and HMS we decide to participate again. This year we rebuild and rewrite all team because we proposed a new architecture and it was necessary to propose also a new world model. We use in our developing process some tools created by us like debugger or decision tree maker. The most important tool is the graphic debugger. With this debugger we can see everything what an agent do and more what he will do. Of course we can observe and analyze all logs off line and see why in a very clear situation he act maybe stupid. In this way the necessary time to develop our team was acceptable (2 months).

Team Leader: Ciprian Candea

Team Members:

Marius Staicu

- Artificial Intelligence Research Group Sibiu
- Romania
- Computer Science Enginer

Gabriela Iozon Simona

- Artificial Intelligence Research Group Sibiu
- Romania
- Computer Science and Matemathics Teacher

Web page <http://airg.verena.ro>

8 Conclusion

The results obtained in this year suggest that we are in a promising starting point for future work. This validates the approach of adding functionality stepwise (first of all flexibility) i.e. improving the 1999 architecture. Following the tests we intend to introduce a new level between the coach and the players - the team level. With this new level, we think that the architecture flexibility will increase substantially. For the game strategy the order holon is still insufficiently adapted. Our intention is to improve this team until next year and off course to win. We wold like to particularly mention the collaborators of the AIRG, namely Adrian Caraiman, Constantin Zamfirescu, Prof. Boldur Barbat and Prof. Daniel Volovici.

9 Bibliography

[1]Candea,C and Oancea,M and Volovici,D (1999). Emulating real soccer, in Proceedings of the International Conference Beyound 2000 Sibiu, pp. 35-38

[2] Giebels,M and Kals,H and Zijm,H (1999) Building Holarchies for Concurrent Manufacturing Planning and Control. Proceedings of the second International Workshop on Intelligent Manufacturing Systems, Leuven, Belgium, pp.49-56

[3] Hino,R. and Moriwaki,T (1999) Decentralized Scheduling in Holonic Manufacturing System. Proceedings of the second International Workshop on intelligent Manufacturing Systems, Leuven, Belgium, pp. 41-47

[4] Kitano,H and Asada,M and Kuniyoshi,Y and Noda,I and Osaw,E. Robocup: The Robot World Cup Initiative

The NOAI Team Description

Anneli Dahlström, Fredrik Heintz, Martin Jacobsson,
Johan Thapper and Martin Öberg

*Department of Computer and Information Science, Linköping university
SE-581 83 Linköping, Sweden*

E-mail: {annda, frehe}@ida.liu.se, {marja319, johth341, marob265}@student.liu.se

Abstract. The NOAI team is built on the RoboSoc framework, which is a system for developing RoboCup agents for educational use. The decision making is behavior based and is a continuation of the work of Paul Scerri et al. on the Headless Chickens III. We have extended their decision trees to directed acyclic decision graphs and made it possible to use full predicate expressions instead of just predicates. We have also added a language for describing the decision graphs. This is to be able to use their, highly successful, strategy editor to graphically design the team strategy.

1 Introduction

This paper describes the most important aspects of the RoboCup simulation team NOAI. The team is built on the framework provided by RoboSoc [2], which is a system for developing RoboCup agents developed for educational use. The team is a part of the evaluation of RoboSoc.

Since we want to use the strategy editor developed by Johan Ydrén and Paul Scerri [4] we use a behavior based decision making that closely resembles their approach. The strategy editor was one of the most important factors in the success of Headless Chickens III (HCIII) in RoboCup'99 in Stockholm.

2 RoboSoc

The purpose of RoboSoc is to simplify the development of RoboCup agents by taking care of the basic problems that an agent has to handle, like interacting with the soccer server and keeping track of the current simulation cycle.

The resulting system consists of three parts, the library, the basic system and the framework. The library consists of basic objects and utilities used by the rest of the system. The basic system takes care of the interactions with the server, like sending and receiving data. It is also responsible for the timing and most of the support for decision making, since it generates events for the decision maker to react on when the world or the internal state of the agent changes.

The framework defines three concepts, used for world modeling and decision support, *views*, *predicates* and *skills*. The views are specialized information processing units responsible for a specific part of the world model, like modeling

the ball or the agent. They are also controlled by events generated by the basic system. The predicates can be used either by the decision maker or the skills to test the state of the world. They work like predicates in a fuzzy logic, and can have a value between zero and one depending on the certainty of the answer. The skills are specialized, short-term planners which generate plans for what actions the agent should take in order to reach a desired goal state. For more information see [2].

3 High Level Decision Making

The decision making architecture is based on the work of HCIII [3]. It uses the same two layers as they do, one for low level skills and one for high level strategy. The strategy layer access the information in the world model through a given set of fuzzy predicates. It also use a behavior based decision making, similar to HCIII, but with some modifications to achieve more efficiency, flexibility and expressiveness. The architecture of the behavior hierarchy tree is described in [1].

In the following text we will separate the person doing the programming of the agent from the person designing the behavior of the agent. The reason is that the agent behavior designer could (or maybe should) be a domain expert, in this case a soccer coach or similar, without programming skills. When we refer to the programmer we mean a person that is capable of changing the actual code and then recompile the program, while the agent designer only needs to change configuration files that are loaded when the agent is started.

3.1 Behavior DAG

The task of a behavior is to choose a lower level behavior to activate. Therefore every behavior is defined by a list of lower level behaviors and the technique used to choose among them. The choice is made with the concept of activations. Every behavior has an activation level and the behavior with the highest activation level is chosen. The activation level is increased or decreased every cycle based on the value of a predicate expression. The amount of change can be specified for each behavior. Different behaviors uses different predicate expressions.

In HCIII the behaviors are layered, a behavior in layer n can only activate behaviors in layer $n - 1$. The layer 0 only contains skills, which sends the actual server command to the soccer server when they are activated.

The layering is unnecessary, it can be generalized to let the behaviors form a rooted directed acyclic graph (DAG) where the nodes are behaviors and the terminals are skills. The decision is made by finding a way from the root behavior through the DAG to a skill and activating it.

The agent designer, who constructs the behavior DAG, may of course still use a layered approach but it is not necessary. She can for instance layer the behavior DAG as in Figure 1.

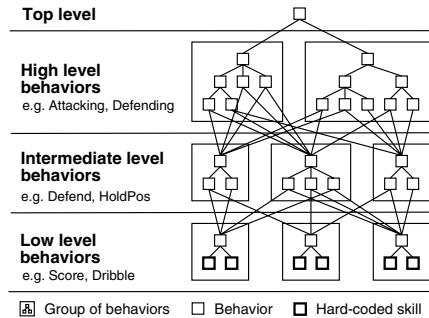


Fig. 1. Example layering and grouping of behaviors.

3.2 Predicate Expressions

The interface between the RoboSoc view system and the behavior based decision system is the predicates. The predicates are programmer defined objects that use the information stored in the views and return a fuzzy boolean value. The predicates hides details so the agent designer can use high level information for the decision making. The predicates are used to update the activation levels in the behaviors.

Since the predicates are predefined by the programmer, the agent designer can not create new predicates when needed. To try to overcome this shortcoming, the designer can create fuzzy logic formulas of predicates with logical connectives. The formulas can be used when updating an activation instead of a single predicate, which was the case in HCIII. Especially the logical connective `not` is very useful, but `and`, `or` and `xor` are also available.

3.3 Activations

The concept of activations in HCIII is very useful, it can make the behavior of the agent very reactive and at the same time delay changes in actions if necessary. By delaying changes, the agent can handle temporarily incorrect information and avoid oscillating between behaviors.

One problem is comparing two options. For example if an agent chooses between dribble to the left or to the right. The decision should be based on the number of opponents to the left and to the right. You want to compare the number of the opponents in these areas.

This is solved by constructing another kind of predicates. Predicates that do not answer a boolean question, but instead gives a gauge value. One example is the `NumberOfOpponentsInArea`, which returns zero if no opponents are in the area and one if all 11 opponents are there. Now you need a special type of behavior that does not use activations when choosing the behavior to activate. This second type of behavior will still have a predicate associated with each choice, but instead of updating activations and choose the option with the highest

activation level, it simply choose the behavior with the highest predicate value. It is up to the agent designer to decide which behaviors should use activations.

A second advantage with these behaviors without activations is efficiency. Every activation in the behavior DAG must be updated every cycle by calling its predicates, even if that part of the graph is never activated. With the behaviors without activation only the active part of the graph needs to call its predicates. By using as many behaviors without activation as possible we can improve the overall efficiency of the agent and possibly have a larger behavior DAG.

3.4 Behavior Description Language

The behavior DAG is specified in a text file which means that you do not need to recompile your agent to change the behavior of the agent. When the agent is started it reads the text file and creates the behavior DAG. The language used to describe the behavior DAGs is made of S-expressions, that is a LISP-like syntax. The files can be created in any text editor, but also by tools such as a strategy editor [4]. Since a language for specifying the behaviors is defined, you can build different types of tools for creating behaviors.

4 Future Work

One possible direction for future work is to extend the behavior system and try to implement as much as possible of the skills with behaviors. Instead of having a dribble skill, it could be possible to build a dribble behavior by using simpler skills like, MoveToXY, KickBallSoftlyToXY, KeepBallToYourLeft and so on. Then more functionality can be decided by the agent designer instead of the agent programmer.

The language for defining behaviors makes it possible to use different kinds of tools to create behaviors. It would be interesting to compare different tools, to see how the speed of development and quality of the soccer agents depend on the different tools. It might also be interesting to apply machine learning in the process of creating strategies and behaviors.

References

- [1] Silvia Coradeschi, Paul Scerri, and Anders Törne. A User Oriented System for Developing Behavior Based Agents. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 173–186. Springer Verlag, Berlin, 1999.
- [2] Fredrik Heintz. RoboSoc a System for Developing RoboCup Agents for Educational Use. Master's thesis, IDA 00/26, Linköping university, Sweden, March 2000.
- [3] Paul Scerri, Johan Ydrén, Tobias Wiren, Mikael Lönneberg, and Per-Erik Nilsson. Headless Chickens III. In Manuela Veloso, Enrico Pagello, and Hiroaki Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*. Springer Verlag, Berlin, 2000.
- [4] Johan Ydrén and Paul Scerri. An Editor for User Friendly Strategy Creation. In *Robocup 1999 Team Description: Simulation League*. Linköping Electronic Press, 1999.

Improved Agents of the magmaFreiburg2000 Team

Klaus Dorer

Centre for Cognitive Science
Institute for Computer Science and Social Research
Albert-Ludwigs-University Freiburg, Germany
klaus@cognition.iig.uni-freiburg.de

1 Introduction

The magmaFreiburg2000 team, as its predecessor, is based on Extended Behavior Networks [4]. This paper focuses on the improvements made compared to the magmaFreiburg1999 team of Stockholm. The main domain-independent improvements have been made by introducing concurrent behavior selection into the Extended Behavior Networks framework [3]. Domain-dependent improvements have mainly been made on server synchronization and game strategy.

Again the performance of the magmaFreiburg2000 has been very satisfying finishing the competition at an equal fifth place. In addition to the good performance of our team it should be mentioned that we also had a big portion of luck when proceeding to the next round by the first coin toss decision of the RoboCup history (sorry again to Sharif Arvand).

2 Team Development

Team Leader: Klaus Dorer

Team Members: Susanne Dorer (software engineer), Metris corp., Germany

Web page <http://www.iig.uni-freiburg.de/cognition/members/klaus/robocup/magmaFreiburg2000.html>

3 World Model

The world model of magmaFreiburg2000 is still a domain-dependent model. Three main improvements have been made concerning world modeling.

First, the preciseness of ball-localization has been improved by allowing higher ball speed changes in localization filters in case of situations where the ball is close to a player. The agent is therefore able to faster react to kicks of other players.

Second, to improve dribbling the agents maintain an access map of their surrounding. This map specifies the distance (in all directions with steps of ten degrees) the ball can be dribbled without loss taking into account opponent

players and the stamina of the agent. In this way players can do few and harder kicks in cases where no opponents are close, speeding up the dribbling. A player will keep the ball close in cases where the player is attacked by an opponent.

And third, to improve clearance the agents maintain a direction map. It provides information at which distance a direction is blocked by an opponent player. When clearing a ball, the agent tries to optimize the kick direction by trading off preferred clearing direction and the information of the direction map.

4 Skills

Improvement of skills has been on lower priority this year although being sub-optimal. In fact, the main deficiency compared to the first four teams has obviously been a lack of hard-kicking ability of the magmaFreiburg2000 agents. Nevertheless, the overall skills of the agents improved, mainly due to the improved synchronization (see chapter 6).

5 Strategy

Main improvement on strategy has been the introduction of marking opponents by defensive players. Defenders and midfielders try to stay close behind (with respect to the own goal) the closest opponent player that is not yet marked. This prevents two players from marking the same opponent and assures that the additional dashes a player needs to do are small. The introduction of this new strategy also gives an impression on the flexibility and simplicity of Extended Behavior Networks. This marking strategy has been introduced into the behavior network by just adding a new indexical functional object 'opponentToBeMarked' and adding a new marking behavior, which was easily implemented by a standard dash method 'dashTo('opponentToBeMarked')'.

6 Special Team Features

One of the most remarkable features of the magmaFreiburg team is the clear separation of domain-dependent and domain-independent parts of the architecture. All issues of behavior selection are dealt with by the domain-independent behavior networks. Perception and action, which are inherently domain-dependent, as well as world modeling are done in separate software modules.

6.1 Domain-independent

Focus of our work has been to introduce a (domain-independent) concurrent behavior selection into the framework of Extended Behavior Networks (EBNs). Two requirements are necessary for concurrent behavior selection: it has to be known what resources a behavior uses and the access to resources has to be coordinated in order to avoid resource conflicts. The first is accomplished by

resource-perceptions providing the agent with the situation-specific amount of resources a behavior uses. The second requirement is accomplished by the introduction of resource nodes into EBNs. The resource nodes allow a decentralized coordination of access to resources making the complete behavior selection process of EBNs decentralized and parallelizable. For a more detailed description see [2, 3].

Concurrent behavior selection has been examined in a series of 30 soccer games. Since version 5 of the RoboCup-soccerserver, commands can be carried out concurrently, if they do not use the same resources. Two identical teams played against each other. The only difference was that one team used concurrent behavior selection, while the other team used serial action selection. The concurrent team's agents were able to do communication, head turning and running and kicking actions concurrently. Since the number of cycles an agent can communicate is restricted to 4% of all cycles and because separate turning of the head relative to the body was only executed in about 8% of all cycles, concurrent behavior selection effectively only took place in 2% of the cycles. Despite this, the team using concurrent behavior selection scored significantly¹ more goals than the team using serial behavior selection (see table 1).

	serial	parallel	p ($n = 30$)
Mean no of goals	2.4	4.3	< 0.001

Table 1. Comparison of serial and parallel behavior selection in the RoboCup domain.

6.2 Domain-dependent

Main domain-dependent improvements were achieved by improving the synchronization to the server. Former synchronization was simply done by sending a command to the server whenever a sense_body perception was received by the server. MagmaFreiburg2000 used a slightly more sophisticated synchronization. Players that are in possession of the ball switch to 75 ms view mode. After receiving a sense_body, the player waits for the next perceptual information. In this way, a player will base the decision for the next action on the perceived current state and not on the estimated state of the world as has been in 1999. This resulted in a significantly better result when comparing a team of agents with improved synchronization to magmaFreiburg1999 to the results of a team of agents with old synchronization to magmaFreiburg1999 (see table 2). For deeper insights into server synchronization see [1].

¹ two samples t-test with $\alpha = 0.01$.

	sense_body	see	p (<i>n</i> = 40)
Mean goal difference	1.45	2.55	< 0.01

Table 2. Comparison of synchronization based on sense_body to synchronization based on see information received by the server. The numbers show the mean goal difference on 40 games against the old magmaFreiburg team.

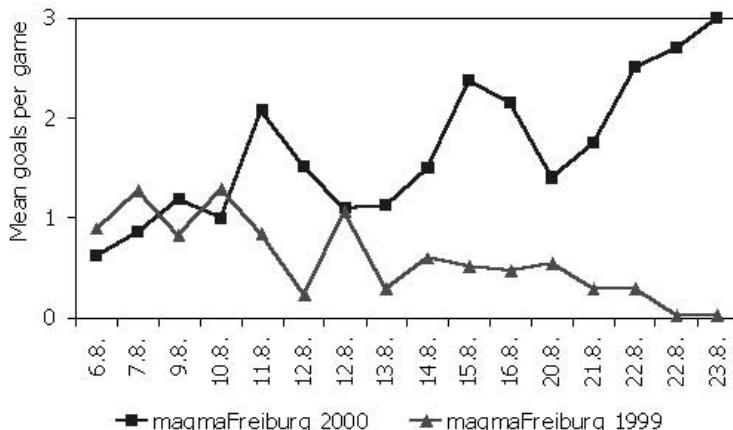


Fig. 1. Improvements made during the last three weeks of preparation for Melbourne. The last three measurements contain the improved server synchronization.

7 Conclusion

Despite the significant improvements on our team, the success of 1999 could not be repeated. This is just one indication that the general improvements made within the RoboCup simulation league have been enormous. This lead to a close-up of lots of teams and resulted in the most breathtaking games ever.

References

1. Butler, M., Prokopenko, M., and Howard, Th. (2000). Flexible Synchronisation within RoboCup Environment: a Comparative Analysis. In: *Proceedings of the Robocup 2000 Workshop*, Melbourne.
2. Dorer, K. (2000). *Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten*. PhD thesis, Albert-Ludwigs-Universität Freiburg, <http://www.uni-freiburg.de/freidok>.
3. Dorer, K. (2000). Concurrent Behavior Selection in Extended Behavior Networks. Available at: <http://www.iig.uni-freiburg.de/cognition/members/klaus>.
4. Dorer, K. (1999). Behavior Networks for Continous Domains using Situation-Dependent Motivations. *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1233-1238, Morgan Kaufmann, San Francisco.

Virtual Werder

Christian Drücker, Sebastian Hübner, Esko Schmidt,
Ubbo Visser, Hans-Georg Weland

TZI - Center for Computing Technologies
P.O.Box 334400

University of Bremen, Germany

{druecker|huebner|esko|visser|weland}@tzi.de
WWW home page: <http://www.virtualwerder.de>

1 Introduction

Virtual Werder is a new team in the simulation league. The team is the result of an initiative which has been formed from lectures on Artificial Intelligence at the Department of Mathematics and Computer Science at the University of Bremen.

The main focus of this team is to use the online coach for the detection of the opponents play system. The motivation and significance for this has been discovered during a structured interview with Thomas Schaaf, the manager of SV Werder Bremen.

2 Special Team Features

Our focus of research clearly is how to detect opponents strategies in a dynamic and real time environment. Our first thought was learning online but the available time to learn is not enough. We decided to manually create input/output pattern of 'typical' play systems and feed them into an artificial neural network. The plan is to learn the patterns and to use the online coach during a game to detect them. According to the classification an appropriate counter-attack will be chosen.

The technology of strategy detection could be useful for other application areas as well. Firstly, the quality of action predictions of physical agents can be improved which plays an important role within the control mechanisms of autonomous agents. Secondly, it is important to improve the robustness and security issues of electronic markets within the area of electronic commerce. We think that the proper analysis of the 'opponents' strategies in this area would help to improve the own situation.

To our knowledge work in the area of strategy detection has been done (see [3], [4], [2]). These approaches are designed for the analysis of games, off-line after playing, to gain new experiences for the next games. Frank et al. ([1]) present a real time approach which is based on statistical methods. A team will be evaluated statistically but there is no recognition of team strategies.

We think that the RoboCup platform is interesting for our research because we have to deal with a dynamic, uncertain, real time environment.

3 World Model

Due to the fact that Virtual Werder relies on the basic capabilities provided by the CMU99 client we did not touch the way the client is representing its world model. In fact we added the ability to broadcast the ball position whenever an agent is close to the ball to minimize uncertainty.

However, we did the first steps to an own world model with the representation of agents on the basis of probabilistic networks but it hasn't been implemented yet.

4 Coach

Our online-coach observes the positions of the opponent's players at given points in time, currently twice per second. A bounding box around these positions divided into eight to eight cells gives an input pattern for a neural network (a player inside of a cell yields an input value of 1.0, an empty cell an input value of 0.0). The neural network was trained with patterns from last year's RoboCup logfiles and own examples. It is able to recognize sixteen different formations that turned out to be played most often. The result of the neural network is the formation the opponent's team most likely is playing, and a probability for this result. If the probability exceeds a demanded threshold, the result is declared valid.

A decision tree is used to integrate the neural network's output and additional information, e.g. the size and position of the bounding box, the count of performed formation changes, the count of off-sides and the score. This leads to a decision whether or not a new counter formation is advisable and of which kind it would be. A chosen change of formation is then broadcasted to the team during the next interruption.

5 Communication

We use two different types of messages: a common one for all players and a special one for defense coordination purposes.

The common message contains the identification of the sender, its position, its distance to the ball, the actual ball position and the degree of confidence of this information. Additionally it contains the identification of the intended receiver in case the ball is passed. The message is sent by the agent who is closest to the ball. This avoids collisions between messages from different players.

The message for the defense coordination additionally contains the position of the defense line. It is only used by a special player determined to be the defense coordinator. In order to avoid collisions between the two types of messages they are restricted to certain cycles. The defense line is used to appoint the positions of the defensive players and to allow offside traps.

A problematic situation seems to arise when two or more players conclude that they are the player closest to the ball and set up a common message. In this

case the informations of both players should be so similar, that it is unimportant for their teammates which message they actually receive.

Basically every player is able to build its own world model, but the communication helps to make this model more precise. It is even possible to add positions of object that cannot be seen at all. With reduced or deactivated communication only additional informations are removed, leading to a worse but acceptable game. Especially the defensive would have the most problems since it needs a good working communication to synchronize the defense line.

6 Skills

Virtual Werder is based on CMU99, so most of the basic skills were already implemented, but we had to add skills to make our agents team-players.

We developed an evaluation function to determine how useful a pass to a position is, based on the distance and direction of the position and the number of opponents which may intercept the ball. This function is used by the player with the ball to find teammates to pass the ball to and by the players which want to receive a pass to get into a good position. The player with the ball evaluates the usefulness of passes to the positions of all its teammates. If teammates are in good positions it passes the ball to the one with the best evaluation result. To get into a good position the agent evaluates 80 random positions in a 60x60 square around it and then tries to get to the best one.

We also implemented a defense line which is organized by a central defensive player as described earlier in this paper.

7 Strategy

The main strategy of our team is to adapt our formation to the opponent by changing the behaviour of the players due to the calculations of the coach. To achieve this we have implemented 22 different agent behaviours. An agent starts with one of these behaviours, but is able to switch to another when necessary, especially when the coach changes the team formation. For the 2000 competition we had seven fixed formations for the coach to choose from, ranging from “cement mixer” (7-1-2) to “maniacal offensive” (0-5-5).

8 Team Development

We have developed a tool called export player to collect input data for our neural network. This graphic interface is used to sample data from logfiles and offers a list of play systems to classify a specific snap shot of a logged game. The tool provides an output file that is used as an input for the Stuttgart Neural Network Simulator (SNNSv4.1) [5]. This program creates C-code for a feedforward network.

Our team evolved from a lecture on artificial intelligence and became a self-organized student project. Virtual Werder has become the basis for further student activities at the Bremen university.

Team Leader: Ubbo Visser

Team Members:

Ubbo Visser

- TZI - Center for Computing Technologies, University of Bremen
- Germany
- Assistant Professor
- did attend the competition

Christian Drücker, Sebastian Hübner, Esko Schmidt, Hans-Georg Weland

- TZI - Center for Computing Technologies, University of Bremen
- Germany
- postgraduate students
- did attend the competition

Web page <http://www.virtualwerder.de>

9 Conclusion

Virtual Werder will send a new team to the 2001 world championships. In terms of sports we try to achieve a place within the first 16 teams of the world. How do we want to do this? Firstly, we will put more work on the detection of the opponents play, not only the overall play system is of our interest but also 'little' strategies such as the four-chain defense line or a static play over the wings. Secondly, we focus on uncertainty, something what we tried to do for the 2000 team already. We believe, that the performance of our team can be improved by methods from this area. Currently, two Masters students are working on these matters.

References

1. Ian Frank, Kumiko Tanaka-Ishi, Katsuto Arai, and Hitoshi Matsubara. The statistics proxy server. In Tucker Balch, Peter Stone, and Gerhard Kraetschmar, editors, *4th International Workshop on RoboCup*, pages 199–204, Melbourne, Australia, 2000. Carnegie Mellon University Press.
2. Taylor Raines, Millind Tambe, and Stacy Marsella. Automated assistants to aid humans in understanding team behaviors. In *Fourth International Conference on Autonomous Agents (Agents 2000)*, Barcelona, Spain, 2000.
3. Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the robocup soccer server. *Applied Artificial Intelligence*, 12(3):165–188, 1998.
4. Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48, 1998.
5. A. Zell, G. Mamier, M. Vogt, N. Mache, R. Hubner, K.U. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, A. Hatzigeorgiou, S. Döring, and D. Posselt. Snns: Stuttgart neural network simulator, user manual version 3.2. Technical Report 3/94, Universität Stuttgart, 1994 1994.

Kakitsubata Team Description

Tetsuya Esaki¹, Taku Sakushima¹, Shinji Futamase²,
Nobuhiro Ito¹, Tomoichi Takahashi², Wei Chen¹, and Koichi Wada¹

¹ Department of Electrical and Computer Engineering,
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555, JAPAN
[{saki,taku}](mailto:{saki,taku}@phaser.elcom.nitech.ac.jp)@phaser.elcom.nitech.ac.jp
[{itoh,chen,wada}](mailto:{itoh,chen,wada}@elcom.nitech.ac.jp)@elcom.nitech.ac.jp

² College of Business Administration and Information Science,
Chubu University
Matsumoto-cho, Kasugai-shi 487-8501, JAPAN
[{g95241,ttaka}](mailto:{g95241,ttaka}@isc.chubu.ac.jp)@isc.chubu.ac.jp

Abstract. In a multi-agent system, it is important how an agent cooperate with the others. However, it is difficult for an agent to cooperate appropriately in a dynamic environment, such as the RoboCup soccer. Therefore, Our team has two main features which allow appropriate cooperative activity, a cooperative protocol and a coach-agent[1]. With these features, our team can cooperate with each other in such environment.

1 Introduction

One of the important problems in a multi-agent system is how an agent cooperate with the others in order to achieve the goal of the system[2]. However, it is difficult for an agent to cooperate appropriately in a environment where the circumstances and the role of the agent often change. Therefore, precise circumstantial judgment and flexible strategy are necessary so that the agent cooperates appropriately in a dynamic environment.

Our team has two main features which allow appropriate cooperative activity in such an environment. The first feature of our team is concerned with to a Strategy Relay Cooperative Protocol. More specifically, when soccer players perform cooperative activity, they communicate to each other by broadcasting their playing strategy to the other players. Second, our team has a coach-agent. The coach-agent can obtain all the information about the game. Thus the coach-agent can make appropriate judgment during the game as well as select a suitable team formation according to the circumstances. Furthermore the coach-agent evaluate the player's strategy. The coach-agent informs the players about the result his evaluation, therefore each player can create an appropriate strategy according to the circumstances.

With the features mentioned above, the players can cooperate with each other in several situation.

2 Strategy Relay Cooperative Protocol

Here, we introduce the Strategy Relay Cooperative Protocol which allows the agents to cooperate in a dynamic environment. In this protocol each agent communicates with the others by broadcasting a Cooperative Strategy and making cooperative activity. In the Cooperative Strategy the agents in an organization cooperate with the other agents in order to solve problems. A cooperative knowledge is created based on the previous strategies used during the game. Each player has such strategy consisting of some steps.

When an agent solves problems in cooperation, he decides the Cooperative Strategy by using the cooperative knowledge, and broadcasts the strategy to the agents around him. The agents which have received the strategy will then evaluate it, and execute its next step or different strategy related to it. In this way, the agents perform dynamic cooperative activity. At this point, a cooperative group formed by the agents in the message arrival range is created dynamically. This is the group of agents which will perform the cooperative activity. Thus, the agents dynamically form a group in order to solve occurring problems. In figure 1, we illustrate a situation of cooperative activity using this protocol.

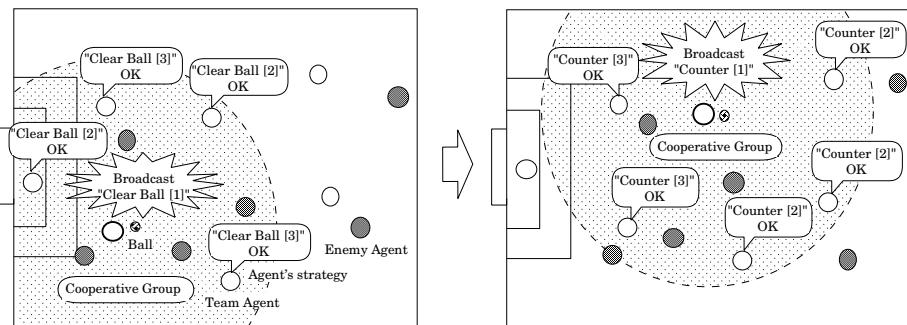


Fig. 1. Strategy Relay Cooperative Protocol

Our team can perform cooperative activity in a dynamic environment by using the protocol described above. Furthermore, by broadcasting the Cooperative Strategy, agents can perform cooperative activity by themselves. In addition, by creating the cooperative group dynamically, our team can compose a strategy of organization according to the circumstances.

3 Coach-agent

Our team has a coach-agent. The coach supports cooperative activity among player agents.

Suggestions of the coach are based on the following:

1. defense line

Our team takes a flat line defense. During the game, the coach checks the formation of an opponent team by analyzing their attack patterns. The attack pattern of the opponent is defined by y-coordinate where the opponent breaks our defense line.

The following cases are assumed that the opponents break our defense line.

- Our defenders spread in a line. (The interval of our defender is 7.5m at most.) The average of their x-positions is regarded as our virtual defense line.
- The ball crosses the virtual defense line toward our goal after the opponents pass or dribble.

And, in order to classify the attack patterns of the opponents, the soccer field is divided into 10 areas (figure.2).

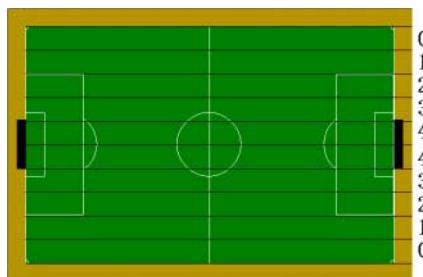


Fig. 2. classifying field

For example, when the opponents attacks along the side (0 and 1 in figure 2),the defense formation will be changed to a wide one.

2. Team formation in response to player status

A robust multi-agent system is a system that the agents compensate troubles that they did. In soccer, lacking their stamina or dead lock situations are examples of agents' troubles. The coach checks the player's motion, and change defensive formation when our agents lose their stamina and our team is defeated.

3. Set play

When play_mode is our goal_kick, the coach analyze who looks like to receive the ball. And, tell the goalie it. The goalie will judge the pass direction by himself, taking the message into consideration.

When the play mode changes into “play_off”, the coach will also send the information mentioned above.

The coach can evaluate its suggestion by checking successive plays, and change its own strategy of advice.

4 Evaluation of strategy by a coach-agent

A player strategy may not be always effective to the opponent type. Therefore, the coach-agent evaluates the player strategy and suggests the more suitable strategy according to the circumstances. In the following, we give a description of the steps for strategy evaluation.

1. A player performs a strategy by using the Strategy Relay Cooperative Protocol.
2. The coach-agent evaluates the strategy based on the information sent by each player.
3. The coach-agent sends to the players the result of his evaluation.
4. The player modifies the strategy by himself.

A player can execute the strategy according to the opponent.

5 Conclusion

In this paper, we described three methods of flexible cooperative action.

In the first method, a coach-agent analyzes the circumstances of the game and advises the team formation to the players. In the second one, as a result of using the Strategy Relay Cooperative Protocol, an agent can perform cooperative activity in a dynamic environment. In the third one, a player can execute a suitable strategy to the opponent based on the strategy that the coach-agent evaluates. Evaluation of strategy by a coach-agent is now under-work. To complete such implementation as well as perform experiments under cooperative action remain the objective of our research.

References

1. Soccerserver Manual, in <http://www.dsv.su.se/johank/RoboCup/manual/>
2. Nobuhiro Ito. A Description-Processing System for SoccerAgents,
RoboCup98:Robot Soccer World Cup II.

PaSo-Team 2000

Carlo Ferrari, Francesco Garelli, Enrico Pagello

Dept. of Electronics and Informatics, The University of Padua, Italy

1 Introduction

Following the experience done in previous competitions, it has been developed the 2000 version of PaSo-Team (The University of PAdua Simulated Robot SOccer Team), a reviewed release of Paso-Team99. During the RoboCup '99 competition in Stockholm some teams suffered synchronization problems with the soccer server: these problems greatly influenced their performances and prevented them from playing successfully. While developing PaSo-Team 2000 the main efforts were dedicated to better understanding timing and synchronization techniques for real-time multi-agent systems. Following the interesting experience done by Kostiadis in developing the Essex Wizzard team [1] we redesigned the synchronization procedures using the multi-threading paradigm. Solving synchronization in a multi-threading environment gives important theoretical hints to approach the coordination for those multi-agent systems made by thousand of very simple concurrent interacting modules. During the Stockholm competition PaSo-Team99 suffered another major problem regarding the actions a player must take when the game is stopped (i.e. when the ball is outside or when a team is offside). For example if a player has to throw-in the ball, he must go outside the field, turn toward the field and eventually kick the ball, performing different actions even if the state of the game doesn't change. As in a reactive architecture the current behaviour can change only when the game state changes, PaSo-Team99 introduced virtual states to ensure a change of behaviour. Instead in PaSo-Team 2000 we simplify the design of these actions introducing multi-step behaviours.

2 Team Development

Team Leader: Enrico Pagello *# (Associate Professor of Computer Science)

Team Members:

- Carlo Ferrari * (assistant professor of Computer Science)
- Francesco Garelli * (graduate student)
- Stefano Carpin * (graduate student)
- Andrea Sivieri * (undergraduate student)

Web page <http://www.dei.unipd.it/~robocup>

* Dept. of Electronics and Informatics, The University of Padua, via Gradenigo 6a, 35131 Padua, Italy

Ladseb-Cnr, C.so Stati Uniti 4, 35100 Padua, Italy.

E mail: {epv, carlo, garelli, shamano, tigre, gremlin, avatar, keter}@dei.unipd.it

3 Software architectures

PaSo-Team 2000 was developed using the Linux Operating System with the GNU C++ compiler. The used library was the standard GNU libc with Posix PThread Extensions; we didn't use other libraries like libsocket.

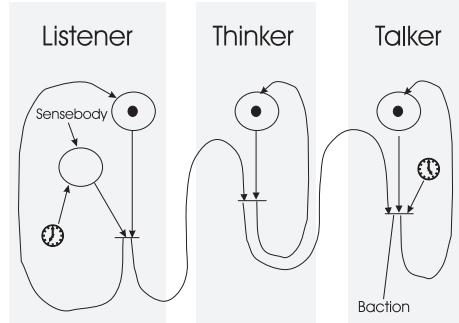


Fig. 1. Threads coordination in PaSo-Team 2000

PaSo-Team 2000 is based on three different concurrent threads:

- Listener: it gets the information from the soccer server and it manages the client timing and synchronization.
- Thinker: it updates the player's internal memory and it executes the suitable behaviour for the current simulation step.
- Talker: it manages the queue of the messages to be sent to the server .

The coordination between the threads follows the rules represented in Figure 1. The listener can be activated either by the sensebody signal received from the server or by a timeout signal. The timeout signal assures a correct activation even if the sensebody signal was lost for an excessive load of the server (or the net); of course the timeout is dynamically computed considering the average delay between the last received sensebodies. Although it is not represented in the figure, the listener module either can delay its activation to wait for a seeinfo message or it can be activated by a message which is not a sensebody. When the listener is activated it empties the socket queue and it re-arranges the possible messages received from the server. Finally it sends an activation signal to the thinker. When the thinker receives the activation signal from the listener it updates the player's memory, i.e. the internal representation of the environment, and it sends messages for the server to the talker thread. The talker thread forwards the messages to the soccer server assuring a minimum time gap between them.

4 The world model

Like PaSo-Team99, PaSo-Team 2000 uses Synthetic Visual Maps (SVMs) for motion control [5]. SVM are a concise representation of the free space around the player. The SVM maps each movement direction of the player with a boolean value that says whether that direction is free or prohibited. The SVM can be seen as a polar representation of the free space in a proper disk centered in the player. This representation can be easily updated at each sensing cycle, in order to consider new game elements that become important, either because they are moving towards the player or because the player itself is moving towards them.

5 Communication

During the 1999 competition it was clear that our team had reduced sensors information. The soccer server sends to the robocup clients only a partial knowledge about the environment around: in particular the server supplies information about the objects inside a 90° view only. The robocup simulation league is an environment where clients can't arbitrate their behaviours in a deterministic fashion: they can't choice the optimal behaviour in every situation because their representation of the current state of the server is not complete. Anyway the players can improve their knowlegde of the environment and therefore their arbitration function sharing their partial information. Infact the soccer server provides the command *say* by which a singular player can broadcast a message to their mates every simulation step. In Paso-Team 2000 we introduced a communication layer between the agents just to insure a better world model construction; we did not use the *say* command to coordinate the clients although. We confirmed the idea of not realizing any coordination via explicit communication at behaviour level: the communication layer involves only the representation of the environment and it doesn't affect the selection of current beaviours. The communication layer uses a token-based protocol because the soccer server doesn't allow many players to send messages concurrently. At every simulation step only one player, the owner of the token, can broadcast messages to their mates; we called this player *observer*. The observer should be the player who owns the best information about the environment for the current game situation. Hence the observer should have good information about the objects around the ball (where the game is) but he should not be involved in the current action because he has not to be the ball owner. Moreover during the match the observer changes because the relative position of the players change; at every step the choice of the next observer should be done by the player with the best knowledge of the server state, i.e. the current observer. According to these requirements we developed a protocol to distribute the knowledge between the clients: at every step the observer send to the mates his representation of the environment and the next observer id-number. When every client receives the message he integrates his representation with the observer's one and if required he becomes the new observer. Unluckily this protocol is not robust enough in the Robocup environment where

the delivery of the packets is not assured. Infact because of network problems the observer's token can be lost. To overcoming this problem we introduced in the communication layer a monitor module. Every player checks periodically if an active observer is present in the team; if the check fails he starts a procedure to elect a new observer.

6 Skills

In PaSo-Team 2000 we introduced multi-step behaviours. Multi-step behaviours are complex actions which can be completed in many simulation steps: they proved to be very effective in stopped game situations (throw-in, catch...). A multi-step behaviour can be aborted when a critical event (like either the lost of the ball or an offside) happens by throwing a C++ exceptions.

7 Conclusions

In the PaSo-Team'99 project we experimentally investigate how much the reaction schema for intelligent agents team must be integrated with some kind of high level reasoning. In PaSo-Team 2000 the major emphasis was on synchronization issues that were solved introducing multi-threading. The overall software architecture was redesigned and some kind of explicit communication was introduced to enhance the accuracy of the world representation procedures.

Acknowledgements

This research work could not be done without the enthusiastic participation of the students of Electronics and Computer Engineering Undergraduate Division of Padua University Engineering School. Financial support has been provided by both CNR, under the Special Research Project on "Real-Time Computing for Real-World" and MURST, under the 60% and 40% Grants.

References

- [1] Kostiadis K. and Hu H., "A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition" In Veloso M., Pagello E. and Kitano H., editors, RoboCup-99: Robot Soccer World Cup III. Springer Verlag, Berlin, 2000.
- [2] H-D. Burkhard, M. Hannebauer, J. Wendler, "Belief-Desire-Intention Deliberation in Artificial Soccer", AI Magazine, Fall 1998.
- [3] E. Pagello, A. D'Angelo, F. Montesello, F. Garelli, C. Ferrari, "Cooperative behaviors in multi-robot systems through implicit communication". Robotics and Autonomous Systems, 29 (1999), 65-77
- [4] P. Stone, "Layered Learning in Multi-Agent Systems", Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, December 1998.
- [5] C. Ferrari, F. Garelli, E. Pagello, "PaSo-Team99" In Veloso M., Pagello E. and Kitano H., editors, RoboCup-99: Robot Soccer World Cup III. Springer Verlag, Berlin, 2000.

Sharif-Arvand Simulation Team

Jafar Habibi, Ehsan Chiniforooshan, Majid Khabbazian, Mahdi Mirzazade,
MohammadAli Safari, HamidReza Younesi
{habibi@,{chinif,khabbazi,younesy,msafari,mirzazad}@linux.ce}.sharif.ac.ir

Sharif University Of Technology

1 Introduction

Like most teams SharifArvand consists of three parts: WorldModel, skills, and strategy. Duty of the WorldModel is getting received messages from the Soccer Sever (SServer[1]), creating a virtual model of the game field and updating the information about each object. Other layers of the program will take the necessary information for playing soccer from the WorldModel in a convenient format. Also any command sent to the server should be reported to the WorldModel. Other parts of the player program are implemented in a multilevel manner. In other words, in the lowest level there are simple skills which only use the WorldModel for undertaking their duties, whereas more complex skills in higher levels use the WorldModel and the lower level skills to achieve their goal. Finally, a strategy uses these skills for playing a soccer game. So, a strategy is the highest level skill. Therefore, a strategy should create, control and destroy skills to play a successful soccer game.

2 WorldModel

The WorldModel is to know all the information about all the existing objects in the soccer field. For representing the amount of error for each data in the WorldModel, each information unit has a probability density function to show its correctness.

2.1 Information Related to Other Objects

To find information regarding other objects, WorldModel can use one or more of the methods explained below:

- Finding this information directly from the received visual information.
- Simulating the motion of the objects (players and the ball) in the case that server has not sent information about those objects, the information is not complete or it is not exactly correct.
- Receiving a message by a player from a teammate. Because sending messages should be done via the WorldModel, and the WorldModel uses the extra empty part of the message to inform other players its own information, when the WorldModel receives a message, it can update some of its information according to that message.

- For the objects that their types are not known because of the long distance, such as players that their uniform number or their team is unknown, the WorldModel uses Finding the Maximum Weighted Matching algorithm[3] to assign each data item to an object (of course it considers and computes the correctness probability in this case too).

Weighted Matching acts on a weighted complete bipartite graph[3] like G with parts X and Y such that $|X| \leq |Y|$, and gives a perfect matching with maximum weight. A perfect matching is a matching (a set of none-neighborhood edges) that covers all vertices of part X . For our purpose, those objects that have been seen in previous (current) cycles creates vertices of Part X (resp. Y), and weight of edges are calculated by considering distance between objects and specified parameters of objects. for example if P_1 denotes a player with number 4 and unspecified team number who can be seen in the current cycle and P_2 denotes a player with number 4 and team number 1 who is seen in the past cycles, P_1 and P_2 can be identical if distance of them does not exceed a limit. So weight of edge connecting them will be computed based on their distance.

2.2 Positioning

We have used an algorithm which acts well in a noisy environment. This method can be used in every environment with arbitrary noise distribution for example in the Middle Size League.

Consider a robot has been located in an environment in which are some fixed objects called flags(let N_f to be the number of them). This robot observes these flags with some noise. Received Visual information by the robot includes Distance(i) and Angle(i) which are respectively, distance and angle with respect to flag i .

For each type of noise one can obtain $P_i(x, y) = f(\text{Distance}(i), \text{Angle}(i), x, y)$ which is the probability density function of existence in $\text{position}(x, y)$ concerning to received information of flag i . Now, we decide to obtain $P(x, y)$ which is the probability density function of existence in $\text{position}(x, y)$ concerning to all received visual information. This function is obtained from P_i s. We could not find a precise formula for $P(x, y)$, but, we suggest a formula which works well practically, however, we explain the reason of using this formula below (h and w are the height and width of the soccer field). Suggested formula:

$$P(x, y) = \frac{\prod_{i=1}^{N_f} P_i(x, y)}{\int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{w}{2}}^{\frac{w}{2}} \prod_{i=1}^{N_f} P_i(x, y) dx dy}$$

$P(x, y)$ must at least have following properties, all of which the above suggestion satisfies:

- $\int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{w}{2}}^{\frac{w}{2}} \prod_{i=1}^{N_f} P_i(x, y) dx dy = 1$ (It is trivial that each P_i must also have this property).
- $\forall i, x, y : (P_i(x, y) = 0 \Rightarrow (P(x, y) = 0))$

- $\forall i, x, y : (P_i(x, y) = 1 \Rightarrow P(x, y) = 1)$ (It is worth to mention that if we have $P_{i1}(x, y) = 0$ and $P_{i2}(x, y) = 1$ for a (x, y) , then we call them inconsistent and in this case the suggested formula is ambiguous ($\frac{0}{0}$)).

Now, we will describe the usage of $P(x, y)$ for robot positioning. Suppose the approximation of robot positioning be Δ . Then, the best position for the robot is the position with the most probability. In other words, it is the position that $\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} P(x + tx, y + ty) dt_x dt_y$ takes its maximum value in it.

Since the computing of $\int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{w}{2}}^{\frac{w}{2}} \prod_{i=1}^{N_f} P_i(x, y) dx dy$ is time-consuming and also this expression is only a constant coefficient for $P(x, y)$, to find the position of the robot, one can find a (x, y) in which $\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \prod_{i=1}^{N_f} P_i(x + t_x, y + t_y) dt_x dt_y$ takes its maximum value. (We have used the simpler formula for this purpose: $\prod_{i=1}^{N_f} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} P_i(x + t_x, y + t_y) dt_x dt_y$)

Finding the maximum point is a slow algorithm in general case. The reason is that this function must be computed in so many points. Hence, we used some methods which does not consider all points. Hill Climbing[6] is one of the methods which we used in this case. In this method, we start with one point and we go to a better point while it is possible, and so this algorithm finishes when there isn't any better point in neighborhood of the current point.

3 communication

Arvand agents communicate with each other frequently. In fact anytime they feel that they have enough new information about their miniworld or other agents decisions (like passing or speeding down because of their low stamina) they will say a message including newest data that they have seen or heard. The only remaining problem is interpreting "having enough new information". Because saying a useless message will cause in preventing some teammates from hearing more useful messages. We used some statistical methods to determine when one must say a message. It is expected that each agent say about one message in each 3 cycles.

4 Skills

It is obvious that for achieving the main goal in the game there are many smaller and local goals that should be achieved. Therefore, skills are implemented for performing these intermediate duties. An important point that should be taken into consideration for implementing the skills is that at any time when a skill is running, we should be able to stop the skill and to activate another skill. All in all, we decided to implement skills in the manner that each skill is a class derived from the main class, TSkill. Also each skill should fulfill at least the main function 'TSkillState TSkill::NextCommand(TCommand &)'. This means that a skill will not send a command to the server itself, but instead it should

just return one command in each call. So a skill is similar to a finite automaton that in each call (step) changes . Of course this automaton can dynamically be changed. Thus, a complete execution of a skill is calling the NextCommand function of the skill successively and sending the commands to the WorldModel until the skill approximately or completely achieves its goal. So we can stop the current skill whenever we want and create a new skill and make it to be the current skill.

5 Strategies

As we said above, a strategy is a high level skill that uses other skills to play a soccer game. In order to do this, we can use many different algorithms. So we may have various strategies. A player can change his strategy in two ways. First, it can decide to change his strategy himself, corresponding to the state of the match. Another way is that a certain player analyzes the match and decides which strategy should be used and informs the other players of the decision. Of course, in the second one, on-line coach is the best choice. We preferred to use the second method for changing strategies. One of the advantages of the second approach is that all players will change their strategies simultaneously.

6 Team Development

6.1 Tactic Designer

One noticeable thing in designing the strategy of this game is that in various cases we can perform the decisions by a simple state-machine or a set of conditions-actions. For instance, suppose the player who has the ball in his kickable area. He knows that if he shoots, there will be a high probability of scoring a goal. so it will be reasonable to shoot. Direct implementation of this state-machine in the source code, in addition to its difficulty and error-proneness will result in a nonflexible code and so production of more and different tactics will be harder.

In developing the SharifArvand team we have used a tool called TacticDesigner. Using its visual user interface it is possible to design the state-machine part of the tactic according to the team's available skills. It is important to know that in the TacticDesigner we have provided the ability of using some parameters which their value is not static and it will be computed by the executive engine during the game.

References

1. E. Corten, K. Dorer, F. Heintz, K. Kostiadis, I. Noda, J. Riekki, P. Riley, P. Stone, T. Yeap, Soccerserver Manual
2. S. Y. Kung, Digital Neural Networks, PTR Prentice Hall, 1993
3. D. West, Introduction to Graph Theory, Prentice Hall, 1996
4. Tom M. Mitchell, Machine Learning, McGraw-Hill, 1997
5. A. Barrand E. Feigenbaum, The Handbook of Artificial Intelligence, W. Kaufmann, San Mateo, CA, 1981

SharifII Soccer Simulation Team

Dr. Jafar Habibi, Ehsan Foroughi, Mehran Motamed
Pooya Karimian, Hamed Hatami, Hossein Fardad

Sharif University of Technology

1 Introduction

This paper describes ParsAI, a team developed and working at Sharif University of Technology. It is mainly concentrated on two distinct works, an idea for decision making components and a simple but powerful easy-to-debug implementation of clients.

2 Special Team Features

Our team's power comes from its simplicity of implementation and powerful design based on layered architecture as described in the next section.

Another feature of the team is the usage of the simple but powerful method of weighted comparison in all components. This method is to some extent similar to a neural network except for the method of weights calculation.

3 The layered architecture of our agent

Handling large programming projects requires a well-designed architecture. A common method is to divide the problem into distinct layers, each of which uses the previous layers' services to provide some predefined services for next layers; therefore, precise definition of each layer is necessary.

Beside all its known advantages, this model allowed us to divide both the problem and the program into many parts and helped us to work perfectly in a group, because it fixes and abstracts all the intermediate interfaces of components.

Figure 1 shows an overview of the architecture. The arrows indicate the way information is passed to other layers.

Our client model has a four-layered architecture as follows:

3.1 The Connection layer

This layer consists of a network socket and some low-level procedures for receiving sensor information from the server and sending commands to it. The main loop of the program simply gets messages one by one from server (using a blocking mode socket), processes them and passes the control to worldmodel layer to update data.

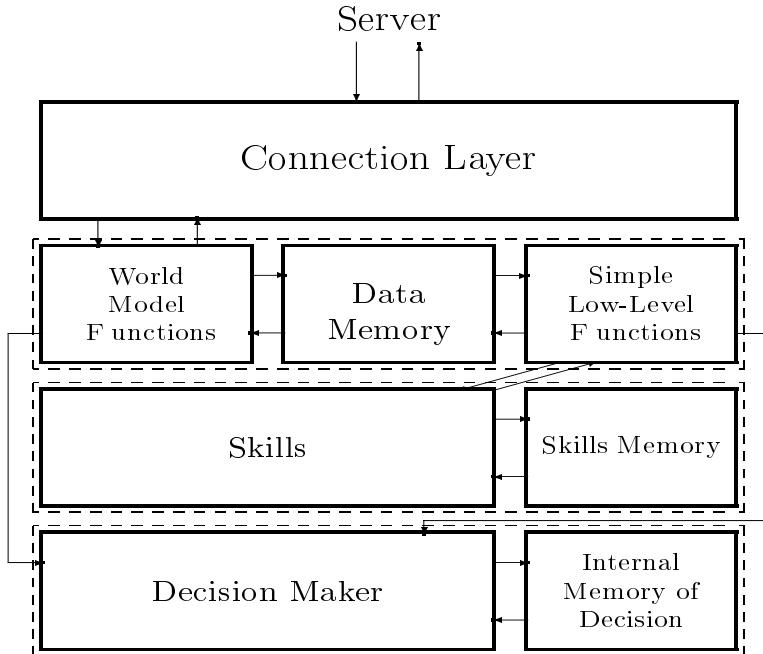


Fig. 1. The agent four layered architecture. From top : (1) Connection layer (2) World model layer (3) Skills layer (4) Decision making layer

3.2 The World Model layer

This layer is very important due to the dynamic nature of the problem. It provides and maintains the information which the agent needs for making decisions. This layer gets raw data from sensors, and parses them. Then it analyzes them to extract relatively useful data. In this part previous data is used for fields that we don't get any information. To store the data into memory we use a structure that stores the time of last updatation and also the reliability (certainty or the range of error) of the data. This information prevents the client from confusion that usually occurs due to old or incorrect data, allowing us to compare values gained from different sensors. Note that all world model updaters have to consider and update these values.

This layer also provides some intermediate functions for sending commands to the server. These functions update the worldmodel assuming the server will receive and change the world model at the end of that cycle. We used this method because the probability of “message reaching server correctly in synchronous time stepping model” is very high and all messages usually reach server within a cycle time.

For the parts of world which can't be seen, our agents use a combination of previous data and the data they receive from the hear sensor. In this combination the certainty values are of great importance that allow the result tending to be based on previous data in first few cycles and tending to hear sensor information

as the time passes on (from the last visual information about that object). In practice the accuracy reduces noticeably when this communication is turned off.

This layer also includes the Positioning component that finds the Player's absolute position in the field using the vision based sensors. In this part we have used the Hill Climbing algorithm to find the player's position that helped us reach the accuracy of less than 0.1-0.2 meter error in average compared to average error of 0.5-0.7 meters in the normal methods (a superb improvement).

3.3 The Skills layer

This layer consists of individual skills components that form the base of soccer playing behaviors of agents. We have implemented and used these skills :

- Passing: Includes Normal Pass, Forward Pass and Goaler Pass. Finds the best teammate considering all opponents by calculating a value for each possibility (the safeness comes here) and finding maximum value.
- Dribbling: Using many methods of dribbling like spinning the ball, running with ball and going through with a sudden movement; the main goal of this skill is to dribble forward toward a destination.
- Complex Kick: This skill, normally used in combination with Goal Scoring and Passing skills, is used to increase the speed of the ball with subsequent kicks toward some predetermined destination.
- Shoot to Goal: This skill considers the possibility of Shooting to opponent goal using both Normal and Complex Kicks and calculates the best direction.
- Getting the Ball: This skill calculates and finds the best way to reach the ball before opponents. When the Agent decides to have control over the ball, and the ball is not in his catchable area, he probably uses this skill.
- Positioning Without the Ball: This skill controls the movements of players without the ball in the field to achieve a good arrangement and positioning.

3.4 The Decision Making layer — Team's strategy

This layer makes decisions based on information received from other layers specially the world model layer. It also includes team strategies and agent reactions to normal and abnormal situations. Implementation is of great importance because of the complexity of this layer, and it needs a powerful abstraction to include all situations so we can easily add new ideas and reactions to it.

The real time, dynamic environment makes the decision making problem more challenging. So while keeping the simplicity we tried a new approach to the problem of choosing a good strategy : Weighted Comparison.

In this method, used both in skills and strategy layer, layer by layer, each component calculates a value for each action. This weight calculated (usually linearly) from values of underlying parts, represents the usefulness of the action so that the strategy layer can choose the best action by comparing this values.

4 Team Development

For the beginning of our team development we used “libsclient-4.00”, a common library for connection layer, first developed by Noda Itsuki, but little by little we customized the library and optimized it for server version 6.00. Also at the last step we added our powerful positioning system to it. Its source code is openly released and can be found in our web site.

We also wrote a debugger and a real-time world model visualizer for our agents during the team development that helped us a lot in the implementation.

Team Leader: Dr Jafar Habibi — email: `habibi@sharif.ac.ir`

Team Members:

Ehsan Foroughi (attend the competition)

Mehran Motamed (attend the competition)

Pooya Karimian

Hamed Hatami

Hossein Fardad

- `{foroughi,motamed,karimian,hatami,fardad}@linux.ce.sharif.ac.ir`
- Country: Islamic Republic of Iran
- All are undergraduate students of computer engineering in Sharif University of Technology

Web page <http://www.parsai.com>

5 Conclusion

We have decided to reduce the complexity of implementation while increasing the power of the team by avoiding case consideration and using formal, universal methods and designs. This layered design will remain as the base of our client.

In the mean time we are working on the improvement of the algorithms and completion of the parts that are not implemented yet. In the Connection Layer and World Model Layer we don't need any noticeable changes except for a better accuracy of the objects in the World Model. There remains some work in the Skills Layer and the main concentration is on Strategy Layer. Some learning based strategies are also planned to be tested in future.

Some new topics are introduced in relation to the decision making algorithm discussed in section 3. The most important of them is usage of the Genetic Programming method to determine the weights. As you may notice, the core of the decision making components is to determine the weights correctly and efficiently. This converts the strategy and decision component of our program into a table of quotients. So we can use Genetic Programming methods to find the efficient table. Note that the problem of writing a successful team is now converted to maximizing the value of a function from a table of inputs to a number representing the success of the team (teams performance).

Essex Wizards 2000 Team Description

H. Hu, K. Kostiadis, M. Hunter, N. Kalyviotis

Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
Email: {hhu,kkosti,mchunt,nkalyv}@essex.ac.uk

Abstract: This article gives an overview of the Essex Wizards 2000 team participated in the RoboCup 2000 simulator league. A brief description of the agent architecture for the team is introduced. Both low-level behaviours and high-level behaviours are presented. The design issues regarding fixed planning and reinforcement learning are briefly outlined.

1. Introduction

In the RoboCup [6] Simulator League, the Soccer Server [11] provides a virtual pitch, and simulates all movements of the ball and players. Players connect to the server via UDP/IP sockets over a local area network, and all communication must be carried out via the server. Communication via the server is heavily restricted, only one teammate can be heard at a time, and the range of the communication is limited. The player may only send one exclusive command to the server in a cycle of 100ms. If no command is sent the player will be idle for that period, and if more than one exclusive command is sent one is picked at random. Visual information is received from the server asynchronously, approximately every 150ms, although this depends on the quantity and quality of the information requested.

The server is responsible for generating 'visual' and 'auditory' information for the players, in the form of text strings, that give a symbolic higher level indication of the agents perceptions. It also takes the commands and their parameters from the player, such as power and direction for kicks. Information passed between the server and clients is susceptible to noise, a player may not kick the ball exactly as intended, and the perception of the pitch also contains errors.

In this article the focus is on the behaviour-based approach for the Essex Wizards team. More specifically, a brief description of the agent architecture is presented in section 2. Section 3 outlines both low-level behaviours and high-level behaviours being adopted. The framework for the fixed plan and reinforcement learning is briefly introduced in section 4. Finally a brief summary is presented in section 5.

2. Essex Wizards Agent Architecture

The main objective for the Essex Wizards team is to build a firm research platform for multi-agent systems research. Currently, a multi-threaded approach is adopted to achieve real-time performance [3][7], and a modular approach [4] is adopted in the overall agent implementation as shown in Figure 1. More specifically:

- Sensors -- It is executed in a separate thread [2]. Incoming visual and auditory information from the server is received and interpreted. The sensors only detect information from a limited area of the pitch, depending on the position and orientation of the player. The old information is passed to the Memory module. This allows the agent to use both current and past information to make decisions.

- **Actuators** -- This is a separate thread. It is responsible for sending the actions to the server at regular intervals to ensure that the player has an opportunity to act. When the sensor thread detects entry to a new cycle, the actuators start sending the queued actions.
- **Play Mode** -- This module keeps track of the current server play mode to ensure that player behaviour is appropriate to the current state of the game.
- **Parameters** -- This module holds game constants for both server variables, such as the width of the goal, and player options such as the passing power.
- **Memory** -- This module is responsible for keeping track of the current state of the pitch, and making that information available to other modules. The memory is updated explicitly when new information becomes available, based on new sensor information. In addition at the end of each cycle the state of the pitch for the following cycle is predicted based on the current state. The predicted information can quickly become out of date, particularly for moving objects, so items in memory have a confidence value associated with them. If an object has just been seen the confidence will be high, as more predictions are made the confidence drops. Below a certain confidence value the object is ignored.
- **Behaviours** -- Based on information from the Sensors, Play Mode, Parameters and Memory modules the Behaviours module selects the best course of action and sends the result to the Actuators module for dispatch to the server. This is the most complex module since it does the “thinking” for the agent.

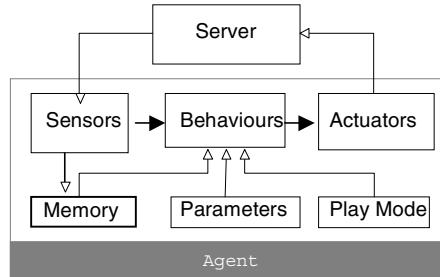


Figure 1 Proposed Agent Architecture

3. Agent Behaviours

The Behaviour module in each agent needs to carry out a number of tasks, including both the low-level behaviours such as moving and kicking, and the high-level behaviours such as selecting where to move to and which teammate to pass to [1], as shown in Figure 2.

3.1 Low-level Behaviours

At the lowest level any decisions made by the agent must be reduced to the core primitive actions provided by the server, i.e. *Kick*, *Turn* and *Dash*. In order to provide the options for high-level behaviours, extended primitives have been implemented, including

- *Akick* (Advanced Kick): It can deal with more complex situations by moving the ball in stages to a position where the desired kick can be made.
- *Move*: If an agent needs to move, it must mix turns and dashes and maybe avoid collisions with other objects to reach the desired location.

3.2 High-level Behaviours

The high-level tactical behaviours are build on top of low-level primitive behaviours, and are currently implemented as a hybrid of Q-learning and rule based decision-making [9].

- *Dribble*: Dribbling involves intermixing kicks, dashes, and possibly turns to move the ball across the pitch, while keeping the ball under the agent’s control.

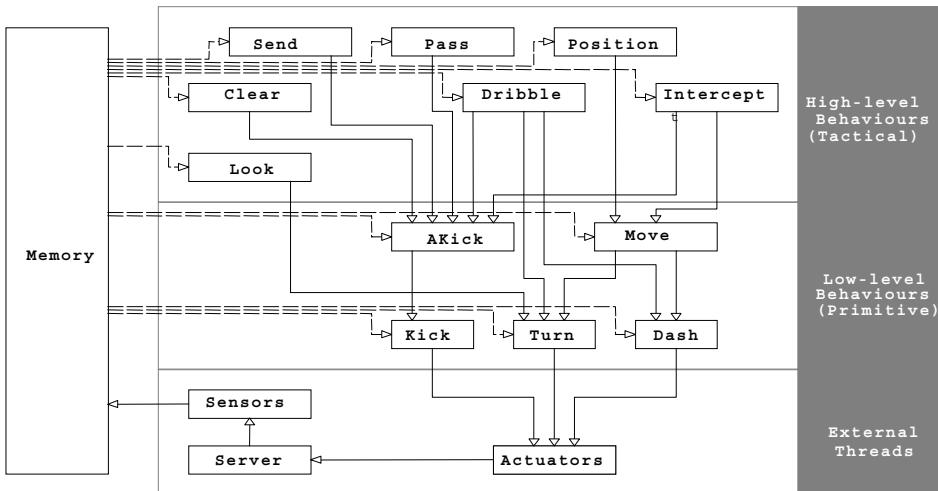


Figure 2 Agent Behaviour Architecture

- Look/Scan: If the ball has not been seen recently, the agent must scan for it by executing a number of turns or turn-neck commands. In some cases the agent might decide to improve its perception and execute a scan.
- Intercept: This involves predicting the trajectory of the ball and a suitable interception point to move to. Currently there are two modes of interception. The choice between them is based on the agent's role, position on the pitch and velocity of the ball.
- Clear Ball: If the agent is near the home goal and in danger of losing the ball to an opponent, and no teammates are in a position to receive a pass it may be necessary to clear the ball to safety. This involves kicking the ball, using the *AKick* behaviour.
- Send Ball: This occurs when the agent is close to the opposing goal. In this situation it attempts to beat offside traps and advance into the enemy half of the pitch.
- Pass Ball: Pass Evaluation analyses the locations of teammates and opponents on the field in order to decide if a good pass is possible. It returns which teammate should pass to, and an estimate of the success.
- Position Selection: The Position Selection behaviour (PSB) examines the current view of the pitch and uses it to suggest a good place to move to. Selection of such a position is a non-trivial task, requiring information about the current role of the agent and the state of the pitch. More details about PSB can be found in [5].

4. Fixed Plans and Reinforcement Learning

Since having a fixed plan for a situation that happens frequently is a good strategy, the Essex Wizards 2000 team has implemented several fixed plans in RoboCup2000. The framework for our fixed plans consists of three major components, namely *Triggers*, *Actions* and *Abort conditions*. The *Triggers* is used as signals to allow or forbid the actions or plans that are predefined. The *Actions* is a combination of low-level and high-level behaviours that are executed sequentially. The *Abort condition* is a safe guard to ensure the conditions of the environment are suitable for the fixed plan being executed.

The reinforcement learning has been adopted in our behaviour-based decision making process since it provides a way to programming agents by reward and punishment without needing to specify how the task is to be achieved [1][7]. Each time the agent receives inputs, it then chooses an action to send. The action changes the state of the environment

and also provides the agent with either a reward if it does well or a punishment if it does badly. The agent should choose actions that maximise the long-term sum of rewards.

It should be noticed that the agents in our implementation not only have different roles and responsibilities, but also have different sub-goals in the team. Hence, every individual agent tries to reach its own goal state, and cooperation emerges when the goals of all agents are linked together. The ultimate goal, scoring against the opposition, becomes a joint effort that is distributed among team members, see [7] for more details.

5. Summary

In order to satisfy all the necessary timing constraints for real-time simulated football-playing agents, a multi-threaded implementation has been adopted in the Essex Wizards team so that the agents can perform various computations concurrently and avoid waiting for the slow I/O operations [3][8]. Moreover the behaviour-based approach plays a key role in building the Essex Wizards 2000 team. A decision-making mechanism based on reinforcement learning enables co-operation among multiple agents by distributing the responsibilities within the team [7].

The current focus for our Essex Wizards team is on adaptive position selection, flexible fixed planning, optimal decision making, and multi-agent learning.

Acknowledgements: We would like to thanks the CMU'99 team who contribute their source code to anyone who works on the RoboCup simulation league. We have benefited from this in our implementation. This research was financially supported by the university RPF award DDP940.

References

1. Balch T.R., Integrating RL and Behaviour-based Control for Soccer, Proc. IJCAI Workshop on RoboCup, 1997.
2. Burns A. and Wellings A., Real-time Systems and Programming Languages, Addison-Wesley, 1997.
3. Butenhof R.D., Programming with POSIX Threads, Harlow, Addison-Wesley, 1997.
4. Hu H., Gu D., Brady M., A Modular Computing Architecture for Autonomous Robots, Int. Journal of Microprocessors & Microsystems, Vol. 21, No. 6, pages 349-362, 1998.
5. Hunter M., Kostiadis K., Hu H., "A Behaviour-based Approach to Position Selection for Simulated Soccer Agents", 1st European Workshop on RoboCup, Amsterdam, 28 May - 2 June 2000
6. Kitano H., RoboCup: The Robot World Cup Initiative, Proceedings of the 1st Int. Conference on Autonomous Agent, Marina del Ray, The ACM Press, 1997.
7. Kostiadis K. and Hu H., Reinforcement Learning and Co-operation in a Simulated Multi-agent System, Proc. of IEEE/RJS IROS'99, Korea, Oct. 1999.
8. Kostiadis K. and Hu H., A multi-threaded approach to simulated soccer agents for the RoboCup competition, IJCAI'99 workshop on RoboCup, 1999.
9. Mataric, J. M., Interaction and Intelligent Behaviour, PhD Thesis, MIT, 1994.
10. Nissanke N., Real-time Systems, London, Prentice Hall, 1997.
11. Noda I., Soccer Server: A Simulator for RoboCup, JSAI AI-Symposium 95: Special Session on RoboCup, 1995.
12. Stone P. and Veloso M., Team-Partitioned, and Opaque-Transition Reinforcement Learning, Proc. 15th International Conference on Machine Learning, 1998.

RoboCup-2000 Simulation League: Team KU-Yam2000

Harukazu Igarashi, Yusuke Yamauchi, Shuichi Iidai
Kinki University, Higashi-Hiroshima, Hiroshima, 739-2116, Japan

Abstract. In this paper we described our KU-Yam2000 team that participated in the simulation league of RoboCup-2000 Melbourne. KU-Yam2000 is characterized by soccer agents that are controlled by a hierarchy of actions. Actions in each level of the hierarchy are separated into modules, which are arranged in a form of a C-language library to be called from a main program.

1 Introduction

Robot soccer is a candidate for the standard challenging problems in Artificial Intelligence. Our two teams, Team Miya and Team Niken, participated in the simulation league of RoboCup97 Nagoya (Japan, August 1997) [1]. Moreover, we sent Team Miya2 to the simulation leagues of RoboCup98 Paris (France, July 1998) [2] and Team KU-Sakura2 to Robocup99 Stockholm (Sweden, August 1999) [3].

Team Miya was characterized by individual tactical play[4]. Individual tactical play does not require communication between the players, so the speed of passing was rapidly increased in RoboCup 97 games, and the team sometimes behaved as if it had been taught some tactical plays. Team Miya proceeded to the quarterfinal match and was one of the best eight teams in the simulation league.

In Team Miya2, a kind of communication between players was realized by using a "say" command so that a passer can make a pass to a receiver without looking around for receivers[5]. Consequently, Team Miya2 was one of the best sixteen teams in RoboCup98.

However, more tactical play is required for the following two reasons. First, the top teams of RoboCup98 showed very high-level skill in individual play. For example, we observed a speedy dribble while keeping the ball near the player's body and a safety pass without being intercepted by the opponent players. Second, the offside rule was introduced at RoboCup98. Thus forward players have to check whether they are in an offside position or not at all times. Some tactics are necessary to avoid the opponent's offside trap and to succeed in setting an offside trap against the opponent team.

For this purpose, we used communication between players for realizing the tactics in Team KU-Sakura2. We succeeded in moving forward players and defense players to a certain degree, which helped Team KU-Sakura2 to pass the qualifying round and proceed to the final tournament. However, our program became very complicated, and it seemed difficult to further improve individual skills in KU-Sakura2. Thus we decided to make a new team where all procedures are modularized to C-language functions after the RoboCup99 competition. Moreover, we improved the format of a player's world model and used it to estimate positions and velocities of the ball and all of the players. That is a basic concept in developing

Team KU-Yam2000.

2 Team Development

We describe the development of our Team KU-Yam2000 in this section.

Team Leader: Harukazu Igarashi, Associate Professor of Kinki University, Japan

Team Members: Yusuke Yamauchi, Undergraduate Student, and Shuichi Iidoi, Graduate student of Kinki University, Japan. H. Igarashi attended the RoboCup2000 competition.

WebPage: <http://www.ip.info.hiro.kindai.ac.jp/igarashi/registration/qualification.htm>

E-mail address: igarashi@info.hiro.kindai.ac.jp

3 World Model

Each soccer agent has a world model in Team KU-Yam2000. The world model consists of information on the ball and all players, including the agent itself. Figure 1 shows the formats of data used to store the information on the ball and players. Let obj_ball and $obj_player(i)$ { $i=1,\dots,22$ } be data areas for the ball and twenty two players, respectively.

In Fig. 1, time t is a time step when the data was obtained through analyzing the visual information that the agent received from a soccer server. Position (x, y) is the position and velocity (v_x, v_y) is the velocity of the object observed or estimated by the agent. The data obj_ball include flags f_p and f_v . The flags take a value of one if the position data and the velocity data were observed directly by the agent in its visual

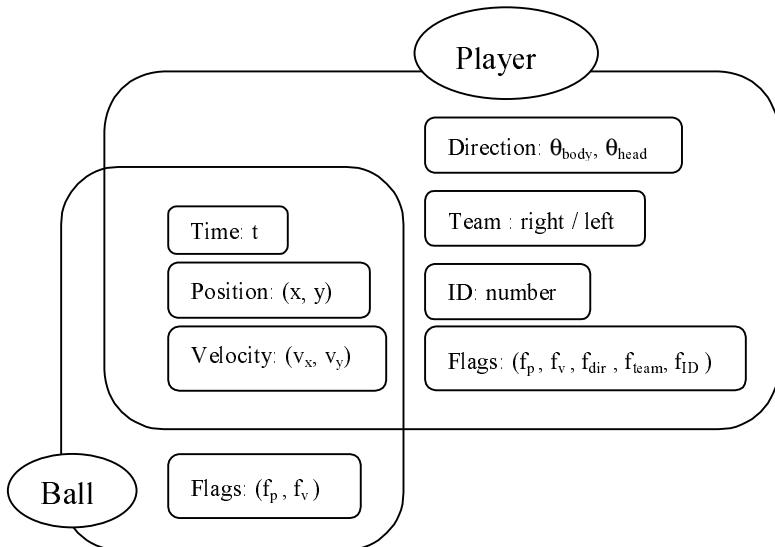


Fig. 1 Data structures for the ball and players

information. The flags take a value of zero if the data are not observed but estimated from the past data.

The data $obj_player(i)$ also include directions of the player's body and head, a team name, and a uniform number. However, the data of a player's team name and uniform number are often missing according to the distance from the observer to the objective player. We prepared the flags f_{team} and f_{ID} for a team name and a uniform number to distinguish all players. The flags f_p , f_v , and f_{dir} show whether the corresponding data are observed or estimated as in the case of a ball.

The items of an agent's world model shown in Fig. 1 are determined by the following algorithm. First, an agent receives its visual data and calculates positions, velocities and directions of the objects included in the visual data. If the object is a ball, then the data are stored to obj_ball immediately. However, if a player's data are included, the data are buffered.

Second, if the agent can recognize the team name and uniform number of the player, we set the flags for the team name and uniform number to one, otherwise zero.

Finally, we convert and store the data buffered in the first step to $obj_player(i)$. The data can be easily assigned to the corresponding $obj_player(i)$ if the player's f_{team} and f_{ID} are one. If the f_{team} is one and f_{ID} is zero, we assign the player to an $obj_player(i)$ that has not yet been determined in the player's team. If there are more than two candidates, we take into account the immediately previous positions of the candidates. If both flags are zero, we only use the previous positional data of the candidates.

4 Hierarchy of Programming Libraries

KU-Yam2000 has three programming libraries to make the programs that control an agent. There is a hierarchy among the libraries, as shown in Fig. 2.

The library at level 1 is based on *libsclient*, which was developed by Noda[6] for

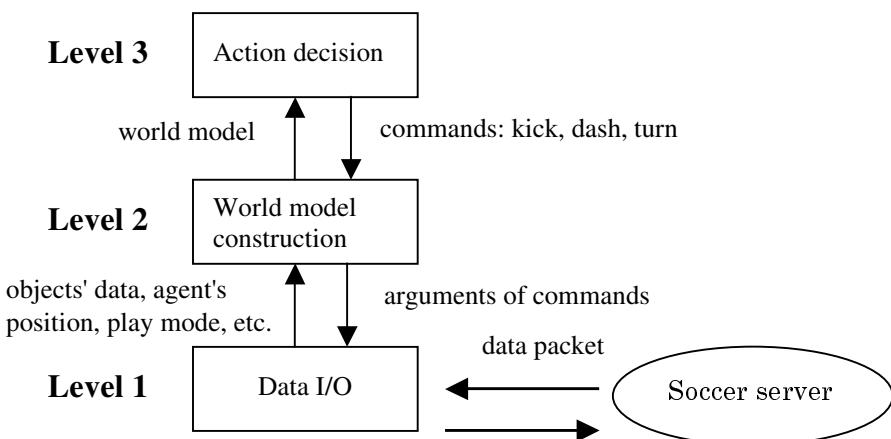


Fig. 2 Hierarchy of libraries used in KU-Yam2000

Soccer server 3.xx and rewritten by Onaizan and the ISIS team[8] for server 4.xx. This library is a set of C-language functions for communicating with a soccer server, analyzing visual and auditory information data, calculating an agent's current position, and recognizing the current play mode.

The library functions at level 2 make an agent's world model as described in Section 3 and issue the basic action commands *kick*, *dash* and *turn*. These commands need some arguments, for example, Power and Direction for a kick command, Power for a dash command, and Moment for a turn command. The value of Power in a kick command is adjusted to avoid wasting the stamina of a player because a kick command decreases the value of a player's stamina and there is an upper limit on the ball's moving speed. Thus an overly large Power value wastes a player's stamina. The library functions at level 2 determine the arguments' values of the basic commands and issue the commands by calling library functions at level 1 to a soccer server.

The library functions at level 3 convert a tactical play, such as a safety pass, a dribble, a shoot or a centering pass, to a series of basic commands. We have already proposed a hierarchy of actions and an action tree for this purpose in Team Miya[4]. We are now developing the library functions at level 3 to control an agent by using the hierarchy of actions and the action tree.

5 Future Work

We are developing basic library functions to improve the soccer clients in KU-Sakura2 that participated in RoboCup99. We will add team play skills, such as passing with communication[5] and adaptive movement of the defense and offense lines, after completing the library functions in this team description. Moreover, we plan to apply an action decision based on optimization [7] to a soccer agent.

References

- [1] <http://www.robocup.v.kinotrope.co.jp/games/97nagoya/311.html>
- [2] <http://www.robocup.v.kinotrope.co.jp/games/98paris/312.html>
- [3] <http://www.robocup.org/games/99stockholm/313.html>
- [4] Igarashi, H., Kosue, S., Miyahara, M., and Umaba, T.: Individual Tactical Play and Action Decision Based on a Short-Term Goal -Team descriptions of Team Miya and Team Niken. In: Kitano, H.(ed.): RoboCup-97: Robot Soccer World Cup I, Springer-Verlag (1998) 420-427
- [5] Igarashi, H., Kosue, and S., Miyahara, M.: Individual Tactical Play and Pass with Communication between Players -Team descriptions of Team Miya2. In: Asada, M., and Kitano, H. (eds.) : RoboCup-98: Robot Soccer World Cup II, Springer-Verlag (1999) 364-370
- [6] <http://ci.etl.go.jp/~noda/soccer/client/index.html#Library>
- [7] Igarashi, H., and Ioi, K., "Path Planning and Navigation of a Mobile Robot as Discrete Optimization Problems", Proc. of The Fifth Int. Symp. on Artificial Life and Robotics (AROB 5th '00), Oita, Japan, January 26-28, 2000, Vol. 1, pp. 297-300.
- [8] <http://www.ida.liu.se/~frehe/RoboCup/Libs/libsv4xx.html#libsclient>

Harmony Team

Hiroyuki Iizuka, Masatoshi Hiramoto, Hidenori Kawamura,
Masahito Yamamoto and Azuma Ohuchi

Research Group of Complex System Engineering
Graduate School of Engineering, Hokkaido University, Sapporo, Japan

1 Introduction

In the simulation part of soccer server, there are many strong teams made by detailed designs of hand-coded program techniques based on top-down approaches. However, it is exciting if it can be possible to design strong soccer team by the computer program itself. It is difficult but interesting and challenging to let a computer program to design intelligent program by itself. The aim of us is to design the strong soccer agent team based on evolutionary technology and to overcome champion team by autonomous self-designed agents. As first step of such aim, we construct co-evolutionary soccer agent team with hand-coded primitive actions.

2 Special Team Features

In order to construct more complex and flexible actions, we adopted meta-level decision-making process. Meta-level decision-making process has a tree structure based on if-then rules using primitive actions, as shown in Fig. 1.

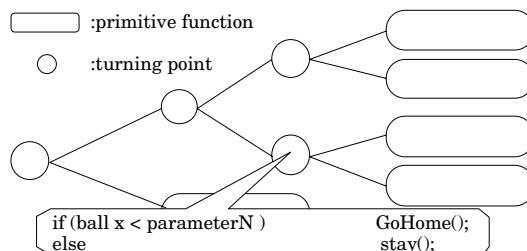


Fig. 1. Meta-level decision-making process.

For reasonable evolutionary search, we prepare four extended primitive actions, i.e., run, dribble, pass and shoot. On the other hands, meta-level decision-making is related to a coordinated strategy among agents, and execute combinations of primitive actions according to if-then rule based decision-making. Each

if-then rule has some parameters which effect a condition part, and behavior of meta-level plays is controlled by such parameters. Meta-level parameters strongly effect the strength of team. This framework seems reasonable for self-designed agents.

Therefore, we try to let our agent team to evolve autonomously by adjusting meta-level parameters based on co-evolutionary technology often used in genetic algorithm techniques.

As similar to the general genetic algorithm, we have a population consists of some individuals. An individual is a team consists of 11 heterogeneous players. Each player has its own role and some parameters influencing to its play style. Beginning with a random initial population, we pick up one individual and applied it to a genetic operator (slightly changing only one parameter in a decision-making process), and perform the game between them in each generation. According to the results of games and behavior based fitness values, the better individual is replaced to another one. After several generation, we expect to obtain good teams.

3 World Model

Agents store information about ball, team mates and opponents. Each object has information about position, speed and acceleration with confidence value like CMUnited do. We refer to CMUnited world model.

4 Coach

We don't use coach client for players to select actions or change strategies. However, we applied coach client to control co-evolution procedure.

5 Communication

Our agents do not communicate at all. We expect that agents acquire effective behaviour without communication.

6 Skills

The four extended primitive actions that we prepare are,

run This action represents a movement from the current position to the player's destination, and is characterized by the destination and a speed for running. According to the destination and the speed, this run action works as a go-home-position function or chase-ball function and so on.

dribble This action is one of the most important actions in order to keep a ball and to get a score. Dribble action is performed based on a speed and a destination of player's dribbling. Speed represents the extent of time that it takes

to move from a current position to the destination with a ball, and is determined according to the stamina of the player or chances to get a score. In the case of high speed, the ball is far from the player because the player kicks the ball more strongly and the risk of keeping a ball increases.

pass Generally, a pass is realized by cooperation between two players. However, this primitive pass action is not defined such advanced technical skills for players. This pass action only defines that the player kicks a ball in a certain direction. According to the direction, this pass action works as a back-pass function or offensive-pass function and so on.

shoot This action is similar to the pass action. The player kicks a ball to the goal in order to get a score under shoot action. There are some cases for shooting. For example, the player wants to shoot as soon as possible regardless of the speed of the ball or on the contrary the player wants to shoot as strongly as possible. This shoot action represents some functions like immediate-shoot or strong-shoot by changing the parameter module.

7 Strategy

We execute Co-evolutionary procedure about 600 matches.

To measure latest team, we held matches that latest team faces ancestral teams. In almost all matches, latest team get scores more than ancestral teams get scores. This instructs that latest champion is superior to ancestral champions, and co- evolutionary procedure performs successfully.

The strategy obtained finally as whole team is following thing. When defender possesses ball, defender dribbles to opponent area, and shoot in opponent area. Offender is near opponent goal. Offender picks up the ball shoot by defender and shoot to goalmouth. When Midfielder possesses ball, if there is no teammate near midfielder, midfielder dribbles, otherwise pass the ball to near teammate. And offender receives ball, then immediately shoot to goalmouth.

8 Team Development

Team Leader: Hiroyuki Iizuka

Team Members:

Hiroyuki Iizuka

- Graduate School of Engineering, Hokkaido University
- Japan
- Graduate student
- Attend the competition

Masatoshi Hiramoto

- Graduate School of Engineering, Hokkaido University
- Japan
- Graduate student
- Attend the competition

Hidenori Kawamura

- Graduate School of Engineering, Hokkaido University
- Japan
- Instructor
- Did not attend the competition

Masahito Yamamoto

- Graduate School of Engineering, Hokkaido University
- Japan
- Associate professor
- Did not attend the competition

Azuma Ohuchi

- Graduate School of Engineering, Hokkaido University
- Japan
- Professor
- Did not attend the competition

9 Conclusion

In order to co-evolve teams (individuals) efficiently, we have to evaluate teams appropriately. However, it is very difficult to measure the goodness (fitness) of a team, because a good team not always wins the game due to much noises, or uncertainty in the game. To get the correct fitness value of the team, many games must be played. However, because computational costs for playing many games are very large, we abandoned to calculate the fitness value of a team by only results of a few games, and adopt the fitness function based on behaviors of players

We examine relations between the goodness of teams and some behaviors. From the result of experiments, we expect to find out that the effective fitness function.

References

- [Stone 2000] P. Stone, M. Veloso, and P. Riley, "The CMUnited-99 Champion Simulator Team", RoboCup-99: Robot Soccer World Cup III, M. Veloso, E. Pagello, and H. Kitano (eds.), Springer Verlag (2000).
- [Luke98] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler, "Co-evolving Soccer Softbot Team Coordination with Genetic Programming", RoboCup-97: Robot Soccer World Cup I, H. Kitano (eds.), Springer Verlag (1998).
- [Pollack 1998] J. B. Pollack and A. D. Blair, "Co-Evolution in the Successful Learning of Backgammon Strategy". Machine Learning 32, pp. 10-16 (1998).
- [Murata 1998] T. Murata, K. Suzuki and A. Ohuchi, "Dynamic Position Arrangements of Soccer Agents with Genetic Algorithms", Journal of Japanese Society for Artificial Intelligence 14(3), pp. 446-454 (1998)

TakAI

Tetsuhiko Koto

The University of Electro-Communications

1 Introduction

The "Gullwing" developed by Tetsuhiko Koto since March 1999 is now assigned to the official name for the second strongest team of Takeuchi Lab. It gained the third position in the Japan Cup Open '99 [1] held in May 1999. Since then, it has been thoroughly revised and it participated the Japan Autumn Camp[2] held in 1999, with the official name YowAI-2 at that time.

2 Architecture

Each agent (player) of TakAI has the architecture shown in figure1. I made all the programs from scratch. The language is C++.

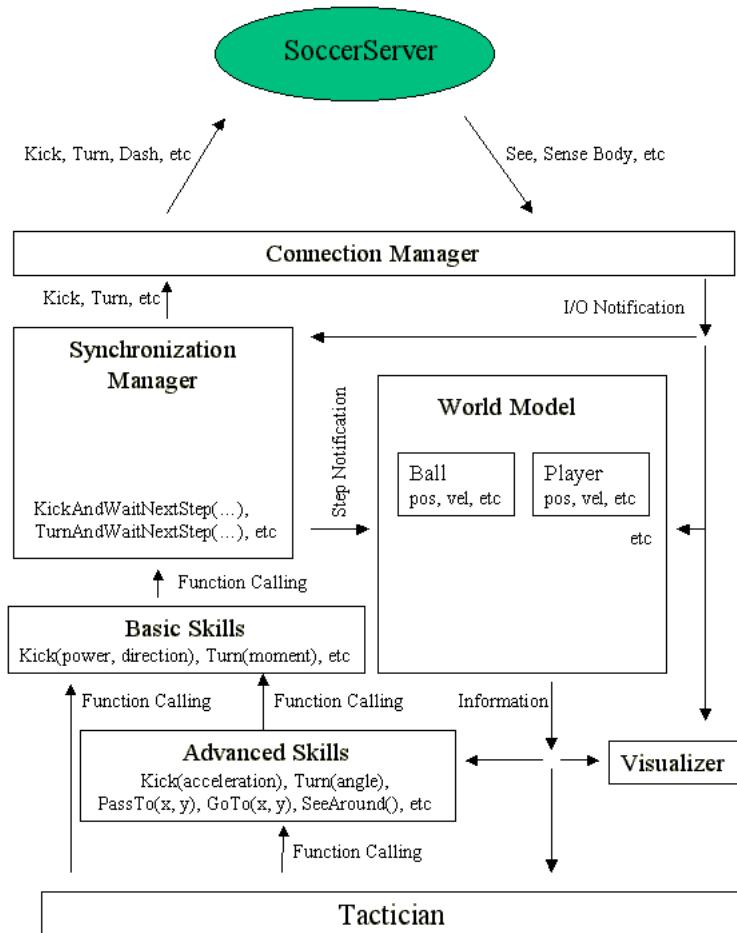
The Connection Manager manages all the communication with the Soccer Server[3]. It sends commands such that kick, turn and dash issued by the Synchronization Manager to the Soccer Server, and receives sensory information such as sense_body and see. Every communication between the Connection Manager and the Soccer Server is notified to other modules as I/O Notifications.

The Synchronization Manager synchronizes the internal clock of the agent and the clock of the Soccer Server. It provides functions such that KickAndWaitNextStep which enables the agent to issue a command and wait for the next clock when it can issue another command without overriding the command it has issued. These functions order the Connection Manager that their own commands be sent the Soccer Server, and watch I/O Notifications to be sure that the Soccer Server clock advances by one step.

The World Model maintains the positions, velocities, and accelerations of its own agent, other agents, and the ball. It watches I/O Notifications, gets the sensory information (visual, auditory and bodily) and calculates the positions, velocities, and accelerations of agents and the ball based upon it. Of course, it reflects the expected effect of the commands the agent issues. It tries to simulate the soccer field as exactly as the Soccer Server does, based upon the information obtained by the Synchronization Manager.

Various Skills are implemented on the basis of the Synchronization Manager and the World Model. The Skills include simple reflective ones, and higher ones such that skill for running to the position designated as an absolute coordinate.

The Tactician decides the player's short term tactics based upon the World Model, and the Skills make the tactics be fulfilled.

**Fig. 1.** Architecture of TakAI

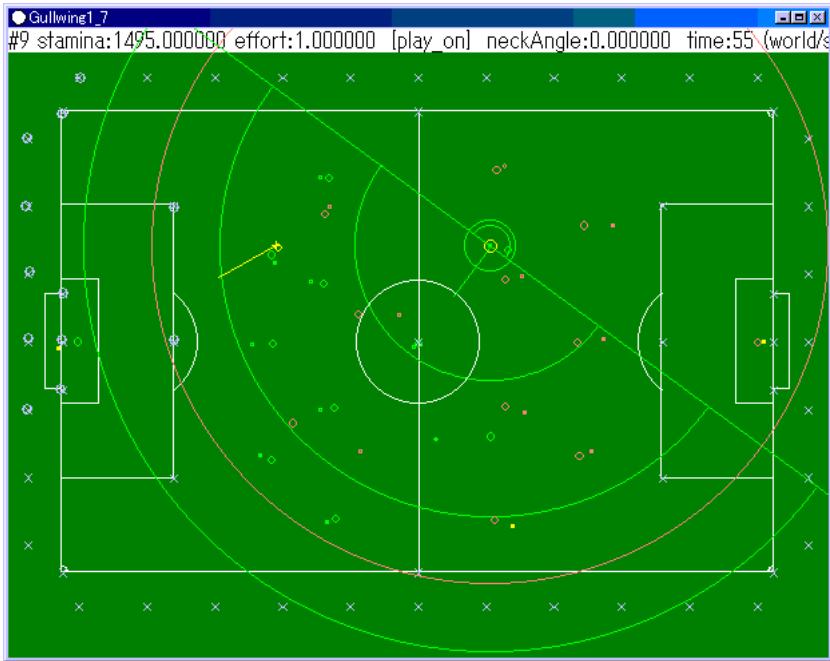


Fig. 2. Screenshot

The Visualizer makes the internal World Model be able to be seen on the display screen, mostly for debugging aids (See figure2).

All the algorithms (skills and tactics) are handmade and the players learn nothing and do not grow up.

3 Play Algorithm

The TakAI tactics can be summarized as follows:

(1) If the ball is too far from the player to kick,

It calculates which one of its teammates can touch the ball first. If it is the self, then it tries to get the ball, else it runs to the position if someone else tells it so, else it tries to take its desirable position; basically, it tries to keep apart from other teammates, not to make the team too crowded in a particular area.

(2) If the player is close enough to the ball so that the player can kick it,

If shooting is plausible, then it makes a shot. Otherwise it computes a simulation to examine that if it kicks this or that direction, then who is the first to touch the ball. If it can find such a kick direction that one of its mates can first touch the ball at the nearest position to the opponent's goalmouth, then it kicks the ball to that direction. In that case, it shouts to the mate to run to the

promising position. If the simulation does not yield any plausible pass course, the player tries to dribble the ball to the opponent's goalmouth.

4 Future Work

Long-range planning:

In the current version, the evaluation function for the player positioning simply relies on the distance from other teammates, and the evaluation function for pass course selection simply relies on the distance from the opponent's goalmouth. Both evaluation functions are too instantaneous or shortsighted, and have no long-range planning or strategy in mind. So I have to explore more intelligent evaluation functions that enables the agent to have time-consistent, long-range plan in its play.

References

1. <http://www.er.ams.eng.osaka-u.ac.jp/robocup/j-info/events/99/jopen99/index.html>
2. <http://rook.fuis.fukui-u.ac.jp/~nishino/robocup99atcamp/>
3. <http://ci.etl.go.jp/~noda/soccer/server/index.html>

PSI Team

Alexander N. Kozhushkin

Program Systems Institute of Russian Academy of Sciences

1 Introduction

Team PSI was developed at Program Systems Institute. This paper is a description of the planning method and common principles of the architecture those we use to construct our software agents. Basic skills and roles of every agent are presented by means of the set of elementary plans. The purpose of the planning process is to compose the extended plan defining the behaviour of the agent from elementary ones. The planning system (or just a planner), built in the agent, modifies extended plans depending on external conditions and the internal state of the agent. It adds new elementary plans to the extended one refining it and controls the execution of elementary plans in a body of the extended plan.

2 Model of the planning system

Consider a model of the planning system built in each player. It is defined by the discrete set of time moments $T = \{0, 1, 2, 3, \dots\}$, a set O of elementary actions which soccer server can execute, a set I of input (or external) states defined by all possible values of data available from the soccer server and the space Ω of internal states of the player. $I \times T$ and O represent input information (*input*) and actions (*output*) of the player, respectively. Play history, or just a play, from the player's point of view may be defined as a map $H : T \rightarrow I \times \Omega$, where $H(t)$ consists of the information from soccer server and of the internal state of the player at t moment. Let $B(i, \omega)$, $C(i, \omega)$ be some logical conditions on the set of full states $I \times \Omega$, and let ϕ and ψ be some maps, $\phi : I \times \Omega \rightarrow \Omega$, $\psi : I \times \Omega \rightarrow O$. Define *elementary plan* of the player as a tuple $p = < B, C, \phi, \psi >$, where B and C are, respectively, its beginning and continuation conditions.

We use some fixed finite set of *basic* elementary plans π . Define the family of subsets $D_i \subset \pi$, $i \in \{1, 2, \dots, n_{lev}\}$, where $n_{lev} > 1$ the number of the hierachic levels, such that $\cup_i D_i = \pi$ and $D_i \cap D_j \neq D_i$ for $i < j$. For every D_i define partial order relation *Prior_i* (*priority*). By means of the family $\{D_i\}$ introduce the interruptability relation *Int* defined on $\pi \times \pi$, with the constraint on it: $Int(p, p') \wedge p \in D_i \Rightarrow p' \in D_{i+1}$. In the rest part of the paper we suppose that if the set π is defined then $\{D_i\}$, $\{Prior_i\}$ and the *Int* relation are defined as well.

Let us explain the meaning of introduced notions. Each elementary plan from π is designed to determine a skill of the player or to solve a particular task. Both B and C are applicability conditions of the plan. But in practice these conditions are different. The aim of the distinction between them is to make behaviour of

agents more stable. The family $\{D_i\}$ i.e. plan levels, determines the hierarchy on π . Levels divide all plans into main and auxiliary ones. The *Int* relation presents the further refinement. It defines which plan *can* be suspended temporarily in favour of another one, auxiliary to the suspended plan. The priority relations play the role of evaluation functions [1] or priority values [2].

Suppose that the set π is defined, we build its extension π^* by the rules:

- (i) $\pi \subset \pi^*$;
- (ii) if $p = < B_p, C_p, \phi_p, \psi_p > \in \pi^*$ then plan $< C_p, C_p, \phi_p, \psi_p > \in \pi^*$;
- (iii) if $p = < B_p, C_p, \phi_p, \psi_p > \in \pi^*$ and $p' = < B_{p'}, C_{p'}, \phi_{p'}, \psi_{p'} > \in \pi^*$ then
 $< B_p \wedge C_{p'}, C_p \wedge C_{p'}, \phi_p(i, \phi_{p'}(i, \omega)), \psi_p > \in \pi^*$, where $i \in I, \omega \in \Omega$.

Thus, π^* is the set of all elementary plans which the agent can use, "tuning" the basic ones to the current conditions.

We define *extended* plan P as a word of a plan language L in the alphabet $A = (\pi^* \cup \{*\}) \times (\pi \cup \{*\}) \times \{0, \dots, n_{lev}\}$. We use the symbol "*" as a abbreviation of the $(*, *, 0)$. L is defined by the next rules (where Q, Q_1 are letters sequences without "*", possibly empty):

1. $* \in L$;
2. if $Q* \in L$ and $p \in D_1$, then $Q*(p, p, 1) \in L$;
3. if $Q * l Q_1 \in L$, where $l \in \pi^* \times \pi \times \{1, \dots, n_{lev}\}$ then $Ql * Q_1 \in L$;
4. if $Q * l Q_1 \in L$, where $l = (< B, C, \phi, \psi, p, k >, p, k) \in \pi^* \times \pi \times \{1, \dots, n_{lev}\}$, and there exists $< B', C', \phi', \psi' > \in \{p' | Int(p, p')\}$, then
 $Ql * (< B' \wedge C, C' \wedge C, \phi'(i, \phi(i, \omega)), \psi', p', k+1) < C, C, \phi, \psi, p, k) Q_1 \in L$.

Every extended plan represents some play history. The letters lying before "*" represent elementary plans, which the agent has used in the past. The first elementary plan after "*" is a current one, i.e. behaviour of the agent is determined by the ϕ and ψ maps of this plan at present. All other elementary plans are those which the agent is going to execute in the future.

For every time t of the play H there are two elementary plan subsets $App_t^H = \{p \in \pi | \models B_p(H(t))\}$ and $Con_t^H = \{p \in \pi | \models C_p(H(t))\}$ of π whose beginning and continuation conditions are satisfied at t moment, respectively. Introduce the notion of the *planning function* as a map $F : L \times 2^\pi \times 2^\pi \rightarrow L$. For every moment t we can build the extended plan defining the agent's behaviour by means of the rules: $P(0) = *, P(t'+1) = F(P(t'), App_{t'}^H, Con_{t'}^H)$. We use the special kind of the planning function defined as follows.

Let functions $f : A \rightarrow \pi \cup \{*\}$ and $g : A \rightarrow \{0, \dots, n_{lev}\}$ are projections, $P \in L$, $S_B \subseteq \pi$, $S_C \subseteq \pi$ and Q is a letters sequence without "*", possibly empty. Define the planning function F by the following clauses:

1. Let $P = Q*$. If there exists the largest element $p \in S_B \cap D_1$ wrt *Prior*₁ relation, then $F(P, S_B, S_C) = Q*(p, p, 1)$. If such p does not exist, then $F(P, S_B, S_C) = P$.
2. Let $P = Q * l_1 \dots l_n$ where $l_1 = (< B_1, C_1, \phi_1, \psi_1, p_1, k_1 >, p_1, k_1)$, $n \geq 1$ and i_{min} is minimal i ($1 \leq i \leq n$) such that for all k ($i \leq k \leq n$), $f(l_k) \in S_C$ and $f(l_k)$ is the largest element of $(S_B \cup \{f(l_k)\}) \cap D_{g(l_k)}$ wrt *Prior* _{$g(l_k)$} relation.

If $i_{min} > 1$, then $F(P, S_B, S_C) = Ql_1 \dots * l_{i_{min}} \dots l_n$.

If such i_{min} does not exist, then $F(P, S_B, S_C) = Ql_1 \dots l_n *$.

If $i_{min} = 1$ and there exists the largest element $p =$

$= < B', C', \phi', \psi' > \in \{ p | Int(p_1, p) \} \cap S_B$ wrt $Prior_{k_1+1}$ relation, then $F(P, S_B, S_C) = Ql_1 * (< B' \wedge C_1, C' \wedge C_1, \phi'(i, \phi_1(i, \omega)), \psi', p, k_1 + 1>) < C_1, C_1, \phi_1, \psi_1, p_1, k_1 \dots l_n;$ if such p does not exist, then $F(P, S_B, S_C) = P.$

We not present any examples illustrating our model here. For this, see [7].

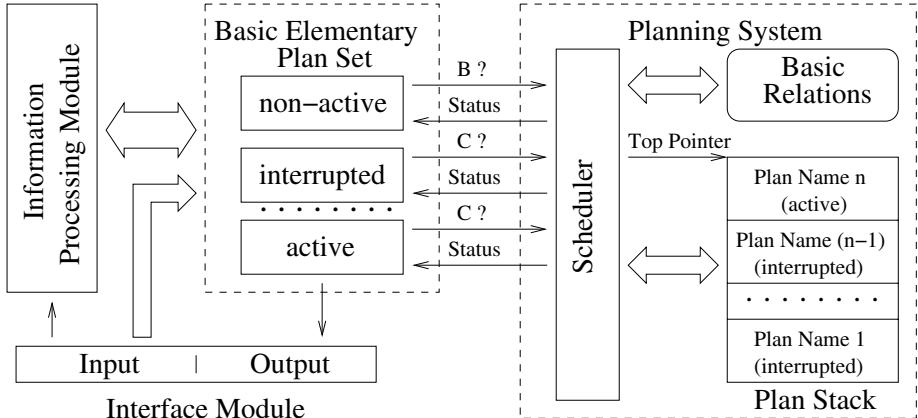


Figure 1: System architecture

3 Agent architecture

Consider the agent architecture. It is schematically illustrated in Figure 1. There are four basic modules. (1) *Interface module* is responsible for the communication with the soccer server. (2) *Information processing module* obtains information that the agent can not receive directly (e.g. absolute coordinates of the player). (3) *Set of basic elementary plans* is, exactly, the set π in our model. (4) *Planning system* controls the execution of elementary plans and builds extended plans in accordance with our definition of the planning function.

Let us describe some of the modules more detailed. The Information Processing module performs routines related to the external world modeling. Its role is analogous to the one of the World Model component presented in [5]. Results of the work of this module are utilized by the elementary plans.

Every plan from the Basic Elementary Plan Set is an independent module. It reads the information from the Interface and the Information Processing module. At every simulation step such module uses these data to test beginning (*B*) or continuation (*C*) conditions. It sends results of the testing to the Planning System module. Planning System can change the current status of the elementary plan by sending respective message to the basic elementary plan module.

At every moment, each elementary plan has one of three statuses: *active*, *interrupted* or *passive*. The active plan can change the internal state and send commands to the soccer server by means of the interface module. The existence of more than one active plan is impossible in the current implementation. Interrupted plans can change the internal state of the agent. Each passive plan tests its beginning conditions only. All of basic elementary plans work in parallel.

The role of the Planning System module is analogous to that of Plan Sequencer in AuRA architecture [6], but basic elementary plans are more powerful entities then motor schemas in [6]. For its work Planning System uses results of the conditions testing which are supplied by elementary plans and static data presenting the family of Basic Relations. There is some similarity between our planning system work and decision-tree searching which is described in [2]. The stack-like structure, Plan Stack, is used to store current elementary plan statuses and internal representation of extended plans. It is illustrated in Figure 2.

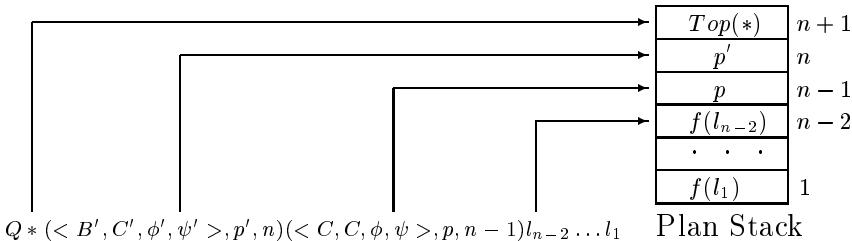


Figure 2: The extended plans internal representation.

4 Conclusion

Our approach is somewhat analogous to that presented in [3, 4], with one essential difference: our planning system works *on-line*, and plans refinements are being made dynamically in case of need. More detailed comparisons deserve the further investigations and are to be presented elsewhere. The further development of the method itself and the more precise formulation of its essence is a goal of further work.

References

1. T. Dean, S. Kambhampati. Planning and Scheduling. *CRC Handbook of Computer Science and Engineering* (1996).
2. S. Coradeschi, L. Karlsson. A Role-Based Decision-Mechanism for Teams of Reactive and Coordinating Agents. *RoboCup-97: The First World Cup Soccer Games and Conferences*. Springer Verlag, LNAI (1998).
3. M. Ginsberg. Modality and Interrupts. *Journal of Automated Reasoning* (14)1 (1995) pp. 43–91.
4. M. Ginsberg, H. Holbrook. What defaults can do that hierarchies can't. *Fundamental Informaticae* (21) (1994) pp. 149–159.
5. H. Burkhard, M. Hannebauer, J. Wendler. AT Humboldt – Development, Practice and Theory. *in Kitano, H. (ed.) RoboCup97; Robot Soccer World Cup I. LNAI*, Springer (1997) pp. 357–372.
6. R. Arcin, T. Balch. AuRA: Principles and Practice in Review. *Journal of Experimental and Theoretical Artificial Intelligence* vol 9, No 2, (1997).
7. Kozhushkin A.N., PSI Team. *in M. Veloso, E. Pagello, H. Kitano (Eds.) RoboCup99; Robot Soccer World Cup III. LNCS*, Springer, Vol. 1856 (2000) pp. 623–627.

Gnez: Adapting knowledge to the environment with GA

Takenori KUBO

Ogura lab. Dept. of information science, Fukui univ. JAPAN,
kubo@rook.fuis.fukui-u.ac.jp

Abstract. This paper describes the Gnez RoboCup simulation team. we have investigated an evolutional acquisition method of the situation recognizer for a RoboCup soccer agent. We apply Hierarchical Fuzzy Intelligent Controller to decide soccer agents' behavior, and implementing the behavior rules, which consist of condition and behavior. We try to acquire ball-kickable condition recognizer with Genetic Algorithm. On a single agent game, we get gene that can recognize ball-kickable condition actually. And the recognizer is effective enough to use as ball-kickable condition.

1 Introduction

The goal of implementation of Gnez is realization the agents which can fit themselves to given environments using Genetic Algorithms. In previous studies, a GA method was used for learning teamwork[3]. We have introduced a framework for automated parameter adjustment to the environment using Genetic Algorithms(GA)[2].

We use a hierarchical controller as a basic structure of an agent [1]. Strategic actions were implemented as a combination of basic actions (kick, dash, turn, etc.). On making the agent system, human programmers need to adjust their programs to realize desired action in trials and errors.

We examine automatic rule adjustment using evolutional method.

2 Team Configuration

Our team is based on CMUnited-99 code and refer to World Model of YowAI.

Coach: We did not use any coach client.

Communication: Our agents did not use any communication.

Skills: We use static rules and tune them, using GA several parameters of rules.

3 Acquisition of situation recognizer using GA

The action rules are expressed in the following format.

$$R_i : \text{if } Condition_i \text{ is true then do } Action_i. \quad (1)$$

A control-knowledge is constituted as a set of such rules. It is important that the *Condition_i* recognize situations precisely for superior action. However, it is difficult to express for the situations definitely.

We show how to adjust the *condition* which is included in the rules, where the control-knowledge rules are given a priori.

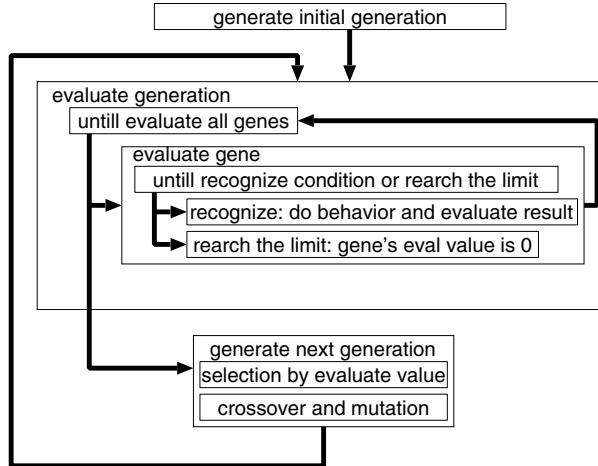


Fig. 1. GA process with evolution under the behavior rules

A gene is evaluated by the count of success and failure under an action activated by a rule that have *condition* associated with the gene. Agent analyzes situation with genes on each decision making. For the next generation some used gene were selected according to the degree evaluated with action results. When all genes are evaluated, a generation change happens.

By a generation change, genes are selected by a evaluation result. And next population is generated with crossover and mutate operation from selected genes.

In this way, Every *condition* is optimized by asynchronous and parallel GA.

4 Experimental Result

In this section, we acquire a *kickable* condition recognizer as a simple example in experiments. *kickable condition* stands for that a player can "kick a ball in that situation".

kickable condition is expressed with threshold value t for distance from the player to the ball. In other words, *kickable* is one real number value and judges whether the player can "kick a ball" by comparing distance to a ball with t .

We use real number value as gene expression of *kickable* in GA. Genetic operations are shown as follows.

crossover operation creates a new gene. Which has a mean value of parents t as its threshold value.

mutate operation creates a new gene. Which has a 0.5-2 times value of parents t as its threshold value.

Only one agent was connected to soccer server in an experiment. Gene pool size is 10 and mutation is 0.5.

A result of threshold value t is shown in the Fig.2. A kick success ratio r is shown in the Fig.3.

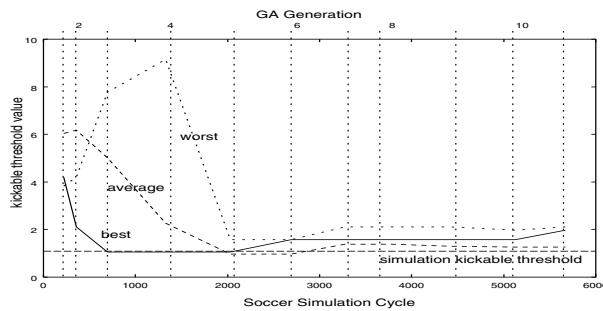


Fig. 2. kickable threshold value.

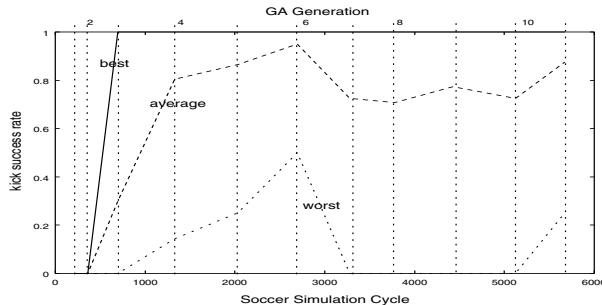


Fig. 3. kick success ratio.

An elite gene have obtained good threshold value at earlier stage. This result show that a *kickable condition* can be obtained using proposed evolutional acquisition method.

5 Summary

We apply an Genetic Algorithms to adapt a soccer agent to an environment on RoboCup soccer simulation.

We intend to apply this proposed method to acquisition of much higher concepts such as strategy and/or tactics for soccer playing. Furthermore we need to examine more complicated *conditions*.

6 Future work

We will improve an evaluation in GA using a coach client. and require an opponent model using coach client.

Team Leader: Takenori KUBO

Web page: <http://bishop.fuis.fukui-u.ac.jp/~kubo/gnez>

References

1. Tomomi Kawarabayashi, J. Nishino, T. Morishita, T. Kubo, H. Shimura, M. Mashimo, K. Hiroshima, and H. Ogura. Zeng99 : Robocup simulation team using hierarchical fuzzy intelligent control. Technical report, RoboCup, 1999.
2. Takenori KUBO, Junji NISHINO, Tomohiro ODAKA, and Hisakazu OGURA. Evolutional acquistion of condition recognizer in the robocup soccor agent. In *Proceedings of Fuzzy, Artificial Intelligence, Neural Networks and Soft Computing (FAN'99)*, pages 155–158, 1999(in Japanese).
3. S.Luke, C.Hohn, J.Farris, G.Jackson, and J.Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup I*, pages 398–411, 1998.

Zeng00 : The Realization of Mixed Strategy in simulation soccer game

Takuya Morishita¹, T. Kawarabayashi¹, J. Nishino², H. Shimura¹ and
H. Ogura²

¹ Graduate School of Engineering, Fukui University, 3-9-1 Bunkyo, Fukui City, Japan

² Faculty of Engineering, Fukui University, 3-9-1 Bunkyo, Fukui City, Japan

{morisita, tomomi, nishino, shimura}@rook.fuis.fukui-u.ac.jp and
ogura@nqueen.fuis.fukui-u.ac.jp

Abstract. We propose and examine a time mixed strategy method. We can obtain the optimal mixed strategy in a simulation soccer by means of this method, even if we don't have the payoff matrix which consists of both teams' strategy. We prepared two symmetric strategys, "side attack strategy" and "center attack strategy" and examine. The result is that proposed approach could obtain optimal mixed strategy in soccer game.

1 Introduction

The aim of this study is to propose an analysis method for an efficiency of strategys in simulation soccer games in terms of the game theory[1].

We propose and examine a time mixed strategy method to get the optimal mixed strategy in a simulation soccer, trying several mixed strategys, by means of getting several payoffs and getting the saddle point, we could obtain a quasi-optimal mixed strategy even if we don't have the payoff matrix which consists of both teams' strategy.

The proposed method can be applied to the class of games such as simulation soccer game.

In the following section, we prepare two pure strategys: "center attack strategy" and "side attack strategy" for experiments.

2 Three layered control system

Our agent design is based on three layered control model that was proposed by Jens Rasmussen[2]. Fig.1 shows a configuration of the control system. Strategy layer makes a decision the way to attack and defend and from sensor infumation and strategy knowledge. Tactics layer makes an action as its role and the strategy from sensor information and tactics knowledge. Behavior layer transforms the action decided by tactics layer into a combination of basic commands (dash, kick etc.) and then it executes those commands.

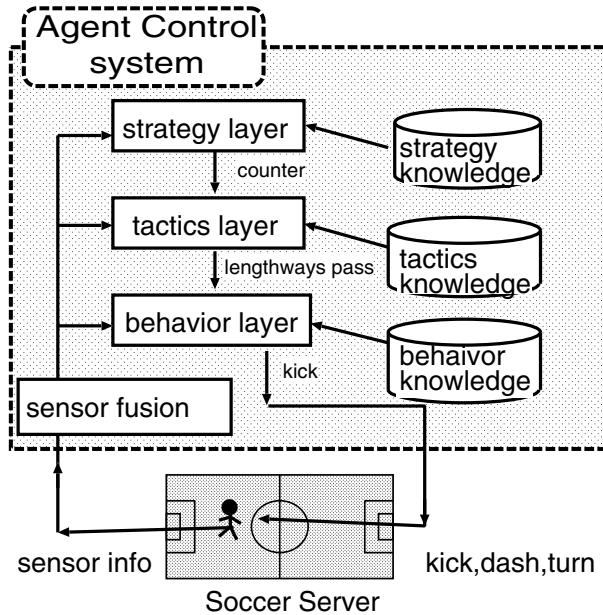


Fig. 1. System configure for The Zeng00 soccer agent

In the control system, tactics layer can choose an action that is different from that of another agent and is based on its role, even if all agents a the same strategy layer and strategy knowledge.

3 Representation of mixed strategy

In case of simulation soccer game, a game has certain interval in which strategys can be executed. We propose the representation of mixed strategy, by alternating two strategys. Fig.2 shows a mixed strategy that executes two strategys repeatedly in order to mix strategy A and B in the ratio of 3 to 7.

In this case, each agent could have the team strategy as same as team-mates have simultaneously.

The proposed representation of mixed strategy advantages: Freely change the mixing ratio. It is easy for each agent to switch simultaneously.

4 Experiments and Result

We prepared two symmetric strategys, “side attack strategy” and “center attack strategy”. We mixed the strategy by time rate and examined some matches, as changing the rate. And, we obtained the relation between the value of game and the rate of strategy switching. We evaluated the value of game with a numerical

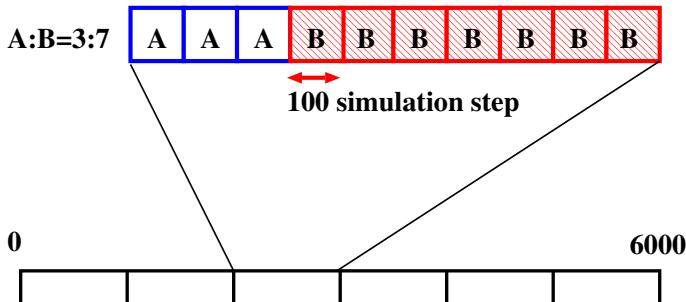


Fig. 2. A method to mix strategy A and strategy B using time.

total of time intervals at which Zeng00 could play the ball beyond to the 40m line.

Where the unit time was set time interval to 100 simulation steps because a strategy must be done in consecutive. We used the team CMUnited99 as a match partner. We set mixed rate, 10:0, 8:2, 7:3, 6:4, 5:5, 4:6, 3:7, 2:8, 0:10, and then examined under every rate. The experimental matches repeated 10 times every mixed rate. Total number of matches is 90.

The Fig.3 indicates the chance of shoot on different mixed rate. The y axis indicates the total time of chance to shoot, and the x axis is mixed rate.

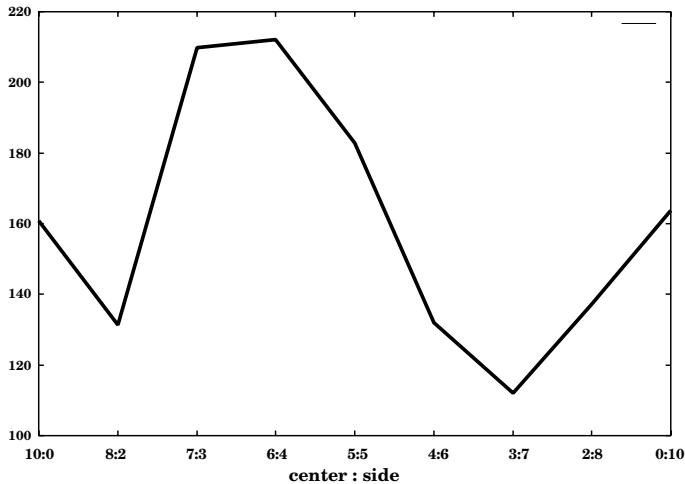


Fig.3. Relation of ball position and mixed rate

The mixed strategys are better than the pure strategys. Mixed rate is the most effective in case of 7:3 or 6:4. Mixed rates 8:2 and 3:7 are more effective than pure strategy.

5 Summary and future works

Proposed approach could obtain optimal mixed strategy in soccer game. This method represents mixed strategy rather than pure strategy and that can be applied to some problems like as the simulation soccer game in which mixed strategy is realized by means of a time alternation.

It is not clear that a result is optimal mixed strategy, because trial number of times is not enough. Therefore, more experiments need to be done.

In the future works, we plan to study on how to express mixed strategy in case of that agents don't synchronize each other, and how to classify the opponents strategy that is not obvious to us.

References

1. Ian Frank. Search and Planning Under Incomplete Information. *Springer*, 1998.
2. Jens Rasmussen. Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Trans. SMC*, 13(3):257–266, 1983.

RoboLog Koblenz 2000

Jan Murray, Oliver Obst, Frieder Stolzenburg

Universität Koblenz-Landau, AI research group
Rheinau 1, D-56075 Koblenz, GERMANY
{murray, fruit, stolzen}@uni-koblenz.de

1 Introduction

RoboCup-2000 was the second world championship, RoboLog Koblenz participated in. To the best of our knowledge, RoboLog Koblenz is the only team in the simulation league, whose agents are programmed in Prolog. The main goal of our research is to make declarative agent programming possible, by writing the agents' control programs in *SWI-Prolog* [6]. Procedural aspects of the agents' behavior can be specified by statecharts, that have become part of the *Unified Modeling Language* (UML) [4]. See also the paper *Towards a logical approach for soccer agents engineering* in this volume.

While the high-level control programs for our soccer simulation agents are written in Prolog, basic skills and predicates are part of the RoboLog kernel, which is written in C++. In the kernel, the *CMUnited-99 library* [5] is integrated, which has improved the performance of RoboLog Koblenz significantly. We were able to enter the double-elimination round in the competition in Melbourne 2000, after winning the last game in our round robin group with a score of 39–0 (the highest result ever scored in one RoboCup game). Finally, we end up on the 13th rank. In addition, we became 5th in the RoboCup European Championship in Amsterdam, June 2000. We finished as the 2nd best German team.

2 Special Team Features

As already mentioned, the main difference to other teams participating in the simulation league is the use of the logic programming language Prolog for RoboLog Koblenz agents. At the beginning of each simulation step, the Prolog system is called. The next action of the agent is determined, taking its knowledge of the world into account. For this, CMUnited-99 library functions among others are exploited. In Section 3, we will explain, how these functions can be accessed in Prolog.

The transition from the data and actions of the *Soccer Server* [2] to the specification of agents by means of logic programming (Prolog) requires the translation of quantitative data into a qualitative, propositional representation. We will discuss this also in Section 3. Another feature of RoboLog Koblenz is its use of communication, because besides basic information also messages in the *Knowledge Query and Manipulation Language* (KQML) format [3] can be exchanged (see Section 5). Following the template for team descriptions used throughout this volume, we will also discuss other topics.

3 World Model

We use the CMUnited-99 low-level library [5] including the world model contained therein. We added some communication primitives to the library, so our players can enhance their knowledge of the world by means of communication (see Section 5). The connection of these skills to Prolog happens in two steps: We use C wrapper functions to build a shared RoboLog library. This library can be used to implement soccer clients in C, or, as a second step, to connect the RoboLog kernel to other programming languages. In our case, some C(++) code is responsible for converting the RoboLog library functions into logical predicates and connect them to SWI-Prolog.

In order to identify and abstract over situations, additional predicates for qualitative reasoning are employed [1]. That means that besides quantitative distances, the higher levels make use of predicates which succeed, if the ball is *reachable* or distances are within certain distance intervals. Unlike in our last year's team, we did not name these quantitative distances. For directions to other objects, there is a predicate that maps the (quantitative) angles to sectors so that it is easily possible to get the qualitative direction of an object to the player.

4 Coach

Although we have implemented a very limited online coach, which can also be programmed in Prolog, we did not use it yet in a competition. We plan, however, to shift some of the computationally expensive tasks, that are now performed by each agent, to the coach.

5 Communication

RoboLog Koblenz agents can use communication for two purposes: (a) sharing their knowledge of the world with team-mates and (b) coordinating actions with other players. Thus, at first, RoboLog Koblenz agents make use of communication in order to enhance their knowledge of the world. Every two cycles one of the agents tells parts of his world model to his team-mates. Each agent sends such a message every 22 simulation cycles. Figure 1 (a) shows an example of a message where player 11 announces the ball position and positions of some players (X- and Y-coordinates, confidence level and distance wrt. the players), if the confidence level is high enough. The string ends with a checksum, in order to enhance security.

```
(a) (hear 285 (player RoboLog2K 11)
      _b -14.6 17.4 0.9 23.2
      _p r 1 -47.0 2.1 1.0 58.3
      _p l 8 1.7 -1.0 1.0 22.6
      C5E3
    )
(b) (<performative>
      :sender
      :receiver
      :language
      :reply-with
      :in-reply-to )
```

Fig. 1. Communication between RoboLog Koblenz agents.

The RoboLog kernel (and also the Prolog interface) provides the possibility of communicating messages in KQML format [3]. This feature mainly is intended for future use, such that, in principle, agents with completely different internal structure will be able to communicate with each other in the common KQML format. In our context, KQML messages contain information about the sender and receiver, the contents language (e.g. Prolog) and message identifiers. We considered the performatives *ask*, *tell* and *sorry*. A template is given in Figure 1 (b). We conducted some experiments, indicating that e.g. the passing success rate could be improved by employing communication.

6 Skills

Since the RoboLog Koblenz team is based on the low-level code of CMUnited-99, the skills of the agents are a subset of the skills the CMUnited-99 players possessed. We adapted skills like *ball interception* and *hard kicks*.

The positioning of our agents is very simple. They have only two basic schemes to work out where to position themselves. First there is a number of fixed positions for each agent, that he has to move or run to in special situations. The second scheme is not as rigid. The agent tries to reach a position which satisfies certain conditions. It tries, for example, to position itself in such a way that there is no opponent closer than 5 m and no one is between the agent and the ball and the distance to the ball is between 7 m and 20 m. Clearly such conditions hold for many positions.

The goalie's behaviors are simple but effective, too. In the *play on* mode, if the ball is dribbled by an opponent, the goalie tries to stay on the bisector of the angle formed by the goal posts and the ball. If, on the other hand, the ball is in a shot, i.e. no player is close enough to be considered dribbling, the goalie chooses one from several behaviors to block or intercept the ball.

7 Strategy

Most of the rules related to the players strategy deal with the local situation of the player: Is the ball kickable, or are there other team-mates closer to the ball? Therefore, the team-work between the players is implicit. But this works, because the knowledge of the behavior of the team-mates is coded into the rules.

However, there are also rules which model the partners situation. For instance if a player wants to pass the ball to a partner, there is a rule which succeeds if this partner can also find a partner for passing the ball to. This rule for explicit team-work is especially useful for the goalie kick-out. For the local situations of the players we used statecharts as described in the paper *Towards a logical approach for soccer agents engineering* in this volume to design the respective behavior. A (visual) tool for aiding the design process might be helpful here.

No special opponent modeling is employed yet. Our agents use only the visual information available and communication to estimate positions of the opponents. The rules for the strategy during a game are static for each opponent and all through a game. This was the major disadvantage during the tournament in Melbourne, since our team could not adapt to our opponents sufficiently.

8 Team Development

The new version of RoboLog Koblenz was implemented by a team of 3 to 6 persons. Besides the authors of this paper, some master students were involved in the development of the team, namely Björn Bremer and Marco Dettori (see list of team members in Figure 2). We used SWI-Prolog [6] as implementation language and made good experiences with it. Even for unexperienced programmers, it was easy to implement soccer players. We used the debugging facilities built into the CMUnited-99 library and an offline coach, but no additional special tools.

Team Leader: Dr. Frieder Stolzenburg

Team Members:

Björn Bremer; Marco Dettori

- Universität Koblenz-Landau
- Germany
- masters student
- did not attend the competition

Jan Murray

- Universität Koblenz-Landau
- Germany
- masters student
- attended the competition

Oliver Obst; Dr. Frieder Stolzenburg

- Universität Koblenz-Landau
- Germany
- researcher
- attended the competition

Web page <http://www.robolog.org/>

The source code and binaries of our team are available under the GNU General Public License.

Fig. 2. The RoboLog Koblenz team.

9 Conclusion

Deductive specification of multi-agent systems for the RoboCup by means of logic programming (and statecharts) is possible, as stated in this paper. We will continue our research in this direction and enter a team for the competition in Seattle 2001.

References

1. E. Clementini, P. Di Felice, and D. Hernández. Qualitative representation of positional information. *Artificial Intelligence*, 95(2):317–356, 1997.
2. E. Corten, K. Dorer, F. Heintz, K. Kostiadis, J. Kummeneje, H. Myritz, I. Noda, J. Riekki, P. Riley, P. Stone, and T. Yeap. *Soccerserver Manual*, 5th edition, May 1999. For Soccerserver Version 5.00 and later.
3. T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, R. Pelavin, S. Shapiro, and C. Beck. Specification of the KQML agent-communication language. Technical report, The DARPA Knowledge Sharing Initiative, External Interfaces Working Group, June 1993.
4. Object Management Group, Inc. *OMG Unified Modeling Language Specification*, 1999. Version 1.3, June 1999.
5. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer WorldCup III*, LNAI 1856, pages 35–48. Springer, Berlin, Heidelberg, New York, 2000.
6. J. Wielemaker. *SWI-Prolog 3.3 Reference Manual*. University of Amsterdam, The Netherlands, 2000.

Open Zeng: An Open style distributed semi-cooperative Team development project from Japan

Junji Nishino, Takuya Morishita, Takenori Kubo *

Open Zeng project, JAPAN
<http://rook.fuis.fukui-u.ac.jp/~nishino/oz/>
nishino@rook.fuis.fukui-u.ac.jp

Abstract. The team Open Zeng is a quite interesting and unique project to study subjects on heterogeneous cooperative teamwork behavior. Twelve players from seven different university and institutions were gathered and played soccer well against ordinary teams. We introduced the Evolutionary development model. The project provides a distributed heterogeneous team development using WWW and eMail.

1 Introduction

The team Open Zeng ,called OZ, is a quite interesting and unique project which aims to study subjects on heterogeneous cooperative teamwork behavior. The team is made up of several team members from different university and institutions in Japan. The members also have their own teams that had attended to RoboCup competitions independent of the team OZ.

The project have several goals, clarifying;

1. how to construct cooperative team work among heterogeneous players.
2. how to steer destibuted/ decentralized development project under insufficient communications.
3. how to describe such different clients' (players') configuration specifications

For these issue we introduced and examined the OZ team development model with behavior based communication. The team works well against uniformed teams that competed RoboCup in 99.

In the RoboCup 2000 competition, OZ held 2nd place of round robbin league and 8-12 of elimination tournament. That proofed very good performance of a heterogeneous team.

2 Clients, players and project members

The team OZ employs eleven different clients, so that the controller configurations for all clients can't be described here. Please refer the original team

* Dept. of Human and Artificial Intelligent Systems, Fukui University JAPAN

Table 1. a Formation of Open Zeng; abbreviations are FW:=Forward player, MF:=Mid field player, DF:=Defense player, GK:= Goalie, Fukui: Fukui University, JAIST: JAPAN Advanced Institute of Science And Technology, Keio: Keio University, Kinki: Kinki University, Mie: Mie University, NITECH: Nagoya Institute of Technology, TITECH: Tokyo Institute of Technology, UEC: The University of Electro-Communications

Position	Player name	developer name	affiliation	Orig. team	code
FW	mitsuhide	Suzuki T.	UEC	YowAI 99	C
	CZ125	Koto T.	UEC	TakAI	C++
	yukichi	Kinoshita S. Tanaka.N.	Keio	11Monkeys 99 11Monkeys 00	C++ C++
MF	Tipl wingless birdie	Shinoda T. Morisita T.	JAIST Fukui	Kakitsumabata Zeng	C++ C++
DF	Dajarenja	Igarashi H.	Kinki	KU-Sakura	C
	Kumanosuke	Hioki T.	Mie	Sfidante	C
	Bob	Ito H.	NITECH	Kakitsubata	Java
	NS2k(ver.haken)	Nakagawa K.	NITECH	NITStones99	Java
	nurikabe	Esaki T.	NITECH	NITStones00	Java
	Castol	Kubo T.	Fukui	gnez	C
GK	savior	Oota M.	TITECH	gemini	C

description in the RoboCup books [1–3]. Some clients have learning ability and others don't have it. Some use structured decision maker and others use rule based one.

Each player has their own position and/or role in the team OZ. The developer select role for his player client by only his opinion without any negotiations between developers. Even some player have NO ROLE but just chase a ball. It is allowed and not illegal in the team OZ project. We hope other player covers that funny teammate.

The table 1 shows a position assignment of the team OZ and each player's affiliation. This formation is not a fixed one. It could be reconfigured as the performance of team, the opponents and the research interests are changed.

To make first positions we negotiated each other by e-Mail communication and information on WWW.

3 Player's name

All players, aka client programs, have their own “player nickname” for personification. It isn't neither a developer name nor an original team name which the client program is involved. The name of each player is very important to determine each player and their behavior. And it is also important on time of discussion so that we can state and understand characteristics of every player. One another point is that each developer could love his own player with its name.

4 Resources and heterogeneous players

In the team OZ project, we study heterogeneous cooperation behavior.

In ordinary researches on cooperative behaviors, homogeneous clients developed by quite a few programmers i.e. by only one or two persons, are used. Especially in a robocup simulation environment, client programs seems to be easy to copy and merge so that they uses same clients ability for different role.

But there are large difference between “dribble of Striker” and “dribble of Defender,” even in simulation environment. This difference is caused by resources rearrangement problem; stamina plan, sensor quality control, CPU time, and coordination of kick-dash-turn commands. We tend to think that the programs could be made in same code but they have to be different in REALISTIC constraints served by the RoboCup standards.

Actual cooperative team development of homogeneous players is quite an interesting issue.

5 Protocols: communications

There is no explicit cooperation protocols such as a message on say command for the team OZ. The reason is that we don't have any development center. The team organizer don't provide any direction of the team but just an environment to be together the players and developers comments.

Although there is no communication in the team OZ, they can play soccer well with a teamwork behavior in experiment matches. They can pass the ball and keep ones zone. This indicates they use implicit cooperation protocol aka a body language. Positioning and running to free space is a signal to cooperative soccer play.

In other hand, actually, investigating a minimal protocol set will be an aim of this project.

6 OZ Team development model: an evolutionary loop

To steer this unique distributed project, we employ WWW , FTP and MailList systems that are shown in the Figure 1. First of all, to call for members RoboCup-J Mail List is used. There gathered twelve members from seven institutes and university. The team member includes from 11Monkeys and YowAI, which are the best two teams of Japan.

This project is on joint institute/university style with NO center development manager. This project is run under completely distributed free development model. A programmer has one player. They contribute each player program on free ftp. All member can obtain all player program including his own player, and freely do test match with them. Team manager also examine them by testing matches and put the log on WWW site. Team members see how his player do on the game and think how change his player next. We call this development model ”Evolutionally team development”. The player code is a gene. It is evaluated by

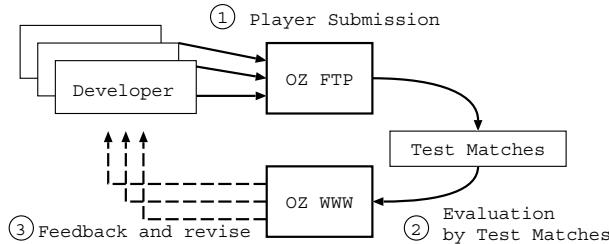


Fig. 1. Open Zeng depelopment support Systems

matches with other players. The open programs can be mixed with new player, this process can be called gene crossing. A new idea for a player code is mutation. These are same as biological evolution process. Free programs for player and this style team development model makes the evolutionary development possible.

7 Conclusion: What should we do for **REAL** cooperation in simulation

The first aim of this project OZ is to show technology and methodology for heterogeneous cooperation studies.

To realize this issue we create a unique team with different players developed independently by each developer. We employ WWW and eMail system for the no center development model. It enable us to realize distributed development and constrain us to develop a player for the team without central planner. It makes the players far different each other.

The heterogeneous, ill-known and less communication team could play soccer well against ordinary teams. The strengthen of OZ had shown some aspects of heterogeneous cooperation. We have to analyze that capabilities.

In the past time, mess of heterogeneous players, i.e. difference of players should be avoided as bad things to analyze a team cooperation behavior. However, in the real world, even uniformed created agents has many differences each other. Thus we should treat such differences to realize **REAL** cooperation. This project will not be closed in JAPAN. The world wide “players” are welcome.

References

1. H. Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*. Springer, 1998. ISBN 3-540-64473-3.
2. M. Asada and H. Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. Springer, 1999. ISBN 3-540-66320-7.
3. M. Veloso, E. Pagello, and H. Kitano, editors. *Robocup-99: Robot Soccer World Cup III*. Springer, 2000. ISBN 3-540-41043-0.

Gemini in RoboCup-2000

Masayuki Ohta

Tokyo Institute of Technology

Abstract. We implemented "Gemini" a client program for the SoccerServer. The objective of this program is testing a lot of learning methods on multi-agent environments. In the current implementation, Gemini can select the most effective strategy for an enemy, using reinforcement learning. Furthermore, we are trying to implement a meta-level learning, which turn each learning function on or off according to whether the learning succeed or not.

1 Introduction

Because it is hard to adjust behavior of each agent in multi-agent environments, machine learning techniques are necessary. We implemented "Gemini" a client program for the SoccerServer[3], to test some kinds of reinforcement learnings[1] on it. In particular, we are interested in learning cooperative behaviors under the following restrictions.

- Agent can get no information from the global viewpoint
- Sensory data includes some errors
- Agent can not communicate with each other

In such environment, reinforcement learning has been applied in many cases. However, because reinforcement learning progress to maximize the reward of the agent itself, when using reinforcement learning in this environment, there is a problem that reward of the whole team may not be maximized. Moreover, when we realize online learning, we can not avoid the possibility of unsuccessful learning. Then, some methods to reduce risks of online learning become necessary, because the risk is increased in such noisy environment.

In this paper, we show the outline of Gemini's architecture and the learning system to solve these problems.

2 Architecture

Gemini mainly consists of the following five modules as Figure 1: Network Interface, Sensor, World Model, Controller and Learning Module. Learning Module includes some learning functions, each of which can have different kind of learning method. Gemini execute the followings in every simulation cycle. This loop will be repeated until the end of the simulation.

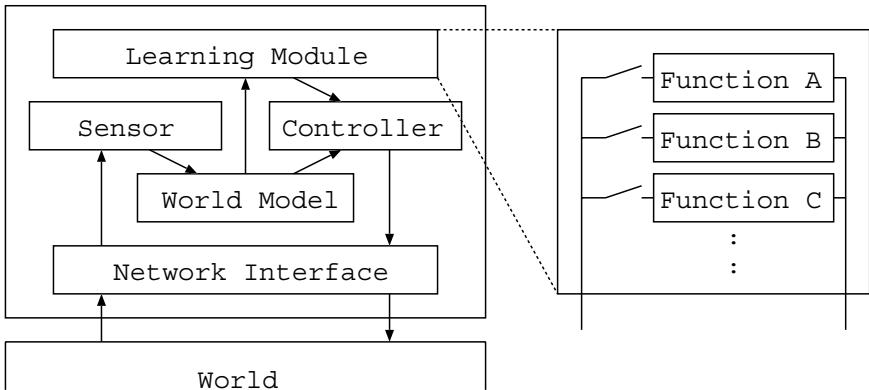


Fig. 1. Structure of Gemini

1. Sensor receives perception information through Network Interface, and put it into World Model.
2. Learning Module executes some learning method, and make some changes to Controller.
3. Controller decides the next action and send control commands to the SoccerServer through Network Interface.
4. World Model estimates the next object's positions and velocities.

The action rules in the Controller are written in “if-then rules”. Considering the World Model, the Controller decides the next action based on this rules. In the “learning and change strategies” phase, some learning functions calculate new variables for the if-then rules in the Controller. Then, Gemini can change its action rule.

At the beginning, all the players (except for the goal keeper) behave just the same. Start with different position, each player learn their best behavior respectively. This makes the difference of actions among each part.

3 Learning

Gemini realizes both online learning and offline learning with the same mechanism. To reduce the risk of online learning, it also have a meta-level learning mechanism, which select only effective learning methods.

3.1 Select the Best Strategy

Gemini get to select the most effective strategy against the opponent with reinforcement learning. There are some learning functions in the learning module,

and each learning function progresses individually. Variables for Controller are calculated by these functions, and they have influence on behaviors such as “base position”, “from which side to attack” and “whether do centering or shoot directly”. A learning function has a neural network, and learns the best strategy by improving it. We are using back propagation to refine the neural network, and give a reward of reinforcement learning as an expected value of the output.

For example, the learning function to decide ”from which side to attack” acquires an expected reward of each policy as following.

- In advance, divide the field into several parts(cell). The division is not changed during the game.
- Neural network calculates the expected reward of each policy using ”position of the ball” and ”the number of the opponent in each cell” for the input.
- Select the policy whose expected reward is the highest, and make some changes into variables for target point. (To do exploration, there is a small possibility to select random policy, too.)
- Every five seconds, if the ball moves forward, expected reward for the policy is increased and refine the network with back propagation with the new expected reward.

In this kind of reinforcement learning, each agent evolves only for their own reward. To raise whole reward of the team without communication, we allow all agents to get reward and reinforce the policy at that time even if the successful is due to other agents[4].

This method can also be used for online learning. But exploration is sometimes very risky, and frequent changes of strategy causes bad influence to the learning of the teammates. Therefore, when we use this method for online learning, Gemini stores each reward while play on, and changes the strategy all at once when the play stops.

3.2 Meta-Level Learning

Gemini operates more than one learning mechanism simultaneously, but all the learning methods do not always go well. To use effective learning function selectively, Gemini has a meta-level learning function that turn on and off each learning.

In the recent implementation, we realize meta-level learning as follows. All the learning functions start with active state. During the simulation, the meta-level learning function records how much reward does each learning function get. If there are some learning function whose reward going down, meta-level learning function inactivate the learning function. Inactive learning functions become active by a random timing, but if the function is inactivated repeatedly the interval goes longer.

4 Summary

In this paper, we describe the outline of Gemini for the RoboCup 2000. It can select the best strategy against opponents with online learning. To maximize the whole reward of the team in no communication environment, we adopt unique way to reward. Furthermore, because the online learning is so risky, the learning module has a mechanism to evaluate the effect of each learning function, and can use only effective learning selectively.

References

1. Kaelbling L. P., Littman M. L. and Moore A. W. "Reinforcement Learning: A Survey" *Journal of Artificial Intelligence Research* 4, pages 237-285 1996.
2. Kimura H., Yamamura M. and Kobayashi S. "Reinforcement Learning in Partially Observable Markov Decision Processes: A Stochastic Gradient Method" *Journal of Japanese Society for Artificial Intelligence*, Vol.11, No.5 pages 761-768 1996
3. Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki and Ian Frank. Soccer Server: A Tool for Research on Multiagent Systems. *Applied Artificial Intelligence*, Vol.12, pages 233-250, 1998.
4. M.Ohta and T.Ando "Cooperative Reward in Reinforcement Learning" Proc. of 3rd JSAT RoboMech Symposia pages 7-11 April 1998.

Team Description for Lucky Lübeck – Evidence-Based World State Estimation

Daniel Polani and Thomas Martinetz

Institut für Neuro- und Bioinformatik, Universität zu Lübeck
D-23569 Lübeck, Germany

polani@inb.mu-luebeck.de, martinetz@inb.mu-luebeck.de
<http://www.inb.mu-luebeck.de/staff/{polani,martinetz}.html>

Abstract. This paper describes the important features of the team LUCKY LÜBECK Melbourne 2000 which is based on the MAINZ ROLLING BRAINS 1999 agent code. In LUCKY LÜBECK, a new method, *Evidence-based World state Estimation*, has been introduced, by which player and ball position can be estimated about an order of magnitude more accurately than before.

1 Introduction

This paper describes the entry of the team LUCKY LÜBECK to RoboCup 2000. LUCKY LÜBECK is based on data structures, functions, skills and player strategy from the RoboCup agent code developed by the first author together with his former team MAINZ ROLLING BRAINS during his time as their team leader [1–3]. Here we give an overview over the most important extensions of the former agent model.

A main aspect of the earlier efforts in [1] had been devoted to the development of a unified concept to represent world knowledge in a transparent way that would be consistent with both sensor readings and agent actions. LUCKY LÜBECK Melbourne 2000 extends this concept in a natural way for near-optimal reconstruction of agent and ball positions via a method which we call EWE (*Evidence-based World state Estimation*). In EWE, new evidence, e.g. sensor readings, is incorporated into the current world model striving to maintain a maximum degree of consistency. Instead of the simple, more-or-less plausible filtering heuristics used before, our new approach is clear-cut, conceptually concise and can be given a well-defined interpretation. The method is general and powerful and can be used in different contexts.

The EWE concept can be seen: 1. as a generalization of nonlinear Kalman filtering [4] towards model-based filtering; 2. as a world state description in terms of a generalized interval arithmetic; 3. as a natural extension of Markovian Self-Localization [5] to spaces other than 2-dimensional positions; 4. as a continuous implementation of BDI [6, 7] for world state estimation, since it strives at consistency between hypotheses about the world state and evidence.

2 Evidence-Based World State Estimation

In EWE, a single-valued state variable in the world model is replaced by a (multi-valued) set or a probability distribution of possible values for that variable. This multi-valued representation of the state variable models the agent's set of hypotheses about the true world state. Given such a multi-valued representation, the arrival of new evidence imposes restrictions on the set of possible and/or probable hypotheses, discarding some of them altogether and modifying the probability of others. For reasons of efficiency, in our agents we concentrated on implementing the EWE concept for the estimation of the most important world state variables, namely the position of the player and the movement of the ball.

Originally, the mechanism developed here was first used to determine ball position and velocity and later extended to also calculate the player position. For presentation purposes we will, however, first describe the calculation of the agent position, followed by the mechanism for ball localization. A first version of EWE was already implemented but not yet activated for Eurocup Amsterdam 2000, since it was not sufficiently stable at that time. The first use of EWE in an official tournament was in the Melbourne 2000 event.

2.1 Determining the Agent Position

The agent position is a two-dimensional vector $p \in \mathbb{R}^2$. As corresponding multi-valued representation in our world model we use a two-dimensional polygon. This representation can be interpreted as a probability density distribution attaining the value $1/A$ in the inside the polygon and 0 outside (A being the area of the polygon). At the same time it forms the basis for a “polygon arithmetic” (a generalization of interval arithmetic to the 2-dimensional plane) which we will need later. Any real-valued quantity s sent to the agent by the server, be it angle or distance, is converted to a quantized value $\hat{s} = Q(s)$ via a *quantization function* $Q : \mathbb{R} \rightarrow \mathbb{R}$. The quantization functions used in the soccer server have the property that the *inverse quantization* $S := Q^{-1}(\hat{s})$ for any given value $\hat{s} \in \mathbb{R}$ is either an interval $[a, b] \subseteq \mathbb{R}$ or empty (if \hat{s} can not result from quantization).

To determine its own position, our agent uses its neck angle, its body orientation and flag landmarks. From the raw neck angle $\hat{\phi}_{\text{neck}}$ as received by the agent, an interval $\Phi_{\text{neck}} = Q^{-1}(\hat{\phi}_{\text{neck}})$ of possible neck angles is computed. The orientation of the view cone is obtained from the observed angle of the field border lines, again yielding an interval Φ_{view} . Similar to arithmetic with single-valued variables, the angle interval for the agent orientation can be calculated to give $\Phi_{\text{orient}} = \Phi_{\text{view}} \oplus (-\Phi_{\text{neck}})$. The operator $-$ denotes inversion of an interval at the origin, reversing its orientation, and \oplus is the convolution operation when interpreting the intervals as probability distributions; for the sake of simplicity, however, we implemented \oplus not as convolution but as interval arithmetic $+$.

The landmark flag position data sent by the server give quantized values for distance and angle. Reconstruction via inverse quantization gives an interval for the possible true distances and one for the possible true angles of the flag. Given these, the set of possible relative flag positions can be reconstructed (sector in

Fig. 1 marked by solid line). The sector is then extended to the polygon marked by dashed lines to simplify further calculations. The same way the computation for Φ_{orient} traces the computation of a single-valued variable for orientation, appropriately rotating P_{flag} (taking into account the interval structure of Φ_{view}) and shifting the result w.r.t. the absolute flag position, we calculate a polygon area P_{player} of possible player positions. This process is repeated for every flag seen during the current time step. The true player position must be consistent with all of them. Therefore we compute the intersection of these regions (applying a probabilistic formalism would also be possible, but for sake of simplicity we selected a interval, or better, “polygon arithmetic” framework). An estimate for the position of the player is then obtained by averaging the corner positions of the intersection polygon. We found this polygon usually to be quite small and the resulting estimated positions to deviate around 5-10 cm from the true positions, which is more than an order of magnitude more accurate than the values obtained by conventional approaches.



Fig. 1. Sector as defined by an orientation and distance interval

We are aware of the fact that CYBEROOS independently developed an approach similar to the one described in this section to estimate player position (private communication, Amsterdam 2000); so did the MAINZ ROLLING BRAINS (private communication, Melbourne 2000). To our knowledge, they restricted themselves to the calculation of the agent position.

2.2 Ball Position and Velocity Estimation

The generality of the EWE method allows us to determine ball movement in a conceptually similar fashion. Here, the approach of incorporating evidence needs to be applied to the phase instead of position space. To simplify the exposition, we limit ourselves to explain a situation where the player is placed at the origin of the coordinate system and looks along the x -axis and the ball is moving exclusively (and undisturbedly) along the x axis, the y -component of its movement vanishing. The equation of movement during a single time step for an undisturbed ball in the soccer server is given by

$$\begin{pmatrix} x(t+1) \\ v_x(t+1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & \lambda \end{pmatrix} \begin{pmatrix} x(t) \\ v_x(t) \end{pmatrix} \quad (1)$$

where λ is the decay rate for the ball movement. The ball state is completely specified by the pair x, v_x . If the ball state is unknown at the initialization, the ball

can be at any x -position inside the field and have any velocity $v_x \in [-v_x^{\max}, v_x^{\max}]$ where v_x^{\max} denotes the maximum speed. This uncertainty defines a rectangle in phase space. When ball information is observed, one obtains a distance (and an angle which vanishes in our example). This distance defines a distance interval, restricting the possible states in phase space to a rectangle relatively narrow compared to the a priori possible x -positions. The speed, however, is still unknown, i.e. the rectangle has still its full extent in v_x -direction. After one time step, the rectangle of expected possible states in phase space for $t + 1$ becomes oblique due to Eq. (1). New evidence cuts a vertical slice from this rectangle; by virtue of its obliqueness, the feasible region in phase space is now much smaller, giving a much better defined value for velocity. In the following steps the size of the feasible region then quickly shrinks, greatly improving localization precision.

Acknowledgements

We would like to particularly mention all the members of the team LUCKY LÜBECK, namely Kord Eickmeyer, Jan Balster, Jan Hendrik Sauselin, Nima Madershahian, Martin Haker and Tobias Kochems. The first author wishes to convey his thanks to the MAINZ ROLLING BRAINS team for the productive work done together that formed the basis for the present research and for the warm atmosphere during his time as their team leader.

References

- Daniel Polani and Thomas Uthmann. Between Teaching and Learning: Development of the Team MAINZ ROLLING BRAINS for the Simulation League of RoboCup '99. In *Proc. of the 3rd RoboCup Workshop, Stockholm*, 1999.
- D. Polani, S. Weber, and T. Uthmann. A Direct Approach to Robot Soccer Agents: Description for the Team MAINZ ROLLING BRAINS, Simulation League of RoboCup '98. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999.
- D. Polani, S. Weber, and T. Uthmann. A Direct Approach to Robot Soccer Agents: Description for the Team MAINZ ROLLING BRAINS, Simulation League of RoboCup '98. In Minoru Asada, editor, *Proc. of the second RoboCup Workshop, Paris, July 1998*, pages 325–329. The RoboCup Federation, 1998.
- F. L. Lewis. *Optimal Estimation - with an introduction to stochastic control theory*. John Wiley & Sons, Inc., 1986.
- Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99); Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence*, pages 343–349, Menlo Park, Cal., July 18–22 1999. AAAI/MIT Press.
- Hans-Dieter Burkhard, Markus Hannebauer, and Jan Wendler. Belief-Desire-Intention Deliberation in Artificial Soccer. *The AI Magazine*, 1998(3):87–93, 1998.
- Anand S. Rao and Michael P. Georgeff. BDI Agents: from theory to practice. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems*, pages 312–319, San Francisco, CA, 1995. MIT Press.

Karlsruhe Brainstormers 2000 Team Description

Martin Riedmiller, Artur Merke, David Meier, Andreas Hoffmann, Alex Sinner,
Ortwin Thate

Institut für Logik, Komplexität und Deduktionssysteme University of Karlsruhe,
D-76128 Karlsruhe, FRG

1 Introduction

The main motivation behind the Karlsruhe Brainstormer's effort in the robotic soccer simulation league of the RoboCup project is to develop and to apply Reinforcement Learning (RL) techniques in complex domains. Our long term goal is to have a learning system which is able to learn by itself the best winning behaviour. The soccer simulation domain allows more than $(108 \times 50)^{23}$ different positionings of the 22 players and the ball - the complete state space considering object velocities and player's stamina is magnitudes larger. In every cycle, an agent can choose between more than 300 basic commands (parametrized turns and dashes), which makes a choice of 300^{11} joint actions for the team *per cycle!* A problem of such complexity is a big challenge for today's RL methods; in the Brainstormers project we are investigating methods to practically handle learning problems of such size.

2 Special Team Features

The outstanding features of the Karlsruhe Brainstormers Agent are its approach to learn the moves with RL methods and the search algorithms used on the strategy level for planning complex coordinated team moves such as pass chains. It is our aim to have one day an agent which has learned its whole behaviour through RL methods. For further information see [1], [2], [3].

3 World Model

Construction of an own world model was not a main focus of our development. Therefore we decided to use parts of the world model sources of the CMU team from 1999 [4]. This world model is hidden behind our own interface, so we can switch one day to our own world model. We also added some communication, so that players can exchange their knowledge about positions of other players (which considerably improves the quality of the world model information). From the tactics point of view it is also interesting to know how the players play with full and noiseless world model information. To this end we use a slightly modified soccer server which communicates the complete noiseless state information to our players. As the construction of the world model is hidden behind an interface we

can choose to play with or without a complete world model. For the development of RL moves we also use another patch of the official server which allows us to run the server faster i.e. without waiting the 100ms each cycle, but just waiting until the agents have sent their commands. In conjunction with the full information patch we have a powerful learn development tool at our disposal.

4 Coach

The Brainstormers 2000 client did not make use of an online coach, although a coach is used to automate unsupervised playing of games. This allows us to test our team in say ten or more games against several other teams and gather statistics.

Another tool which uses coach commands is Frameview2d. This is a viewer for moving two dimensional graphics. It uses frames for the representation of geometrical data (like points, lines, circles). For example, it is possible to put contours of a robot into a frame and later on just to specify the frame position and angle, all coordinates conversion are done automatically. For a longer list of features see <http://i11www.ira.uka.de/~amerke/frameview>. By now there is a module which can be used instead of the official soccer monitor of the simulation league. By using the zoom facility it can be used e.g. for monitoring or debugging of moves. So it is possible to watch a kick sequence produced by our RL kick move or the turns and dashes during our RL intercept ball move. We also use it to edit game situations which can be saved and then reproduced with our coach tool. It can be easily extended to visualize other 2d worlds, e.g. it would be a straightforward task to use it as a monitor of a middle/small size league robot.

5 Communication

Our agents communicate basically two different kinds of information : first, if an agent makes an important decision affecting teamplay (such as a pass), he communicates this decision so that the receiving player knows he will receive a pass and can therefore act faster than his opponents. Otherwise the agents simply communicate their world information so every player is able to have a better view of the world.

6 Skills

Most of the nontrivial skills are learned with RL methods. We refer to them as moves. A move is a sequence of basic actions. Since each move has a clearly defined goal, it is straightforward to apply RL methods to solve the problem of getting an optimal decision policy. We applied Real-Time Dynamic Programming methods that solve the learning problem by incrementally approximating an optimal value function by repeated control trials. Since the state space is continuous, a feedforward neural network is used to approximate the value function.[1]

7 Strategy

Altough it is our long-term goal to learn a high level strategy with RL methods, our research is currently not advanced enough to use a learned strategy efficiently in our client. Nevertheless this does not mean we did not develop any strategy. Basically the client chooses from two different strategies : a strategy which considers that he has the ball and a strategy for positioning if he doesn't have the ball. In the latter case the team switches the formation according to whether it is attacking, defending or in midfield play. In this formation every player is assigned a radius of action in which he may move around to perform passive tasks such as defend against an attacking opponent, receive a pass etc. If the player is in possession of the ball, he uses a planning algorithm which searches for pass chains. If no good passes can be played (or no goal shot), he looks for alternatives like dribbling, holding the ball etc.

8 Team Development

The development team for the Karlsruhe Brainstormers 2000 client consisted of 6 people. We meet once a week to discuss more or less important matters and coordinate the client development. After the championships in Stockholm 1999 we decided to entirely redesign our client, so we rewrote most of the code from scratch, implementing advanced software engineering concepts such as object handlers. The client is thoroughly modularized, which makes it easy to test and implement new strategies, moves etc. The decision process is organised hierarchically : on the top level a strategy is selected at the beginnig of the game (e.g. goalie or 'normal' player). This strategy selects a policy according to the playmode and overall situation. The policy defines how the client has to behave, i.e. he selects a move. A move is a sequence of basic actions, so it returns an action which is sent to the server. Furthermore, since four types of commands can be sent in parallel to the server (main commands, turn-neck commands, say commands and change-view commands) we have four parallel decision chains.

To coordinate our development we use CVS, a powerful revision control system. To generate documentation from our code we use doxygen. Everything is developed on Linux machines.

Team Leader: Dr. Martin Riedmiller

Team Members:

Artur Merke

- Computer Sciences
- Germany
- graduate student
- did attend the competition

David Meier

- Computer Sciences
- Germany

- graduate student
- did attend the competition

Andreas Hoffmann

- Computer Sciences
- Germany
- undergraduate student
- did attend the competition

Alex Sinner

- Computer Sciences
- Luxembourg
- undergraduate student
- did attend the competition

Ortwin Thate

- Computer Sciences
- Germany
- undergraduate student
- did attend the competition

Web page <http://www.karlsruhe-brainstormers.de>

9 Conclusion

Our success (vice-champion) at the championships in Melbourne showed us that the RL approach is very promising. We plan to extend its use to larger scale problems such as positioning, planning and teamplay. Distributed RL will be one of our main research interests for the Brainstormers 2001 project. And yes, we will be participating at the RoboCup world championships in Seattle 2001!

Finally we want to thank all our friends, family and all the people who supported us in any way.

See you in Seattle!

References

1. M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, C. Kill and R. Ehrmann. Karlsruhe Brainstormers - A Reinforcement Learning approach to robotic soccer. In A. Jennings, P. Stone, editors, *RoboCup-2000: Robot Soccer World Cup IV, LNCS* Springer Verlag, Berlin, 2000.
2. A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, (72):81–138, 1995.
3. M. Riedmiller. Concepts and facilities of a neural reinforcement learning control architecture for technical process control. *Journal of Neural Computing and Application*, 8:323–338, 2000.
4. Peter Stone, Patrick Riley and Manuela Veloso. The CMUnited-99 Champion Simulator Team In M. Veloso, E. Pagello and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*. Springer Verlag, Berlin, 1999.

ATT-CMUnited-2000: Third Place Finisher in the Robocup-2000 Simulator League

Patrick Riley¹, Peter Stone², David McAllester², and Manuela Veloso¹

¹Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
pfr,mmv@cs.cmu.edu
<http://www.cs.cmu.edu/> pfr, mmv

²AT&T Labs — Research
180 Park Ave.
Florham Park, NJ 07932
dmac,pstone@research.att.com
<http://www.research.att.com/> dmac, pstone

1 Introduction

The ATT-CMUnited-2000 simulator team finished in third place at Robocup-2000. It was one of only three teams to finish ahead of the previous year's champion, CMUnited-99. In controlled experiments, ATT-CMUnited-2000 beats CMUnited-99 by an average score of 2.6–0.2.

ATT-CMUnited-2000 is based upon CMUnited-99 [4], which in turn is based on CMUnited-98 [5]. In particular, it uses many of the techniques embodied in the previous teams, including: Flexible, adaptive formations (Locker-room agreement); Single-channel, low-bandwidth communication; Predictive, locally optimal skills (PLOS); and Strategic positioning using attraction and repulsion (SPAR).

The ATT-CMUnited-2000 team explores several new research directions, which are outlined in the sections below.

2 On-Line Planning for Set Plays

CMUnited-99 uses a variety of “set-plays” for the various dead-ball situations (goal kick, free kick, corner kick, etc.). However, the set plays have very little variation among executions. ATT-CMUnited-2000 uses dynamic, adaptive planning for set-plays. The coach for ATT-CMUnited-2000 makes a new plan for player and ball movement at every offensive set play.

A plan is represented as a Simple Temporal Network, as introduced in [2]. A Simple Temporal Network (or STN for short) is a directed graph that represents the temporal constraints between events. For example, some of the events used here are “start pass”, “start goto point”, and “end goto point.” The temporal constraints are used to represent the parallelism inherent in having multiple agents as well as allowing the agents to detect failures during plan execution.

Muscettola *et al* have described a “dispatching execution” algorithm for STN execution[3]. The algorithm allows easy and efficient propagation of temporal

constraints through the network. Our execution algorithm extends the dispatching execution algorithm to the multi-agent case. We use the dispatching algorithm (unchanged) to maintain information about time bounds for executions of events. Every cycle, the agents monitor conditions on plan execution and work towards bringing about the execution of events in the appropriate order.

In order to create plans, we use models of opponent movements. Because of the short time span in a simulated soccer game, we decided to fix models ahead of time and to choose between them during the game. Selecting a model should be much faster than trying to actually create a model online. Conceptually, we want an opponent model to represent how an opponent plays defense during setplays. We expect that a wide range of decision-making systems of the opponent can be roughly captured by a small set of models. In order to handle the uncertainty in the environment and to allow models to represent more information than just a simple predicted location, we use probabilistic models. Given the recent history of the ball's movement (from the start of the set play for example) and the player's initial location, the models give, for each player, a probability distribution over locations on the field.

In order to use a naive Bayes update to select between models during the game, we make the following assumptions. While the assumptions are probably not completely correct, we expect that we can still get reasonable model selection with these assumptions.

1. The players movements are independent. That is, the model may generate a probability distribution for player x based on everyone's starting locations. However, what the actual observation is for player x (assumed to be sampled from this probability distribution) is independent from the actual observations of the other players.
2. The probability of a particular starting position and ball movement are independent of the opponent model. This assumption is questionable since the ball movement (i.e. plan) generated depends on the opponent model.

However, the model which collectively makes the best prediction may not be the model we would like to use for planning. We want to use the model that predicts the players closest to the ball the most accurately because the planning will depend mostly on players closest to the ball. The model that most accurately captures those players should give us the best plans from the planner. Further, we would like to have a smooth transition in how the players the players affect the probability of a model as their distance changes. We have created a weighting scheme which can be applied to the naive Bayes update in order to achieve these properties.

The planning process largely consists of solving a path planning problem with moving, probabilistic obstacles (where the obstacles are the opponents). We use a hillclimbing approach which balances the safety of a plan with the benefit obtained (in terms of final ball location) if the plan succeeds.

For defensive set plays, the coach assumes that the plan which the opponents execute will be similar to previous set plays. The coach instructs players to go to locations which would be good if the ball and player movements were identical to

previous plays. The coach does this by drawing a boundary around the current ball position and tracking where the ball crosses this boundary.

3 Computing Interception Times

A significant difference between ATT-CMUnited-2000 and its predecessors is that ATT-CMUnited-2000 introduces the concept of a *leading pass*. In the past, most teams have only considered passing the ball directly to a teammate. However, there is a potentially large advantage to be gained by passing the ball in such a way that the teammate can “run on” to the ball.

In ATT-CMUnited-2000 the player holding the ball considers hundreds of possible kick directions and kick powers and, for each hypothetical kick, calculates the time for each other player (both teammate and opponent) to reach the ball. This entire calculation is done once each simulation cycle whenever the player is holding the ball. Forward simulation methods that reason about discrete simulator cycles, such as those used by past champion teams Humboldt [1] and CMUnited [5], are not computationally efficient enough to evaluate hundreds of potential passes (although they are very effective and tractable for evaluating small numbers of passes). ATT-CMUnited-2000 abstracts the simulation to continuous time and uses a modified Newton’s method to efficiently approximate earliest interception times.

For an arbitrary ball position B_0 , initial ball velocity V_0 and receiver position R_0 , we want the least time required for the receiver to reach the ball assuming that time is continuous, the receiver can run at a fixed velocity of V_r starting immediately, and the ball continues in its current direction with instantaneous velocity after time t given by $V = V_0 e^{-t/\tau}$. The receiver velocity V_r is taken to be the maximum player velocity in the simulator — one meter per second. The velocity decay parameter τ is adjusted to match the simulator at the discrete simulation times — τ is about 2 seconds. The position of the ball after time t , denoted $P(t)$, can be written as follows.

$$P(t) = B_0 + V_0 \tau (1 - e^{-t/\tau}) \quad (1)$$

The distance between the initial receiver position and the ball position at the interception time must equal the distance the receiver can travel. This condition can be written as follows where $\|X\|$ denotes the length of vector X .

$$V_r t = \|R_0 - P(t)\| \quad (2)$$

To formulate the problem in a way appropriate for Newton’s method we first define $g(t)$ as follows.

$$g(t) = \|P(t) - R_0\| - V_r t \quad (3)$$

Condition (2) can now be written as $g(t) = 0$. Unfortunately, this condition often has three different roots. To find only the smallest root we set t^0 to be the d/V_r where d is the distance from R_0 to the line defined by the point B_0 and the vector

V_0 . This gives that t^0 is no larger than the smallest root. The standard Newton's method update sets t^{i+1} to be s^{i+1} as defined by the following equation.

$$s^{i+1} = t^i - g(t^i)/g'(t^i) \quad (4)$$

We now modify this update rule to ensure that if t^i is no larger than the smallest root then t^{i+1} is also no larger than the smallest root. Let $U(t)$ be the unit vector in the direction from R_0 to $P(t)$. We can rewrite equation (4) as follows.

$$s^{i+1} = t^i + g(t^i)/(V_r - e^{-t^i/\tau} V_0 \cdot U(t^i)) \quad (5)$$

Our modified update rule is the following where s^{i+1} is defined by equation (5).

$$t^{i+1} = \begin{cases} s^{i+1} & \text{if } V_0 \cdot U(t) < 0 \\ t^i + g(t^i)/(V_r - e^{-t^i/\tau} V_0 \cdot U(s^{i+1})) & \text{if } V_0 \cdot U(t) > 0 \text{ and } g'(t^i) < 0 \\ t^i + g(t^i)/V_r & \text{otherwise} \end{cases}$$

A proof is available upon request that the sequence t^0, t^1, t^2, \dots converges to the smallest root of (2).

4 Conclusion

In empirical tests, ATT-CMUnited-2000 beat CMUnited-99 by an average score of 2.6–0.2 over the course of 33 10-minute games. There are two major improvements to ATT-CMUnited-2000 which helped it achieve these results and place third at RoboCup2000. The online coach is used to make a new plan for each setplay. The planning algorithm uses a model of the opponent's movements in order to evaluate generated plans. Given a set of models before the game, the coach selects which model best describes the current opponent. The players also use a fast numerical technique in order to evaluate hundreds of kicking options when they have the ball. Thus, they are able to execute successful leading passes, rather than only passing directly to teammates.

References

1. H.-D. Burkhard, M. Hannebauer, and J. Wendler. AT Humboldt — development, practice and theory. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 357–372. Springer Verlag, Berlin, 1998.
2. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
3. N. Muscettola, P. Morris, and I. Tsamardinos. Reformulating temporal plans for efficient execution. In *KR-98*, 1998.
4. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In Veloso, Pagello, and Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 35–48. Springer, Berlin, 2000.
5. P. Stone, M. Veloso, and P. Riley. The CMUnited-98 champion simulator team. In Asada and Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 61–76. Springer Verlag, Berlin, 1999.

Headless Chickens IV

Paul Scerri, Nancy Reed, Tobias Wiren, Mikael Lönneberg and Pelle Nilsson

Institute for Computer Science, Linköpings Universitet

1 Introduction

The Headless Chickens IV (HCIV) were developed using a prototype end-user agent development environment called EASE (End-user Actor Specification Environment). The experience provided a great deal of data about general and domain specific properties of the system. A secondary research goal was to generalize the strategy editor used to specify the team strategies of last years team to specify team strategies for agent of different types via the use of an XML interface. The poor performance of the team (3 losses and a draw in the competition) is attributed to the early stage of the research as well as the very high computational complexity of the agent runtime architecture.

2 Special Team Features

In 2000 the aim of the work on the Headless Chickens RoboCup soccer team was to use an existing end-user actor development system in a second domain. EASE is a prototype system developed to allow relative end-users to specify the behavior of intelligent agents for iterative simulation environments[7]. Until now EASE has been primarily used for developing air-combat agents for a simulation environment called TACSI. Using EASE in a second domain allowed us to learn more about the strengths and weaknesses of the underlying architecture and, in particular, learn which ideas could be generalized and which could not.

The EASE runtime engine is essentially a behavior-based architecture[3,1] with a few enhancements to make it easier for relative end-users to specify the behavior of complex actors in the EASE development environment. Development of players is completely graphical once a few Java classes have been created to interface EASE to the simulation environment.

One of the key features of EASE is an interactive view of the internal reasoning of a player. The viewer, nick-named *The Boss*, not only shows the hierarchical arrangement of behaviors in the actor at runtime but also allows users to start, stop and pause behaviors at runtime[8]. This means that while testing an agent the user can interactively manipulate the agent's internal reasoning to determine the effect of different potential specification changes. Also, the user can interactively get the agent to carry out specific tasks at runtime even though the agent as specified would not have done that task.

During the development of the HCIV, *The Boss* was found to be very useful for helping to incrementally develop and debug players. For example, when the

players were doing something wrong behaviors could be started one at a time allowing the developer to determine which behavior(s) was causing the problem. Also the developer could work effectively with incompletely specified players by interactively issuing commands, for example if no strategies had yet been defined for handling free kicks when a free-kick occurred the developer could interactively direct a player to chase and kick the ball.

Behavior-based architectures use a variety of methods to determine a final agent action given a number of competing behaviors[6]. EASE uses a method based on an algorithm by Pirjanian[4], primarily designed for robots. Pirjanian's algorithm relies on checking all possible actions, albeit in an efficient manner. In real-time interactive simulations running on normal hardware such an approach is unacceptably slow. EASE uses a probabalistic, anytime algorithm[10] to find reasonable actions quickly then incrementally find better actions, if time permits. One property of the algorithm is that when the situation changes dramatically it may take longer than one cycle for the agent to determine the optimal action, but it will take whatever action it has found after the length of one cycle. For air-combat simulation this is acceptable as actions usually involve picking a new heading or speed which can be subsequently slightly adjusted as better headings or speeds are found, e.g. in the first cycle selecting a heading of 127° then in the next cycle finding a slightly better heading of 125° before finally settling on 123°. Such adjustments are not noticed as the turn will usually take many cycles to execute. However, the algorithm ran into problems in the RoboCup domain when trying to kick the ball. The first action the agent takes changes the situation dramatically not allowing for slowly improving the action. Until some "hacks" were introduced the agent would often kick the ball in very strange directions or with strange power values – simply because the action determination algorithm did not have enough time to find a better action. This experience has led us to look to more sophisticated probabilistic techniques such as simulated annealing or genetic algorithms for searching for a good action[2, 5].

3 World Model

The HCIV world model saves very little information from cycle to cycle and does very little reasoning on the information it has. The information that is stored across cycles is usually in relation to objects that may not be seen in the next cycle. The simplicity of the world model is primarily due to the implementation being so new, not any design philosophy.

A feature of the world model is a viewer to show at runtime of most of the agent's calculations, in intimate detail without the designer having to embed debugging code. The *Calculation Debugger* provides a useful tool for investigating problems with the agents behavior very quickly. However, the Calculation Debugger relies on a very computationally expensive method of doing calculations which slows a agent's overall behavior.

4 Communication

There is no communication between players. Each player knows the situations that will lead to different team strategies and the triggers for changing strategies are designed to minimize confusion, e.g. by using general ball position or referee calls. Communication might be used to improve the players model of the world, however so far we have found that when accurate information is required, for example when dribbling, the relevant player has best access to that information. In cases when other players have the best information it is usually not so important that the player has very accurate information. One exception to this would be for passing but our team strategy of passing to positions from the pre-defined strategies make this irrelevant.

5 Skills

The skills of the players are completely specified within the EASE graphical development environment. In the RoboCup domain, building skills in EASE has a significant advantage and a significant disadvantage over programming in low level code. The main advantage is that EASE enforces a good functional breakdown where each function must be in a separate behavior. For example dribbling the ball and obstacle avoidance will be in separate behaviors. The behavior fusion algorithm automatically combines the two functions. On the negative side graphical specification was not well suited to the types of calculation that needed to be done in player skills (a mathematical specification style may have been better suited).

6 Strategy

A secondary research goal of the HCIV development was to test the generality of the team level strategy editor developed as part of the 1999 team (HCIII). The strategy editor allows high level specification of team strategies and has been found to be very useful[9]. In 1999 the editor was closely coupled to a particular behavior based architecture that was being used. This year an XML import and export interface was created for the editor allowing a variety of different agents to use the high level strategy specification. EASE agents (i.e. HCIV agents) as well as HCIII agents worked with the same high level specification.

The XML files can be read (and interpreted) by a variety of different agent types. The input XML files specify the modes of play that the agents know about (e.g. defensive corner kick), the types of actions that the agents can take in each mode (e.g. pass and dribble) and the available styles of play (e.g. defensive or attacking). The output from the editor is 11 XML files, one for each player detailing the positions the players should play in each mode, the style of play they should adopt in each mode and any actions the agents should take in each mode, e.g. where and when to pass, dribble and shoot.

7 Team Development

The HCIV development team consisted of 5 people from Linköpings Universitet in Sweden. About two man months were spent interfacing EASE and RoboCup and developing EASE player specifications by three under-graduate students, Tobias Wiren, Mikael Lönneberg and Pelle Nilsson. EASE was primarily developed by HCIV team leader Paul Scerri under the supervision of Nancy Reed.

Eight of the players that actually played in the 2000 World Cup games were untouched HCIII players and three were EASE players, primarily because the very high computational load of the negotiation algorithm prevented more HCIV players being used. All RoboCup specific aspects of the EASE players were developed from scratch except for server interface code which was adapted from HCIII. Most of the basic agent specification and agent runtime code of the HCIV was used without change from that used to produce simulated air-combat pilots.

The work was part of a larger project, funded by Saab Aerospace, NUTEK and CENIT, aimed at developing techniques for end-user specification of intelligent agents for simulation environments.

7.1 Relevant Webpages

See the Headless Chickens webpage at <http://www.ida.liu.se/~pausc/RC99-main.html> and the EASE webpage at <http://www.ida.liu.se/~pausc/Project.html>.

References

1. Rodney Brooks. Intelligence without reason. In *Proceedings 12th International Joint Conference on AI*, pages 569–595, Sydney, Australia, 1991.
2. Petru Eles, Krzysztof Kuchcinski, and Zebo Peng. *System Synthesis with VHDL*, chapter 5 "Optimization Heuristics". Kluwer Academic Publisher, December 1997.
3. Maja Mataric. Behavior-based systems: Main properties and implications. In *IEEE International Conference on Robotics and Automation, Workshop on Architectures for*, pages 46–54, Nice, France, May 1992.
4. Paolo Pirjanin. *Multiple objective action selection and behavior fusion voting*. PhD thesis, Dept. of Medical Informatics and Image Analysis, Aalborg university, 1998.
5. C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, London, 1993.
6. Julio Rosenblatt and Charles Thorpe. combining multiple goals in a behavior based architecture. In *Proceedings of 1995 International Conference on Intelligent Robots and Systems (IROS)*, Pittsburg, PA, 1995.
7. Paul Scerri and Nancy E. Reed. Real-time control of intelligent agents. In *Technical Summaries of the Software Demonstration Sessions, Agents 2000*, pages 28–29, June 2000.
8. Paul Scerri, Nancy E. Reed, and Anders Törne. An approach to directing intelligent agents in real-time. In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, pages 114–115, 1999.
9. Paul Scerri, Johan Ydrén, and Nancy E. Reed. Layered specification of intelligent agents. In *Sixth Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)*, pages 565–575, August 2000.
10. S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 1996.

Mainz Rolling Brains 2000

Birgit Schappel, Frank Schulz

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

1 Introduction

The team Mainz Rolling Brains participated in RoboCup world championships for the third time this year. The team is based on the improved technical level from last year but has a completely new decision level. The latter level was modularized and we introduced communication for world model and tactical information.

2 Special Team Features

Confronted with the "problem" of a large developer team we had to find a way to make it possible for all team members to take part in the development process. Having a well tested technical layer from last year's agent we could concentrate on the improvement of the world model, the introduction of a tactical model and the modularisation of the decision layer.¹ The first difficulty of the modularisation approach is to detect suitable tasks within the agent. The classification in five tasks as shown in figure 1 is quite natural and worked well.

Each task corresponds to a module. The modules evaluate the adequateness and the chance of success of a particular action in the current situation and return a grade representing the rating of the chosen action. The module with highest grading will be called to act and is responsible for the execution of the action. The Master Control coordinates the evaluations and actions of the different modules in the decision layer.

The basis of decision for the modules is the world model (see section 3) and the tactical model. The latter provides tactical information like

- BallControlState (SAFE, DANGER, LOST),
- TacticalActionState (CONTROL, PASS, GOALKICK, REACT),
- the time an opponent or a team member will need to reach the ball,
- the three players of the own and of the opponent's team who can reach the ball first.

The tasks of the individual modules are explained below. As we did not develop heterogenous players the different modules include special routines for the goalie if necessary:

¹ The whole structure of our agent can be found in [1].

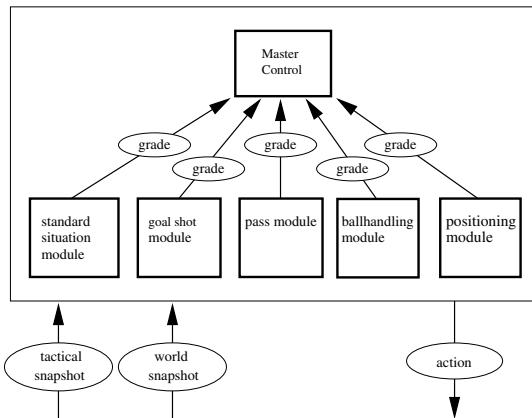


Fig. 1. Agent Control

- Standard situation module: Handling of standard situations like kick off, corner kick and free kick as well as goal kick.
 - Goal shot module: Obvious.
 - Pass module: Selection and realisation of passes to other players or into free space.
 - Ballhandling module: Decides whether to intercept the ball, to dribble or to tackle opponents. The module also provides special routines for the goalie, especially catching and tackling routines.
 - Positioning module: Positioning of the players on the field depending on the current situation of the game as well as constructing and avoiding offside traps. This module is also responsible for controlling the stamina in the long term.

A problem of the modularisation approach is to adjust the grading returned by the different modules to each other. This process and the testing took a considerable amount of time. To reproduce the decisions of the modules we developed a graphical tool called replayer that shows for a fixed timestep or a sequence of timesteps the exact server data, the players world model and the players decision trace for a game that was recorded.

3 World Model

Agents store data of all objects on the field for a time window of about 30 steps. The players can retrieve snapshots for the current timestep and even timesteps in the future which is provided by a simulation component. Objects that are not seen will also be simulated. They may be updated by inter-player communication as well. To handle server asynchronicity agents wait for the first sense body message in every new cycle before they update the world model.

Each object comes with a quality value indicating the reliability of the data so that it is possible to discern uncertain information.

4 Coach

The online coach is used to handle standard situations. He communicates his world model, in particular the position of the ball. The players use this information for planning free kicks, kick-ins and goal kicks. At the moment the coach does not make any tactical decisions about e.g. the formation or positioning of the team or the opponent. We also used the offline coach for experimental and training set-ups. By constructing special game situations it was possible to test single modules.

5 Communication

Communication is an important feature of this year's team. Players decide on their own when to communicate with each other, but are restricted to certain *communication channels*. In general a message from a player contains its world model and its tactical model. Each information a player has about the world also contains a reliability value. Based on this value other players hearing the message rate the usefulness of the information transmitted for their sight of the game. We are using five communication channels (simulation step modulo 5):

- First channel: emergency channel
This channel is used when a player changes the tactical action state of the game e.g. when he passes the ball to a team mate.
- Second channel: special situation channel
This channel is only used by the player who believes that he has the best view of the ball. Typically this is not the player controlling the ball but the one next to this player.
- Third channel:
No communication is allowed on this channel at all to ensure that the messages sent via the fourth channel are not blocked by the server.
- Fourth channel: world model channel
Some players (either the players with even numbers or the players with odd numbers - to reduce network traffic) tell their world model and their tactical model "to whom it may concern".
- Fifth channel:
No communication is allowed here either to ensure that the messages sent via the emergency channel are not blocked.

6 Skills

When talking about our agent's skills we have to distinguish between the low-level (kicks in general, super-kick, dribble, intercept) and the high-level skills (passes, positioning, attacking an opponent).

The low-level skills of the agents are based on last year's skills. We did not use learning techniques for ball control, e.g. shooting, dribbling, and placing, but analytically solved that convex optimization problem. All boundary conditions (kick not harder than allowed, keep ball out of collision etc.) can be described by circles. So it is sufficient to get the best solution out of a finite number (24) of velocities.

To determine a target position for a pass (i. e. a team member or a point on the field) the player considers a corridor of a certain size between his own and the target position. Good passes are passes with opponent-free corridors and suitable targets.

Positioning on the field consists of two parts: The first part comes from strategical formation of the whole team, i.e. the players keep their relative positions to each other. Subsequently the players try to optimize their positions in the allowed area by a gradient method.

7 Team Development

As mentioned above this year's team is based on the 1999 Rolling Brains. We would like to thank our former team leader Daniel Polani for his support and commitment for the Mainz Rolling Brains. He helped a great deal in setting up our group and in finding some sponsors, as well as in composing our team description for RC 2000 ([1]). We wish him all the best personally and for his new team Lucky Luebeck. Last but not least we want to thank Thomas Uthmann for his support of our student team:

Team Leader: Thomas Uthmann and Frank Schulz (A)

Team Members: ² Axel Arnold (A), Felix Flentge, Manuel Gauer, Erich Kutschinski, Christian Meyer (A), Birgit Schappel, Christoph Schneider, Holger Schneider, Ralf Schmitt (A), Goetz Schwandtner (A).

Web page: <http://nautilus.informatik.uni-mainz.de/~robocup/>

8 Conclusion

The idea of the modularisation of the decision layer turned out to work well, thereby providing a frame for distributed code development and facilitating the contribution of code to the agent by all of the team members mentioned above. For the next year we are looking for a way to speed up detecting wrong decisions based on inadequate grading. Furthermore the role of the online coach shall be extended, e.g. by constructing opponent models.

References

1. THOMAS UTHMANN, CHRISTIAN MEYER, BIRGIT SCHAPPEL, FRANK SCHULZ
Description of the Team Mainz Rolling Brains for the Simulation League of RoboCup 2000, <http://nautilus.informatik.uni-mainz.de/~robocup/>

² All students are graduates; A indicates attendance at RoboCup '00

Team Description of Spatial-Timer

Kousuke Shinoda, Susumu Kunifugi

Graduate School of Knowledge Science, Japan Advanced Institute of Science and
Technology Ishikawa 923-1292, Japan
{kshinoda | kuni}@jaist.ac.jp

1 Introduction

Our team aims at the improvement of soccer agent's ability of situation recognition and action decision by saving environmental information as a spatial image. In real time simulation, such as soccer game, the agent needs to have all environmental information in order to make a right judgement of situation and make decision own action[Ito99, Nitta88]. However, soccer agents are limited the sight information are acquired from soccer server[SS98]. Then, our team aims at the improvement of agent's ability of situation judgement with saving of the spatial image in the spatio-temporal database[Erwing98]. As a result, soccer agent can make judgement own environment with not only sight information from soccer server but environment outside the sight range used by this spatio-temporal database. However, the information, which is given from soccer server, is uncertainty and incompleteness. Therefore, it is difficult for agent to make the spatio-temporal database. Then, when a spatial information was saved in the database, the agent made to complete the lack information used by the difference of these items of information in our team. As a result, our agent can make a judgement own environment with virtual spatial information of the spatio-temporal database, whose information is wider than the range of normal range of the agent. The following, this paper explains our spatio-temporal database and the methods complement of lack information in this database.

2 Team Architecture: Spatial Temporal Database

2.1 Spatio-Temporal Database

In the following points, there are advantage points for the soccer agent, which uses the spatio-temporal database[Erwing98].

- The agent can generate Time series information of a specific soccer player by accumulating spatial information.
 - It is possible to accumulate and analysis of soccer agent's dynamic element.
 - It is possible to infer the internal condition of the other agent and forecast the transition of the event in soccer field.
- The agent completes a lack information of an object, such as uniform number or team name of player in soccer game.

- The soccer agent becomes to be able to use the other player's information with lack information, because the database completed it. As a result, the soccer agent has the increasing amount of usable sight information.

2.2 Component of spatio-temporal database

In this paper, the spatio-temporal database is composed of the following elements.

- **Temporal Data :** This is one of spatial information at specific time. The Spatio-Temporal database accumulates this temporal data according to the time series.
- **Time Map Card :** This is data format, which saves time and position of the soccer player in the other player's information.
- **Time Map :** “TimeMap” is data format to store time series information of one object. The length of time series information is not fixed but dynamic with length more than constancy.
- **Spatio-Temporal Database :** This saves all players’ “TimeMap” on the soccer field. This is characterized by which manage data with two different attributes, such as temporality and spatiality.

2.3 Working flow of processing of the spatio-temporal database

The following five processes compose process of spatio-temporal database.

1. **Translation:** New information is classified to each object's data, its data is stored in TimeMapCard.
2. **Verification:** The hypothesis is verified.
3. **Identification:** The object information is identified to store generated “TimeMapCard” in appropriate TimeMap. The identification mechanism of the object is described in detail in section 2.4.
4. **Complement:** The object with lack information is complete with the hypothesis derived by identification mechanism.
5. **Insertion:** Accumulation of identified “TimeMapCard”.

2.4 Identification mechanism of the Object

Fig.1 is example of a TimeMap as for soccer agent player. In its figure, One player S had visual information of three soccer agent players from time t-2 to time t. Moreover, Player S identified “Player A, B and C” at time t-2, t-1, and “Player A, B” at time t. At time t, Player S does not identify Player C.

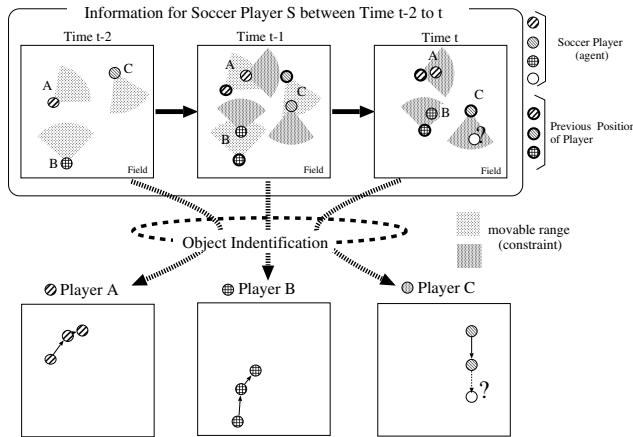


Fig. 1. Construction of Time Series Information

In information at time $t-2$, $t-1$, Player S makes time series information of each object used by two temporal dataset because all object data is completeness in these temporal dataset. In this circumstance, different objects' data in different temporal dataset are identified same object's data is called “Object Identification”.

It is problem that the object is incomplete information in identifying the each object. Fig. 1 shows “Player A, B” is identified already, so it was no different to do object identification at time t . However, one object was not identified. We expect that object is “Player C” used by other temporal data at time $t-2$, $t-1$, but our expectation cannot say certainty identification. When we implement spatial-temporal database, we think two methods of treatment of the object with lack information.

1. The database uses only the TimeMapCard of the object with complete information except one with incomplete.
2. The agent completes the lack information with the hypotheses, which are generated by identification mechanism, and it uses its data for a TimeMapCard. In this case, it is necessary to verify the correspondence[Kuni87].

Contradiction by adding the hypothesis is not caused in the former because it uses only complete information. However, spatial information, such as temporal data, is not always complete information. Moreover, there is a possibility to sometimes remove the majority of information when database excludes the object with lack information. On the other hand, the latter has the possibility to cause contradiction. However, its method has a possibility to expand the amount of information that can be usability. For the agent makes own action decision on real time, it is important point that the amount of information which can be used should be large. Therefore, we selected the latter in this paper.

2.5 Correspondence maintenance mechanism

The hypothesis might contradict at all time. Thus, When the agent used contradicted information which is assumed to be truth to understand the situation, there is possibility to do a wrong judgment for the agent. Therefore, it can be said that the maintenance of the correspondence is one of important function. This machanism has two main point as following:

- Whether the correspondence is maintained to complete information and the hypothesis in accumulated data set is verified by using constraint.
- We set the reliability of accumulated information. As a result, we set a constant restriction of no use of old information to the candidate hypothesis in the process of identification.

3 Team Composition: Strategy

Our soccer team “Spatial Timer” was implemented by Java language, and run on the jdk 1.2.2. Moreover, our team is composed with the following elements.

- **Goalie :** Defending Goal Player
- **DF :** Defensive Player
- **MF :** Mid Position Player
- **FW :** Offensive Player
- **Coach Client :** The agent can command all friend players on the field.

In our team, the soccer agent, which is actually arranged on the field. is four kinf of players above. As for each player, a suitable action decide algorithm for the role is applied. Moreover, the coach client is an agent who can give the instruction to the player on the field when the game is stopped temporarily form outside of the field. However, the coach client sent soccer agents on the field the entire situation of the game.

References

- [Ito99] N.Ito, K.Shinoda, K.Nakagawa, K. Iwata, X.Du and N.Ishii. Action Decision with Incomplete Information: A Temporal Database Approach, SMC'99 IEEE Vol.VI pp.722–727 1999.
- [SS98] I.Noda,etc. Soccerserver manual ver.4 rev.00 <http://www.robocup.org/> Technical report 1998
- [Nitta88] K.Nitta, Representation and Reasoning of Temporal Knowledge Journal of Japanese Society for Artificial Intelligence(in Japanese), 1988
- [Kuni87] S.Kunifugi Hypothesis-based Reasoning Journal of Japanese Society for Artificial Intelligence(in Japanese), 1987
- [Erwing98] M.Erwig,M.Schneider,R.H.Güting Temporal Objects for Spatio-Temporal Data Models and a Comparsion of Their Representation. LNCS1552 Advance in Database Technology

Polytech100

L. Stankevitch and S. Akhapkin

Computer Science Dept.

Saint-Petersburg State Technical University

Saint-Petersburg 195251, Russia

stank@phtf.stu.neva.ru asv_stu@yahoo.com

Abstract. Soccer makes a good example of the problem of the real world, which is moderately abstracted. This play has been chosen as one of standard problems for study on multi-agent systems. We are developing the soccer agent basing on the cognitive approach. In this year, we focused on synchronization problems and correct field representation into agent memory.

1 Introduction

The distributed artificial intelligence (DAI) and multi-agent systems (MAS) research directions became active at present. We see wide use of application MAS-technology for design DAI real-time control systems for robots grouped with the common work goal. The complexity of the design DAI real-time systems with MAS-technology has caused to using the machine learning (ML). The computation complexity is being replaced with system learning. The max reaction to situation can be gotten only taking dynamic changes into account.

Soccer (association football of robots) is team game in which players have a cooperation. This is a real-time game where situation changes dynamically. Soccer was chosen as one of problems for studying on multi-agent systems. We are designing the soccer agent using the cognitive approach. We present the decision algorithm for cooperative action among soccer players. We developed the learning modules for partial implementation of decision making at high and low behavior levels of soccer agent. This soccer agent can be used as a client of Soccer Server for participation in simulation league of RoboCup. The soccer team for participation in RoboCup was organized in Saint-Petersburg State Technical University in 1999.

2 Special team features

Our researches are focused on following agent features:

1. The agent is designed as a cognitive system that is the learning intelligence system with nervous system behavior, function, and structure. The knowledge acquisition is produced with learning in the work process. The knowledge is being kept and used in the associative neurological form.

2. The specific evaluation function used for a decision making. This function is used at high level control with a decision tree. It can be tuned during the learning process for adaptation to environment.
3. The middle level behavior function set is used for agent's individual tactics with operative change for adaptation to game situations.

We widely use this research in teaching students on course AI and intelligent control systems. Our students make experiments with different variants of control models on course practice work. Soccer Server platform is very useful for our targets, because it is interesting and easy understanding by students. We try to develop special learned modules based on neural and neural logic networks. In our opinion, these modules can advance tactic behavior of team.

3 World Model

World model is separated on two parts: field representation (or in other words snapshot of situation) and field objects modeling if can't get information about it. Currently the modeling part is simple: there are only tracking ball and simulating agent parameters (stamina). Each other player (teammate or opponent) is kept in agent memory as point on field with confidence parameter. In addition, it is used simple filter for deletion of "ghost" objects. "Ghost" is object that agent should find in current visual information, because it is stored in memory early, but agent can't do it. Also it is used communications between agents for get more rich information about of the world.

To handle asynchronous acting and sensing cycles agent has special synchronization module. Currently synchronization scheme based on sense-body information is implemented. After receive this information agent arms timer to get any pause before acting in hope to receive newest visual information. It is method allow to avoid most problems with cycle loses and cycles with two acting command.

4 Coach

In recent, online coach not used, but it is planned. In our opinion, online coach can help for solution of strategic tasks. We believe that using some statistical information, such as percent of successful passes, main directions of opponent attacks and other, coach can change tactic of team to solving these problems.

5 Communication

The multi-agent soccer system is client-server program environment. A team of agents must have cooperation and collaboration. Therefore the team agents have their own roles and formation. Let

$$A = \{A_1, \dots, A_m\}; R = \{R_1, \dots, R_n\} \quad (1)$$

where A_i – i -th agent; R_i – i -th role; F – formation is components U_j from roles R as

$$F = \{U_1, \dots, U_m\}; U_j \in R \quad (2)$$

where r_{ij} – role of j -th player in i -th formation. A map $[A, R]$ is flexible as it depends on time. The roles can be assigned to different homogeneous agents. Formation can affect the agent's external behaviors by specifying inter-role interaction. Since roles can be changed among formations, their specific interactions cannot be included in the role definitions. Instead of it the interactions are a part of the formation specification.

6 Skills

The behaviors (dribbling, intercepting, passing and etc) are realized as targets formed by tactical level. For that an appropriate low-level skills (running, avoiding obstacles, kicking to position and etc) and current local situation are used. For example, ball intercepting behavior uses only states ball and nearest obstacle. The goalie has some specific features. It could predict ball motion direction and intersect its. All individual skills of agent are implemented on hard algorithm with tuning parameters.

The correct evaluation of situation and position of the players is fulfilled with special evaluation function (stress function). This function evaluates the position where the player has ball with relation to teammates and opponents. The function for the i -th player is:

$$S_{ij} = F\{C_{ij}, W_{ij}, \sum_{q=0}^n [\beta_q \cdot P_i(q, M(t))]\} \quad (3)$$

where C_{ij} – interaction force coefficient of i -th and j -th players from coach (reserved); W_{ij} – adaptive coefficient for mapping of the game experience with similar opponent (reserved); β_q – prediction coefficient on period t ; $P_i(q, M(t))$ – position evaluation with position $M(t)$ teammates and opponents at moment q in future; F – constrained function. In situation that requires defensive behavior are used decision tree (DT). But practice showed that this method have problem, for example in some situation behavior of agent are began oscillate – switching before two behaviors and player begin to make many useless actions. So select behaviors method can be changed in next version of agent.

7 Strategy

The soccer agent has a multilevel behavior like football man does. Agent consists from following levels: tactical, behavior and skills. The tactical level selects current behavior based on global (about all players, markers and ball) information and strategic parameters that have initialization on start of game. In future it is planned use dynamical change of team strategy.

Some of strategy parameters are included in team configuration file. The following parameters are included in that file: home position for attack and defense, passing capability, activity zone of player and other. Also in configuration may present more than one strategy – we planed to switch strategy "on fly". So before game, human coach can prepare some strategy for team. We experimented with different strategies and found some useful set of it. For example, a defensive tactic against best team is more appropriate. The handles of setplays are simple and implemented through standard behaviors, but it is planned to make setplays behaviors with previously defined plans.

Team Leader: L. Stankevitch

Team Members:

Lev Stankevitch

- Saint-Petersburg State Technical University
- Russia
- Associated professor
- Remote participation

Sergey Akhapkin

- Saint-Petersburg State Technical University
- Russia
- Graduate student
- Remote participation

Sergey Vasiliev

- Saint-Petersburg State Technical University
- Russia
- Graduate student
- Remote participation

8 Conclusion

In result, it is designed the cognitive agent that has flexible behavior structure with acquisition of knowledge by learning. This agent was being used for participation in RoboCup_2000 and also will be planed participate in RoboCup_2001. In future, we plans: to finish fully autotuning synchronization module and correct field representation. It will be allowed us to attack the following problems: online coach (for dynamically change of strategy), using predefined combination in game, selecting optimal position (offense and defense) and opponent modeling. Also we plans to use some ML instruments (neuronets or neurologic) for replace any combinatoric algorithms.

Team YowAI-2000 Description

Takashi Suzuki, Sinnosuke Asahara, Hideaki Kurita and Ikuo Takeuchi

Department of Computer Science
The University of Electro-Communications
Chofu, Tokyo 182-8585, Japan
robocup@takopen.cs.uec.ac.jp

Abstract. Team YowAI-2000 is an improved version of YowAI-1999 which was originally developed by Takashi Suzuki. It won the RoboCup Japan Open 2000 Championship. However, it does only slightest dynamic cooperation among agents. For example, `say` and `hear` commands are not used. It was developed as an experiment to examine how far player's individual skill can go without cooperation.

1 Introduction

Team YowAI-2000 is an improved version of YowAI-1999 which was originally developed by Takashi Suzuki. The name YowAI may allude something relevant to AI, but it comes from a basic Japanese word *yowai* that means “weak” and it is an honored name that will be given to the strongest team among those developed in the Takeuchi laboratory of the University of Electro-Communications when a representative team needs to be selected for a championship competition.

The team was originally named Dango-Evolution I by Suzuki. As its name indicates, the team is an evolved version of Dango which was also developed by Takashi Suzuki.

We first sketch the team Dango and then the team Dango-Evolution I which was improved to the YowAI of year 2000.

2 Team Dango

Team Dango, whose name is also a Japanese word that means “a number of persons or things which gather closely to each other in rather a wide space”, has a good stamina management system, so that agents can run all the time without being apparently worn out. It also employs dynamic positioning and dynamic role change of the agents by some implicit relative positioning strategy and by simple negotiation using `say` and `hear` commands.

The Dango agents tend to be more or less “ball-centric” so that they seem to follow the ball in a mass within area of about 20 meter radius without taking care about global positioning balance. Seemingly relentless running of agents is, however, so cleverly controlled by the stamina management system that no one agent actually runs too much compared with others. The team could make a good game with some of Japanese top teams in early 1999. But it was easily defeated by a team that kicked and passed the ball widely across the pitch (soccer field).

3 Team Dango-Evolution I

Suzuki decided to develop a totally different kind of team based upon the team Dango, when he noticed that it was too early to develop a sort of dynamic cooperation among agents and it was much more important to promote the agent's individual ability, especially for constructing accurate world model in real time based upon the sensed information, for running, for kicking, for dribbling etc.

Suzuki's methodology was a little extreme on developing the Dango-Evolution I. He decided to abandon any dynamic cooperative action, thereby any usage of **say** or **hear** command. This controlled experiment principle would reveal the importance of individual agent ability before cooperative actions, he thought.

But it does not mean that Dango-Evolution I excludes all cooperation among agents. Its agent passes the ball if it thinks passing is more appropriate than dribbling in a given situation. Field player agents take fixed area positioning. For example, the left-side defender will not move out of his fixed area, left-side of its own half ground. There is no dynamic role change among the players. This fixed positioning allows an agent to kick the ball to a predefined direction if it cannot decide the best pass direction within the limited time, because his partner is probably there. Most of these cooperative actions can be considered static, that is, programmed as predefined actions.

On the contrary, each agent is highly tuned to make his own world model as precise as possible without communicating with each other. To achieve this accurate world modeling ability, Suzuki analyzed the soccerserver source program extensively. Fundamental idea is to retain sensed information with error estimation (Fig 1). Multiple sensing would reduce the error estimation quite efficiently. Basic skills of the agent are also thoroughly tuned to exploit this accurate world modeling.

Combined with the Dango excellent stamina management , these individual abilities of agents enhance the team power considerably. At 1999 year end, Dango Evolution I became to be able to play almost even games with CMUnited 1999, the 1999 World Champion team. It was honored with the name YowAI-2000.

Suzuki's successor Shinnosuke Asahara improved the YowAI further with respect to these basic skills and the following two tactical points. First, when a player runs with the ball near opponent field corner, it will make a centering kick toward a team mate who is presumably ready to make a shot directly when receiving the centering. This is the first of cooperative tactical moves incorporated into YowAI. Second, forward player's shooting range is enlarged in order to overcome excellent goal keeping skill some team developed up to that time. That is, offensive players became to make a shot at more distant position from the goal mouth. This simple improvement made YowAI be able to win CMUnited 1999 more often.

In fact, YowAI-2000 won the championship of the RoboCup Japan Open 2000 held at Hakodate (a north city of Japan) in June, 2000, defeating the defending champion 11-Monkeys-2 of the Keio University with the score 7-0 in the final game. It did not allow any goal by opponent teams throughout the competition.

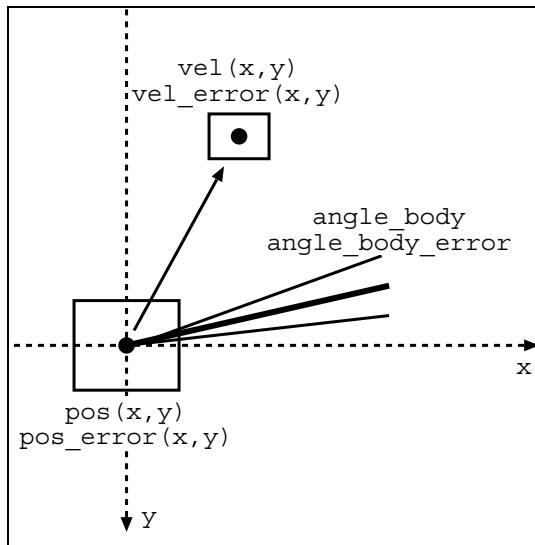


Fig. 1. The internal representation of sensed information

YowAI-2000 is written in C++ (about 10,000 lines). It should be emphasized that its CPU load is very light compared with other teams. On Pentium III 400MHz processor, it consumes only below 5 or 10 percent CPU time even when all eleven agents are run on a single CPU.

4 Soccerscope

Here, it is worth introducing the Soccerscope developed by Hideaki Kurita, which played a great assisting tool for Suzuki to develop YowAI-2000. Improvement of basic skills needs a good visualization tool to make sure what can be achieved or lost by a program modification. Without such a tool, the programmer can only guess the effect of his effort indirectly by conducting a number of exercise games. Especially, the accuracy of an agent's world model cannot be examined only by inspecting the soccermonitor display. Figure 2 shows the real position of players and a player's world model simultaneously with different colors, so that error in the world model can be easily examined. Kurita's Soccerscope is a visualization tool by which the programmer can examine what he wants. Soccerscope is written in Java and is upward compatible with the soccermonitor.

For the YowAI development, Suzuki and Kurita made a joint work so that Kurita implemented features one after another, which were requested by Suzuki in the course of his development. The Soccerscope communicates with the soccerserver with some program patches, so that it can display player's view range, stamina, world model, and even its intention if the user desires. Other features

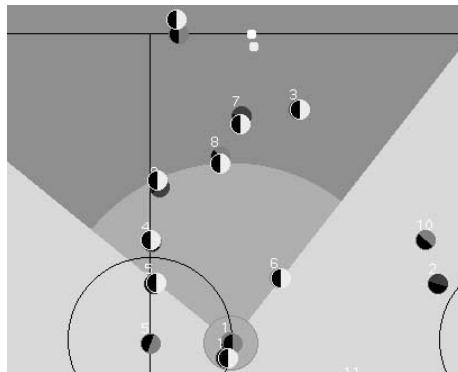


Fig. 2. Display of a world model overlapped with the real world

of the Soccerscope are slow motion, zoom-in display, reverse play, trajectory display etc. Zoom-in display helped the dribble skill improvement significantly.

It would not be possible to tune skills up to this level if there were not the Soccerscope.

5 Toward Next Step

YowAI-2000 could not be ranked as one of world top teams in the RoboCup 2000. The reason is obvious. It excludes almost all dynamic cooperation among agents because of a sort of controlled experiment principle. The next step of YowAI is surely the incorporation of cooperative actions based upon the achieved individual skills.

We are now porting the YowAI-2000 C++ code into Java in order to make the improvement easier. A lot of software components are extracted in this porting process. The lightness of the YowAI code is successfully inherited to the Java code.

However, the next YowAI (YowAI-2001) would not be a direct successor of YowAI-2000. Koji Nakayama is now developing a more strategically cooperative team, taking after some of excellent features of YowAI-2000.

The Soccerscope is also under improvement to make it able to display more about cooperative actions and intentions of agents.

11monkeysII

Naotaka TANAKA, Mio YAMAMOTO

Keio University

1 Introduction

Constructing a system that can cope with a dynamically changing environment is one of the greatest interests in the field of software engineering. And, biologically inspired systems, such as brain-nervous system, genetic system are already modeled as neural networks and genetic algorithms(GAs in short), and its usefulness is reported in various fields. However, despite of its advanced information processing system, immune system and endocrine system still has a long way to go before it is modeled like other ones. This may due to their complexity and randomness of the system. Nevertheless, the nature of the biological immune system, it is dedicated to self-preservation under hostile environment and enables the creature to maintain its life. Immune system is also capable of learning, memory, and pattern recognition.

2 World Model

Besides building in the biologically inspired system, we also made progress in building the world model. Though noises are included in the information received from the SoccerServer, 11Monkeys didn't have a strong strategy to cope with it. So, the clients happened to misunderstand its position. For this uncertain information, the team formation was ruined. In 11monkeysII, we made some changes on this point, changed the of calculation, and succeeded in building a better world model. This enables the strategic layer to give a better decision to the team.

3 Coach

Coach client is not used in the 11monkeys. We want to keep each client autonomous, let it determine its behavior from the information it has. That is the reason why we didn't use the coach client.

4 Communication

Once the first attacker is determined, and it seems better to switch its position with another player, they communicate with each other before they switch their position.

5 Skills

On developing our team, we used a 3D visualizer developed by H. Kera (Keio University). Unfortunately, this vizualizer doesn't work on-line though it enabled us to check the player's action. By using this tool, it made us easier to develop the client's personal skills.

6 Strategy

The 11monkeysII has a 3 layerd structure.The most fundamental layer is the individual layer.This is a layer formed by each clients.Each client has its own immune network to determine its action.This layer alone is powerful enough to play the game.And the second layer will be the tactical layer,which controls some set play.It enables the players to cooperate with their teammates more effectively.The last layer is the strategic layer which will control the game.

Each client will determine their behavior by its own immune system,i.e. by the function of individual layer. But, as soccer is a team sport, the game would be more exciting with some team plays. So, according to its situation, if the biologically inspired system judges that team play is more effective (for example it is better to pass the ball to a teammate than to dribble by itself), the strategic layer will determine the first attacker. And the tactical layer will enable the client to execute the team play. Once the first attacker is determined, and it seems better to switch its position with another player, the communicate with each other before they switch their position.

Team Leader: Naotaka TANAKA

Team Members:

Naotaka TANAKA

- Keio University
- Japan
- Master 1st grade
- did attend the competition

Mio YAMAMOTO

- Keio University
- Japan
- Master 1st grade
- did attend the competition

Web page <http://www.yy.ics.keio.ac.jp/~mio>

7 Conclusion

We have been working on the 11monkeys for few years, and because of the change in SoccerServer, our research theme etc. we are planning to rebuild our new team for 2001.

All Botz

Jacky Baltes

Centre for Imaging Technology and Robotics
University of Auckland
Auckland, New Zealand
j.baltes@auckland.ac.nz

1 Introduction

There are four features, which make the All Botz a very unique ROBOCUP team:

- The “robots” are cheap toy cars.
- The “robots” are non-holonomic (car-like).
- The robots do not use coloured markers, manual tagging, or bar codes.
- The videoserver can view the playing field from any angle instead of only from directly overhead.

Instead of custom built robots, the All Botz team consists of cheap remote controlled toy cars, which can be purchased at any toy store. The cars use coarse D-A converters to provide proportional steering and speed control. Cost is the main advantage of the All Botz; a standard team costs around \$10,000 USD, compared to the \$200 USD for the All Botz team.

The toy cars are non-holonomic. The biggest disadvantage is that the robots can not turn on the spot. This makes the path following and path planning problems more difficult.

Most teams use some additional markers on the robot to determine the orientation of the robot. Furthermore, all other teams use either distinct markers or manual tagging to identify the robots. The All Botz determine the orientation and identity of the robots using image processing and by correlating the movement commands with the observed behaviors of the robots.

Like most teams competing in the small sized league, the All Botz use a global vision system. However, it is not necessary that the camera is mounted directly overhead, but it can be mounted on any angle as long as the whole field can be viewed. A side view introduces large perspective distortions as well as occlusion problems.

The All Botz competed for the second time in the ROBOCUP competition. Our goal in 2000 was to qualify for the final and finish in the top eight of the competition. Unfortunately, we did not achieve our goal. The All Botz won their first game against Yale 1:0. We lost our next games against the RoboRoos and the Singapore Field Rangers 14:1 and 6:0 respectively and failed to qualify for the final.

2 Team Development

Team Leader: Jacky Baltes

Team Members:

Jacky Baltes	Benn Vosseteig	Daniel Shih-rong Kao
– Uni. of Auckland	– Uni. of Auckland	– Uni. of Auckland
– New Zealand	– New Zealand	– New Zealand
– Team Leader	– Student	– Student
– attended	– attended	– attended

Web page <http://www.citr.auckland.ac.nz/~jacky>

3 Robot Orientation and Identity

Most teams use colored markers, manual tagging and bar codes to determine the orientation of their robots. These solutions have severe disadvantages since they do not scale up to larger teams and to more flexible camera positions. For example, the bar code of the Roboroos is unusable in our vision setup, since important feature points will be occluded by the marker ball. Also, since many different colours need to be determined or small features need to be detected, the resulting calibration is unstable. For example, even small changes in the lighting conditions often require a re-calibration.

It is difficult to distinguish between more than six different colours reliably over the whole playing field. Other teams spend a lot of time calibrating their colours. Bar codes or other geometric properties of the shapes are hard to distinguish under perspective distortion.

Based on these observations and our experiences at ROBOCUP-99, we started work on the design of a new video server HORUS. The design goals for the new video server were to provide a scalable, robust, flexible solution to the orientation and identification problem.

If we do not want to use additional patterns, then what else is there? The only information left is the image of the robot itself. So the goal was to design a video server that uses only a single marker ball and no other patterns on the robot.

Figure 1 contains three zoomed views of our robots from our video camera. The views correspond to the worst case (i.e., the robot is at the far end of the playing field). As can be seen, the most prominent features are the edges along the top of the robot. Other features (e.g., the wheels are not always visible and hard to distinguish). Therefore, we decided to use these edges as features and to infer the orientation of the robot from them.

However, it is *impossible* to determine the orientation of the robot from a single image. Given the angle of the edge, there are four possible orientations for the robot, which can not be distinguished without further information.

Furthermore, since all robots have exactly the same shape, it is impossible to identify the robot from a single image. Therefore, we use additional information



Fig. 1. Some sample images of our robots taken at the far side of the field.

(e.g., history of the cars, current commands, motion of the robot) available to the video server to disambiguate the orientation and to identify the robot.

Given the real world coordinates of the robot, the surrounding image corresponding to a square area of the diameter of the robot is extracted. The maximum size of this window depends on the geometry of the camera position. In most “practical” situations, the size of the window is less than $64 * 64$ pixels. All further processing is limited to this local neighborhood. The image is divided into four regions, which are shown in the Fig. 2.

- Pixels that are more than half a diameter away from the position. These can not be part of the robot and are ignored.
- Pixels that belong to the marker ball or are very close to it. These pixels are usually noisy and are ignored.
- Pixels that match the top colour of the robot.
- Pixels that belong to the contour of the robot. These pixels are determined after tracing the contour of the robot using a standard edge walking algorithm.

Figure 2 shows the output for the three sample images given in Fig. 1. The contour of the robot is shown in black. As can be seen, using even a very coarse colour calibration, the edges of the robot can be traced accurately.

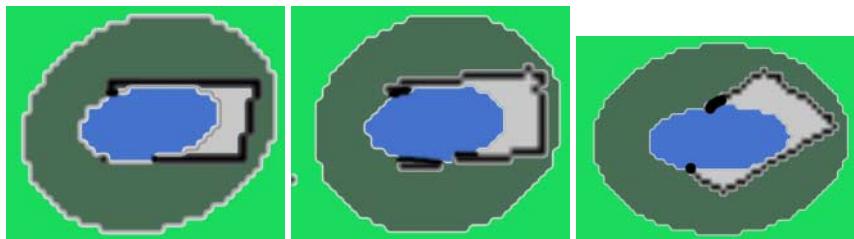


Fig. 2. The image of the robot after preprocessing. Pixels that are too far or too close are ignored. Pixels matching the colour of the top of the robot and pixels on the contour.

Given the position of the edge pixels, a general Hough transform is used to compute the possible orientation of the robot in the first quadrant. The Hough transform is a popular method in computer vision to find lines and other shapes. The basic idea for the Hough transform is to collect evidence to support different hypothesis. The evidence for different hypotheses is accumulated and the hypothesis with the strongest support is returned as the solution.

4 Identification of Robots

As mentioned previously, since all robots in our team look identical, the vision information is insufficient to identify them. HORUS uses two additional sources of information to determine the identity.

Firstly, HORUS predicts the motion of the robot and tracks it. Should the robot be found in the predicted position, its identity and its associated probability is not changed. If the robot is found in the neighborhood of the predicted position, its identity and confidence value are not changed.

Secondly, HORUS observes the motion of the robot over a number of frames and assigns it one of seven states: not moving, forward left, forward straight, forward right, backwards left, backwards straight, and backwards right. The actual steering angle is not determined, so there is no difference between, for example, full left and gently left.

HORUS listens to the communication between the clients and the transmitters. When a transmitter receives a command (e.g., Robot 3 forward left!), HORUS stores the last command sent to the robot and observes the motion of the robots. Should the motion of the robot agree with the command, the probability of the identity is slightly increased (a factor of 1.1). On the other hand, there are many reasons why a robot does not follow the command: malfunction, an unknown obstacle in the path, noise in video processing, the robot is being pushed by another robot, occlusion, etc. In this case, HORUS slowly decreases the probability of the identity assignment by a factor of 0.9.

5 Conclusion

This year, the development focused on extending the video server so that it does not require any coloured markers, manual tagging, or bar codes to determine the orientation and the identity of our and our opponents robots. The All Botz are the first team that correlates movement commands with the observed behavior of robots to identify them.

Although we had some problems, the approach proved possible and we believe the remaining issues in the robot identification can be solved before next year.

There is much work left to be done with the agent architecture. More goals and plans will be added to the individual agents and a richer communication language between the robots and the videoserver will be developed.

4 Stooges

Jacky Baltes

Centre for Imaging Technology and Robotics
University of Auckland
Auckland, New Zealand
j.baltes@auckland.ac.nz

1 Introduction

The 4 STOOGES are a small sized ROBOCUP team, which grew out of the ALL BOTZ who competed at ROBOCUP-99.

One of the main differences is that the 4 STOOGES are a team of fully autonomous robots. All sensors, actuators, and power supply (CMOS camera, and battery) as well as all processing is housed on the robot.

The localization, path planning, and task planning problems for local vision teams are more difficult than those for global vision robots. However, an autonomous robot is more realistic for applications outside of the ROBOCUP domain. Therefore, the development of robust solutions to the above mentioned problems is very important to drive the development of practical mobile robotics.

This was the first year that the 4 STOOGES entered the ROBOCUP competition. Most of the development was on the design of a mobile robotics platform and the image processing of the robots. The main contribution was the development of a new image capture routine which allows significantly higher framerates (30 fps as opposed to 5 fps) than the original one.

2 Team Development

Team Leader: Jacky Baltes

Team Members:

Jacky Baltes	Gary Henson
– Uni. of Auckland	– Uni. of Auckland
– New Zealand	– New Zealand
– Team Leader	– Student
– attended	– attended

Andrew Thomson	Weidong Xu
– Uni. of Auckland	– Uni. of Auckland
– New Zealand	– New Zealand
– Student	– Technician
– attended	– could not attended

Web page <http://www.citr.auckland.ac.nz/~jacky>

3 Hardware Platform

The 4 STOOGES use last year's mechanical platform of the ALL BOTZ, the old but reliable Tamtech toy cars. However, the 4 STOOGES added local control and local vision resulting in a fully autonomous robotic system.

The cars are controlled by an Eyebot controller. The Eyebot controller is a MC68332 processor based controller, developed by Prof. Thomas Bräunl from the University of Western Australia.

The Eyebot controller includes a parallel interfaces, a serial interface, A/D converters and 12 servo control outputs and two motor control circuits. It also provides an interface for the Radiometrix serial transceiver. The biggest advantage for small robotics projects is that a CMOS camera is available for the controller. The Eyebot controller also comes with some libraries for image capture and processing.

The CMOS camera provides a 80x60 image with 24 bit colour information. The colour information is not very good, since the camera uses a cheap Bayer RGB pattern. The image sensor has 160 rows. The odd rows provide red and green and the even rows return green and blue colour information.

Since our mobile platforms can not turn on the spot and since we do not have dead reckoning sensors in our robots, we decided to add a pan servo on the robot. Using this servo, the field of view of the robot is extended in the horizontal direction. Figure 1 is an image one of the 4 STOOGES.

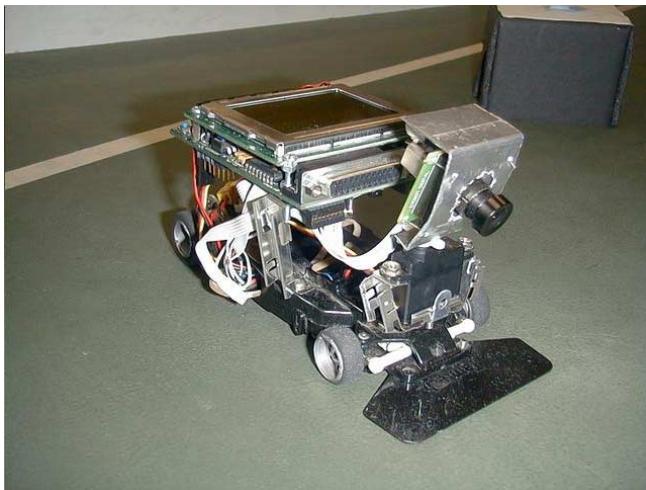


Fig. 1. An image of one of the robots of the 4 STOOGES. The Eyebot controller is mounted on the top. The pan servo and the camera are mounted on the front of the robot.

4 World Model

Localization of the robots is a difficult problem for the local vision teams in the ROBOCUP small sized league. Other teams using local vision added special corner or side markers to make localization easier. Since the small sized league is intended mainly for global vision teams, it does not make any such concessions.

The only two distinguishing features in the domain are the two goals on either side of the playing field. The problem is that the 4 STOOGES are not able to see the goals in most positions.

Since the 4 STOOGES do not have shaft encoders, they can not use dead reckoning for the periods where the robots are not able to observe the goals. Instead, the 4 STOOGES use the walls to recalibrate their relative short term motion. Note that by observing the wall, only the orientation angle can be accurately determined.

The 4 STOOGES recalibrate their orientation whenever they are able to observe a goal. The orientation is updated using the relative change of orientation from walls and lines on the field. If no feature can be detected by the robot, a “best guess” of the motion of the robot is used.

5 Skills

The code for the field players is almost identical. The only difference is their home position. The team strategy employed by the 4 STOOGES uses one defense player, a left striker, and a right striker.

The goal keeper starts out on the goal line and attempts to stay on this line as much as possible. It uses a simple feed forward control to keep the goalie between the ball and the goal. If it can not see the ball, it moves back towards the home position, since this provides it with the best view of the playing field.

The strikers always approach the ball if they can see it. If a striker can see the goal and the ball, it will attempt to shoot at the goal (if the direct line through the ball points at the goal) or will try and maneuver itself behind the ball. If the robot can not see the goal, but has recently recalibrated its orientation, it will use this information to either kick the ball into the opponent’s direction (if it is facing the opponent’s goal) or to dribble the ball (if it is facing its own goal). In all cases, it attempts to keep the ball away from the opposition.

6 Special Team Features

The original Eyebot controller was limited to five frames per second, because of the large interrupt latency for each byte transferred.

The 4 STOOGES changed the camera interface routine to use a carefully timed polling and were able to increase the frame rate to 30 frames per second. Since the source code for the Eyebot BIOS is not freely available, we had to disassemble the BIOS and “hack” the BIOS to support the higher frame rate.

Another features of the 4 STOOGES was that a lot of the image processing code was based on the video server of the ALL BOTZ, a global vision team also from the University of Auckland.

7 Conclusion

This year was the first year that the 4 STOOGES entered the competition. Clearly, a local vision team is solving a more challenging problem than a global vision team. It comes therefore as no surprise that the 4 STOOGES were not competitive against their global vision opponents.

Of more interest to the 4 STOOGES were the results of the local vision derby which was held for the first time this year. The games between the local vision teams were boring. Most of the time, the robots had trouble finding the ball. In case of the 4 STOOGES, this was due to the poor camera mounting as well as the poor quality images returned from the CMOS camera.

One lesson that we learned from the competition was the camera mounting was not useful. To recognize the ball even at comparatively close distances (about 50cm), the camera had to be mounted at such an angle that balls that were really close to the robot were not recognized anymore.

We changed the mounting of the camera to look straight ahead. This will allow us to see the ball when it is close to the robot.

One of the main research directions of this work is the development of efficient role assignment for multiple agent systems. Therefore, we will add communication between robots for next year's competition.

CIIPS Glory

Small Soccer Robots with Local Image Processing

Thomas Bräunl, Petter Reinholdtsen, Stephen Humble

Centre of Intelligent Information Processing Systems (CIIPS)

Dept. of Electrical and Electronic Engineering

The University of Western Australia, Nedlands, Perth, WA 6907

Abstract. CIIPS Glory is using a local intelligence approach to solve the task of robot soccer. In this respect our system is much closer to the F-2000 league (mid-size) than other approaches in the F180 (small-size) league. We are using *EyeBot* controllers, the *RoBIOS* operating system and the *EyeCam* digital on-board camera systems (CMOS). Our robot family also comprises 6-legged and biped walking machines, an omni-directional robot and an autonomous airplane.

1 Introduction

The CIIPS Glory robot soccer team (Figure 1) has competed in RoboCup tournaments in Singapore (1998, [1]), Melbourne (1999), and Melbourne (2000). In the first two events it was the only team using a local intelligence approach without employing global vision, while at Melbourne 2000 three teams used this approach. Since the basic robot design has already been published in [2], we will concentrate in this article on innovations and improvements of our design.

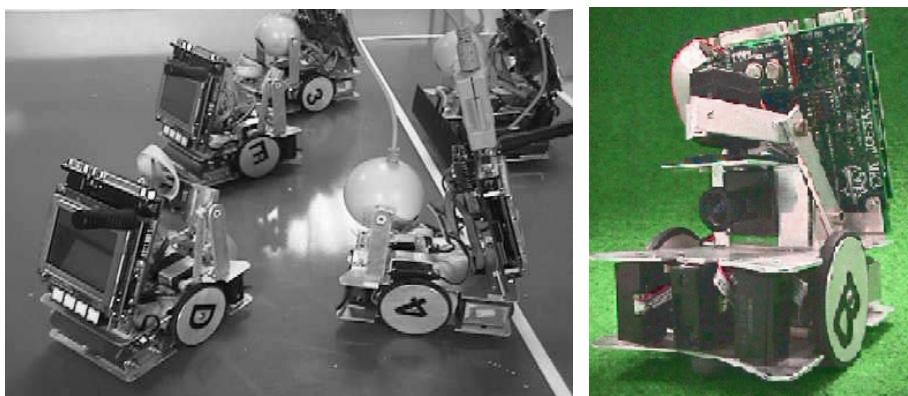


Fig 1. *EyeBot* soccer robots with local intelligence

We experienced major difficulties in the past robot soccer competitions, which lead to a number of extensions and modifications of our existing robot design.

Problems

- Frame-rate too slow
- Poor self-localization
- Poor ball-tracking behaviour
- Limited robot interaction

Solutions

- Integrate FIFO buffer in camera interface
- Integrate compass sensor
- Change from tilting to panning camera
- Add optional wireless network

2 Team Members

A number of people contributed to the CIIPS Glory 2000 team, of which Petter Reinholdtsen, Mari Wang, and Stephen Humble went to the competition in Melbourne. Petter Reinholdtsen, team leader and volunteer from Norway, spent a year at UWA to develop the on-board software. Mari Wang, student at UWA, developed the goal-keeper software. Stephen Humble, volunteer from Australia, worked on both mechanics and software. Philippe Leclercq, Ph.D. student at UWA, developed the image processing library. Mark Gaynor, volunteer from Australia, helped with the radio communication and electronics. Klaus Schmitt from Univ. Kaiserslautern developed operating system routines and sensor drivers. Thomas Bräunl worked on operating system routines and is director of the Mobile Robot Lab at UWA.

3 Hardware Platform

Each of our robots comprises a microcontroller system (*EyeBot*) interfaced to a digital color camera, distance sensors, shaft encoders, compass, DC motors, servos, wireless module, and a graphics display. All image processing is done on-board. We are interested in research on autonomous mobile systems, so we took this clearly disadvantaged robot soccer approach instead of using a global overhead camera.

Since we rely on local image processing on a microcontroller, we had to make concession about image resolution (80x60 pixels in 32 bit color). On the other hand, we could not afford a too low frame rate, or we would see a passing ball only once in a frame sequence and could not tell which direction it went. Directly interfacing a digital CMOS camera to the CPU allows to do image processing at a rate of 3.5 frames per second (fps). Only after the competition we finished integrating a hardware buffer, which significantly reduces the number of time consuming interrupts required for data acquisition. This now gives us a framerate of about 15 fps.

Self-localization is an important task for our robots, since we do not use a global positioning system like an overhead camera. We rely on dead reckoning from a specified starting position and orientation. However, a robot will soon lose its exact position and orientation due to wheel slippage or - much worse - collision with another robot. Therefore, we integrated a digital compass to our robots. In the robot soccer application orientation is more important than position, because it guarantees that a robot is heading for the right goal. The robot position can be updated by its local infrared sensors whenever it gets close to one of the side walls or a corner.

The camera mechanics was changed from tilting to panning in order to improve ball tracking. The camera can be moved sideways at a higher speed than the whole robot can turn, so this will allow us tracking of balls moving faster across a robot's field of view than it would be possible with a static camera mount.

An implementation-independent soccer protocol allows us to test different implementations of the robot control programs side-by-side.

4 Wireless Communication

There are several reasons for a communication network in a multi-robot scenario:

- a. To allow robots to communicate with each other
- b. To remote-control one or several robots
- c. To monitor robot sensor data
- d. To run a robot with off-line processing (combine previous points)
- e. To create a multi-robot monitoring console

Our approach accomplishes all these requirements by using a "virtual token ring" structure [6]. At the initialization of the network, one robot takes on the task as "ring master" and determines which other robots want to participate in the communication system. Each robot (and each base station) is identified by a unique ID number, which is being used for addressing a robot. A specific ID number is used for broadcasts, which follows exactly the same communication pattern. In the same way, subgroups of nodes can be implemented, so a group of robots receives a particular message.

The "virtual token ring" system enables a number of robots to exchange data with one another over the same frequency by taking turns in data transmission - only the robot which holds the token is allowed to transmit. Transmitted data includes ball detection data (current ball position and velocity vector), local robot position and orientation, and plan details to be shared with other robots.

5 Self-Localization

The newly developed self-localization method combines short range distance sensor information with dead reckoning from the wheel encoders. It is based on ideas from [3], [4] and [5] with additional consideration for range sensors with limited accuracy and reach. The robot collects distance information from the sensors synchronized with the camera frame rate at 3.7 times per second. The sensor information is stored in a list with a time stamp and the position given by read reckoning. This list is then processed to find lines which are matched against the soccer field map to estimate the current position (see Figure 2).

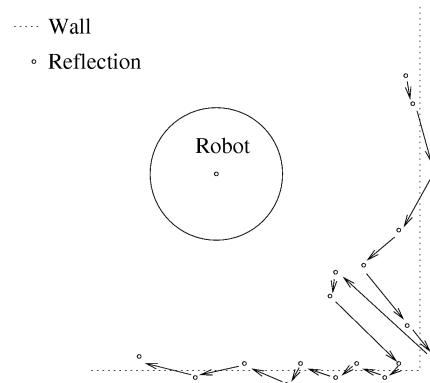


Fig 2. Robot self-localization

Processing is done as follows: First, points are sorted based on their angle from a well chosen origin. The origin should be inside the soccer field and have no obstacles (or as few as possible) between itself and the walls, e.g. the current robot position. The angles between the points are then used in sorted order to calculate an angle histogram. This groups points into histogram classes. Points are then sorted depending on the number of members in the histogram class and the angle from the origin. This way, points located on the same wall should be grouped together. The points are then traced to find lines, starting with the first point in a group. The next point is added to the line if the angle between the two points is in the same histogram group as the current point. The end result is a list of lines which are matched with the map of the soccer field. To avoid lines which are not part of a wall, line segments shorter than 20 cm are ignored.

This method works quite well, but requires some time to process the list. A robot blocks for about 10 seconds every time the list of points is processed. Clearly, this makes this self-localization method of limited value in a real-time competition environment. Further problems arose from uncalibrated sensors, inaccurate sensor readings, sensor reflections and incorrect readings from angled walls.

Acknowledgments

The authors would like to acknowledge the work of all students who participated in the previous CIIPS robot soccer campaigns: Birgit Graf for the first soccer team 1998, who also proof-read this paper, Andrew Barry and Ilario Dichiera for the soccer team 1999. More information is available on the Internet, where also the source code of the soccer program and a robot simulator are available:

<http://www.ee.uwa.edu.au/~braunl/eyebot/>

References

1. Th. Bräunl, B. Graf, *Robot Soccer with Local Vision*, Pacific Rim International Conference on Artificial Intelligence, Singapore, Nov. 1998, pp. 14–23 (10)
2. Th. Bräunl, B. Graf, *Small Robot Agents with On-Board Vision and Local Intelligence*, Advanced Robotics, vol. 14, no. 1, 2000, pp. 51–64 (14)
3. K. Graves, W. Adams, and A. Schultz, *Continuous localization in changing environments*, in: Proceedings of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, pages 28–3, 1997.
4. J.-S. Gutmann, T. Weigel and B. Nebel, *Fast, Accurate, and Robust Self-Localization in Polygonal Environments*, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99), Kyongju, Korea, 1999
5. A. Schultz, W. Adams, and B. Yamauchi, *Integrating Exploration, Localization, Navigation and Planning With a Common Representation*, Autonomous Robots, 6 (3), pp. 293–308, 1999
6. P. Wilke, Th. Bräunl, *Flexible Wireless Communication Network for Mobile Robot Agents*, Industrial Robot International Journal, to appear, 2000

ViperRoos 2000

Mark M. Chang, Brett Browning¹, Gordon F. Wyeth

Department of Computer Science and Electrical Engineering

University of Queensland, QLD 4072, Australia

<http://www.elec.uq.edu.au/~chang/viperroos>

chang@csee.uq.edu.au, brettb@cs.cmu.edu, wyeth@csee.uq.edu.au

1 Introduction

The ViperRoos are a team of soccer playing robots that made their debut in the F-180 league at RoboCup-2000. Each Viper robot is completely autonomous and relies on on-board vision, rather than an overhead camera, as its primary means of perceiving the world. In addition to participating in robot soccer competitions, the Viper robots have already been used successfully for other research investigations (for example [1]). The ViperRoos represent an innovative step towards cheap, autonomous robot hardware for use in both RoboCup competitions and general mobile robotics research.

This paper describes the technical aspects of the ViperRoos system, with a focus on the robot hardware and software. Performance-wise, the robots acquitted themselves well against teams with overhead cameras where the Viper robots are at an obvious disadvantage. The robots finished the round robin with one win and two losses. Figure 1 shows the ViperRoos team for RoboCup-2000.

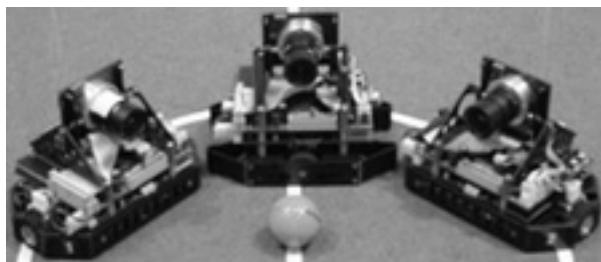


Fig. 1. The ViperRoos team: 2 Field Robots and Goalie (center)

2 Hardware Architecture

The VIPER robot platform extends from the reliable and proven base of the UQ RoboRoos [2] through the addition of on-board camera, dedicated vision processing hardware, and half-duplex radio communications. Figure 2 shows a schematic of the VIPER robot's hardware components.

¹ Now a postdoctoral fellow at Carnegie Mellon University

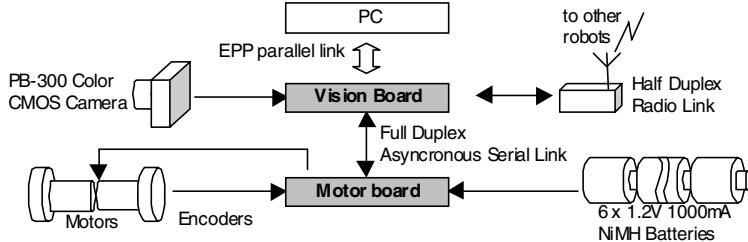


Fig. 2. Block diagram of Viper robot's major hardware components

Mechanically speaking, the VIPER is almost identical to its RoboRoos ancestor [2]. The robot uses the same aluminum chassis, which houses two coreless DC motors in a differential drive configuration. The chassis also houses two packs of 3 NiMH battery cells, which provides power to the entire system. Feedback from each motor is derived through externally mounted HEDS-9100 shaft encoders.

Electronically, the VIPER is built around two loosely coupled, custom-built processor boards dedicated in both a hardware and software sense to different parts of the robot intelligence problem. The motor controller board is built around a Motorola MC68332 MCU running at 20MHz and represents a second generation of the RoboRoos processor board. The MC68332 processor offers some computational power but primarily provides dedicated motor controller functions through its Timer Processor Unit (TPU). The motor board contains the ancillary power electronics for the MC68332, 512K of Flash memory, 256K of SRAM and the MOSFET H-Bridges for driving the DC motors. The vision board is built around a Hitachi SH7709 processor, part of the SH-3 series MCU's. The SH7709 provides reasonable computational power (up to 80MIPs) with a full set of peripheral units. The vision board communicates with the motor controller board via a 60kbs serial link utilizing the serial communication peripherals. The memory interface peripheral on the SH7709 enables glueless interfacing to a wide variety of memory including 16MB of 100MHz SDRAM for general program and data, up to 1MB of Flash, and 512K of SRAM for image capture. The Direct Memory Access (DMA) controller provides a simple digital interface to the Photobit PB-300 color CMOS image sensor housed on a separate custom-built board. The SH7709 captures and stores image data into SRAM via DMA whilst the processor computes the previously captured frame.

The Viper can communicate to the outside world via a half-duplex FM radio link, running at 19,200bps using the remaining free SCI port on the SH7709 processor and a BiM module from Radiometrix. The robot also has a high-bandwidth debugging interface via an Extended Parallel Port (EPP) connection to a PC. This high-bandwidth channel, which enables real-time video debugging, requires a cable connection and is not used while the robot is in motion.

3 Vision

On-board vision is the primary means of perception for the Viper robots. The performance of the visual perception system has direct impact on the effectiveness of

the robot as a whole. Thus, most of the development effort for the ViperRoos in their debut year focused on evolving robust and efficient vision routines.

The vision processor receives frames at a frame rate of around 10Hz from the PB-300 CMOS image sensor. Each image is 512x128 color pixels arranged in a Bayer 2G format with two green, one red and one blue pixel for every 2x2 pixels. With a 2.8mm F1.4 CS-mount lens, the robot is able to view objects from 15cm to a field length away with horizontal viewing angle of 79 degrees.

The first stage of vision processing sub-samples and transforms the color space of the image from RGB to YUV to produce a 128x32 YUV image. The UV components of the filtered image are segmented into pixel types using predefined lookup tables. The UV segmentation table is calibrated before game time using custom written PC software to display the UV content of real time, raw image input graphically. Color segmentation in the UV space is, within our experiences, generally more robust to lighting variations than a similar RGB classification scheme. The segmented image is then filtered using opening operation of binary mathematical morphology [4] on each pixel type to remove noise artifacts. Once the pixels have been colour-segmented, a straightforward region growing algorithm, using the classified pixel types as seeds, is applied. The resulting regions, provided they meet certain criteria such as size, are interpreted to produce estimates of the relative location of obstacles and the ball. For game planning purposes, the egocentric bearings to objects in the image are translated into a Cartesian coordinate frame centred on the field itself.

For obstacle avoidance, a different mechanism is used. In short, any two adjacent pixels that are not field green are interpreted as obstacles to be avoided. The vision system produces what can be interpreted as a local, egocentric *proximity* map of obstacles in the environment. This information is then used by the navigation system to navigate a safe path to the desired location.

4 Intelligence Schemas

The ViperRoos are a behavior-based system where the active behaviors depend on the visual perception of ball. The current implementation is quite similar to the RoboCup 2000 RoboRoos control system [2] and is shown in figure 3. Since the field and goalie robots have different roles in the game, they exhibit different behaviors within the team. Each field robot wanders in its pre-designated search area until it sights the ball. When the robot sights the ball the kicking behavior that aligns the robot with the ball and kicks it, becomes active. In all other situations the robot reverts to wandering. The field robots navigate using a reactive, biologically plausible navigation schema that is an extension of the RoboRoos approach. For brevity, the system will not be discussed here and the interested readers should consult [1]. The goalie uses two behaviors: blocking and waiting. When the goalie sights the ball, the robot moves sideways to attempt to intercept the ball before it reaches the goal. When waiting the goalie returns to the center of the goal mouth and attempts to realign its position in preparation for the next block. When the robots are not actively kicking or blocking, they perform localization where they estimate their position using the goals and walls as landmarks. The current implementation uses a simple mechanism where the path integrated, or dead-reckoned, coordinates are updated whenever they appear to be substantially in error.

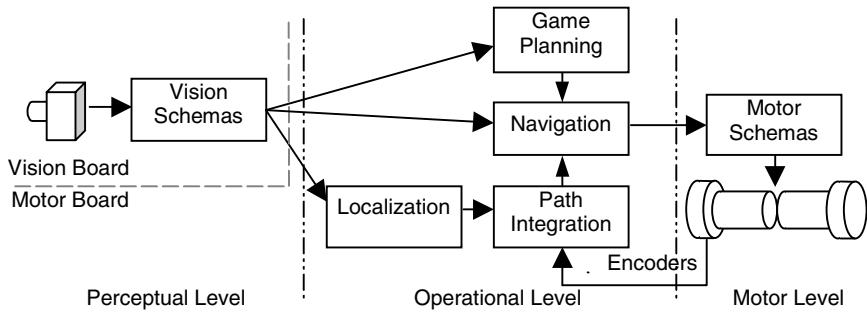


Fig. 3. The major components of the ViperRoos software control system.

5 Conclusions and Future Work

In summary, this paper has described the debut implementation of the ViperRoos on-board vision robot soccer team. Despite the disadvantages of using local vision against global vision, we believe that the overall performance of the ViperRoos at their debut RoboCup competition was satisfactory and that their strong performances against other local-vision teams demonstrated the potential of the system for further development.

There are a number of issues that we will be addressing in the coming year. In particular, we wish to develop:

1. Hardware upgrade to improve the frame rate and quality of visual system.
2. Inter-robot communication strategies that enable the team to both cooperate and overcome their perceptual and actuation limitations. Redesign of the communication hardware to improve latency and bandwidth.
3. Special kicking hardware to improve the accuracy and effectiveness of kicking thereby making the team more competitive.

References

1. Browning, B.: Biologically Plausible Spatial Navigation for a Mobile Robot. PhD Thesis, Department of Computer Science & Electrical Engineering, University of Queensland, Australia, (2000)
2. Wyeth, G. F., Tews, A.: UQ RoboRoos: Kicking on to 2000. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors: RoboCup-2000: Robot Soccer World Cup IV. LNAI, Springer Verlag, (2001), in this volume
3. Jonker, P., Caarls, J., Bokhove, W.: Fast and Accurate Robot Vision for Vision based Motion. In Peter Stone, Tucker Balch, and Gerhard Kraetzschmar, editors: RoboCup-2000: Robot Soccer World Cup IV. LNAI, Springer Verlag, (2001), in this volume
4. Parker, J. R.: Algorithms for Image Processing and Computer Vision. John Wiley & Sons, New York (1997) 68-102.

RoboCup 2000 (F180) Team Description : UPMC-CFA team (France)

Jerome Douret, Thierry Dorval, Ryad Benosman, Francis Bras, Gael Surtet,
Thomas Petit, Nadege Quedec, Denis Philip, Gilles Cordurié, Mario Rebello,
Didier Abraham, Nicolas Couder, and Marco Marcon

Université Pierre et Marie Curie
4 place jussieu
75252 Paris cedex 05, boite 164
France

rbo@lis.jussieu.fr, Gilles.Cordurie@cicrp.jussieu.fr
http://www.robo.jussieu.fr/rbo/Robocup/UPMC-CFA_robocup_team.htm

Abstract. This paper describes the team UPMC-CFA that participated to the EuroCup'2000 and RoboCup'2000 F180-league competition. We will start by presenting the mechanical and electrical design of our robots. We will then introduce the vision program explaining the colometric and geometric calibrations it needs. Finally we will explain the implemented behaviour and the way we controled our robots.

1 Introduction

UPMC-CFA is a team composed of student coming from two different sections of UPMC. The team is composed of six electrical engineering students belonging to the engineering school IFITEP and four computer vision students belonging to the DESS IE. The team has three advisors that have been involved in all Robocup's championships since Nagoya in 1997. The UPMC-CFA 2000 team is a new team that was created in september 1999. Our main motivation is to use Robocup as a test platform for new approaches in computer vision.

2 Mechanical Design

Our robots are designed to fit the F180 size regulation. They are built out of aluminum frames that protect their inner parts. The robots have a differential drive with two active wheels that are not in the middle to allow a more complicated and unpredictable trajectories and a better twisting to shoot the ball. The maximum speed is about 1.2 m/s. The robots where supposed to carry kickers, but due to a lack of time they were not used.

3 Electrical Design

The microcontroller used is an Atmel 89S53 that has the advantage of being programmable in situ with 12kb flash memory running at 24 Mhz. For radio



Fig. 1. Inside UPMC-CFA robot

communication we used the classical Radiometrix working in the band 433Mhz and 418Mhz. The robots are powered by two Maxon motors with an integrated 19:1 gear, we used a dual H bridge motor driver L6202. The batteries were specially assembled to fit in the robots and are from GP Batteries our sponsor. Finally a Altera 7032 generates the pulse with modulation for the motors. All the described devices were mounted on a single custom card (see Fig.1) including the stabilized power supply.

4 Computer Vision

4.1 Hardware Description

We used a PCI-framegrabber and a progressive CCD camera JAI M70 that were given to us by our sponsor IMASYS. The M70 has the advantage of needing only one cable for data and power directly connected to the Pc. We capture RGB images of size 640*480 at a rate of 30 fps. We used the GIPS library developped by IMASYS running under Windows NT to control the camera.

4.2 Color Calibration and Regions Labelling

The output signal of the camera is RGB. A color is defined as a three component vector containing the values for the red, green and blue color planes. A first stage is manually made where colors are selected, for each color we compute a mean and a standard deviation. The image is then filered, an image containing only the desired colors is generated. In a second time a second detection based on HSV color space is applied only on the detected zones to reduce computation. A scalar is assigned to each color corresponding to the objects seen, we obtain

then a 8 bits images where for example the field is coded zero. A fast labelling algorithm is then applied giving for each detected region its gravity center and the number of pixels it contains. For more information concerning the labelling method, the reader can reffer to [1].

4.3 Recognizing robots



Fig. 2. Caraterizing the robots and their orientations

The coding of the robots is very close to bar code widely used. Each robot has a number of thick and thin bars (see Fig.2). The bars are chosen so that we can have the information on orientation and at the same time allowing to retrive the robot's position with a very high accuracy. We start by finding the color of our team represented by the color of the ping pong ball, then we look in a circle around it the number and type of bars. Once this operation done we reffer to a lookup table and according to the number of thick and thin bars we can deduce the robots' number.

4.4 Geometric Calibration

The geometric calibration allows us to locate our robot on the table. The geometric calibration retreives from image coordinates the position of the robots according to a coordinates system on the table, taking into account the height of the observed object. It is based on projective geometry and more specifically colineations [2]. A two heights calibration pattern is needed shown by Fig.3. Once the calibration is done we need to enter the height of any object seen by the camera to obtain its right position on the table. The precision obtained is less than 2mm, and relies on the fact that the image coordinates are subpixel.

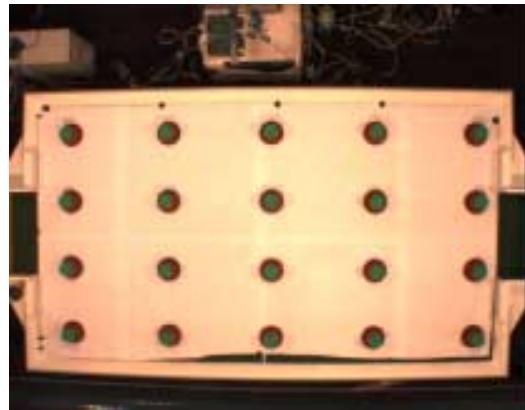


Fig. 3. Geometric Calibration Pattern

4.5 Behaviour

The behaviour of the robots is motivated by different flags corresponding to specific situations and robots positions. The field is cut into different zones each having a priority. The knowledge and the analysis of these flags gives an analysis of the situation of the game. Each robot according to its position, the position of other robots and the ball will have a specific behaviour. The behaviour is based on micro and macro orders. Each robot has simple behaviours each one corresponding to an action. The robots use a prediction algorithm that gives the position of the ball and the opponents in the next two images. The behaviour of the robot varies according to situations for example if the team is winning or not.

5 Conclusion

We find robocup a very good tool for pedagogy and research. Participating to Robocup made us develop new camera calibration techniques specially aimed to the F180 league that might be used in other robotics fields. Our main goal for the next competition is to participate in the middle size league to develop new vision techniques for mobile robots.

References

1. Lionel Lacassagne, Maurice Milgram : Motion detection labelling data association on tracking in real time on risc computer ICIAP'99, pp 154:162, September, Venice, Italy
2. J.G Semple and G.T Kneebone Algebraic Projective Geometry, Oxford University Press 1952

MuCows

Andrew Howard

University of Melbourne

1 Introduction

The University of Melbourne entered the RoboCup contest for the first time in 2000, fielding teams in both the F180 and Sony Legged Leagues. This paper describes certain aspects of our F180 team: the MuCows.

The MuCows played three (official) matches in the course of RoboCup2000. The scores were as follows: won 14:0 against the 4 Stooges (University of Auckland, NZ); lost 0:7 against Big Red (Cornell, USA); lost 0:1 against CFA (UPMC, France). The score-line in the first match, against NZ, reflects the differences in the technology employed by the two teams. NZ was at a great disadvantage from the outset as a result of their decision to use local (on-board) rather than global (overhead) vision. While their robots had difficulty finding the ball, our robots had access to accurate global state information (obtained from an overhead camera). Consequently, this match was effectively a test of the MuCows' ability to score against an open goal. The second match, against Cornell, was a great deal more interesting. Cornell (who went on to win the competition), demonstrated three decisive technologies: effective dribbling, effective kicking and omni-directional drive. In contrast, while our robots were equipped with very similar dribbling and kicking mechanisms (see Section 7), they lacked the omni-directional drive. Not only were the Cornell robots more maneuverable, they were able to use this maneuverability to enhance the effectiveness of their dribbling and kicking mechanisms. This match was effectively a test of our defense (somewhat weak) against Cornell's sustained attack. The final match, against UPMC, was by far the closest of the three matches. Both teams employed similar technology, resulting in a hard-fought match. Unfortunately (for us), UPMC was able to score a goal late in the second half using a clever 'spin-kick' technique. This technique allowed UPMC to attack the goal from low angles, making blocking extremely difficult.

The competition highlighted a number of strengths and weaknesses in our team. The strengths were: the robot chassis, which proved to be rugged and reliable; the vision system, which as quick to calibrate and very reliable; the communication system, which was by far the most capable system fielded by any team. On the other hand, our weaknesses included: poor player skills (the robots could not shoot straight); poor teamwork (the robots executed fixed roles and failed to exploit opportunities). In short, our team was strong on the engineering aspects of the problem (hardware, software architecture), but weak on the science aspects (AI, teamwork).

The remainder of this paper will focus on our team strengths; we will present a high-level overview of the robot hardware and software architectures, together with a somewhat more detailed look at the specific technologies of vision and communication. This paper is perhaps most useful to would-be RoboCup competitors as a ‘cheat-sheet’ on building a reliable hardware and software platform on which to base further development.

2 Team Development

Team Leader: Andrew Howard

Team Members:

Jason Thomas, Thanh The Pham,

Lam Moc Truong, Kevin Phuoc Duy Nguyen

– University of Melbourne, Australia

– Undergraduate students (present a competition)

Andrew Peel, John Horvath, David Hornsby, Thomas Weichert

– University of Melbourne, Australia

– Technical support (present at competition)

Web page <http://www.cs.mu.oz.au/robocup>

3 Hardware Platform

We employ a fairly conventional F180 set-up, with both on-field and off-field components. On-field, we employ a set of five simple-but-sturdy robots. The robots are equipped with a differential drive system, odometry and battery sensing, a relatively powerful 8-bit RISC CPU and a spread-spectrum communications module. Very little computation is performed on the robots; they implement basic PID velocity control, but otherwise act as “remote controlled” vehicles. Off-field, we employ a single overhead camera and frame grabber, a cluster of three PCs, and a spread-spectrum communications module. The vast majority of the sensing and computation is carried out by the off-field components.

The communications technology employed by our team deserves particular attention, as it is one of the strengths of our team. The overwhelming majority of the competitors in the F180 league employ Radiometrix RPC transceiver modules. These modules are compact, low-power, cheap and easy to use; unfortunately, they will only transmit on two frequencies: 418 and 433MHz. This presents two difficulties: from an organizational standpoint, it implies that only two teams can be playing or practicing at any given time; from a reliability standpoint, it implies that one must expect significant interference (this is a shared band). After early experiments with these devices, we decided to invest some time in researching alternatives, and eventually located a multiple-channel spread-spectrum digital transceiver from Innomedia. This device operates at 2.4Ghz, has 12 channels (thus avoiding the frequency clash issue) and uses spread-spectrum technology (thus reducing the interference). We were able to manufacture an adapter board that makes these modules pin-and-protocol compatible with the

Radiometrix RPC modules, making the transition to these devices very straightforward. Cost is comparable to the RadioMetrix RPC modules, but with less latency, higher through-put (140kb/s vs 40kb/s) and better reliability. The devices are, however, somewhat larger and require more power. During the contest, we were one only a few teams not to experience radio interference problems.

4 Software Architecture

The MuCows software architecture is built around a set of modules that interact by passing messages. The message-passing mechanism is based on a publish/subscribe metaphor; any given module will *publish* some message types and *subscribe* to others. This design has a key advantage – since modules have no direct knowledge of each other, whole modules can be replaced without affecting the remainder of the system. Thus, for example, the system can be made to interact with either the physical robots or a simulation of the robots through the replacement of a single module. The message-passing mechanism also has the additional advantage that it facilitates the creation of distributed systems; during the contest, the total system was in fact distributed across three computers.

5 World Model

We are able to construct a fairly complete description of the state of the world. The state description includes: the position and velocity of the ball; the position, orientation and velocity of our own robots; the position, orientation and velocity of the opposition robots. All of this information can be obtained more-or-less directly using raw data from the overhead camera, together with post-processing through a set of Kalman filters. The basic algorithm is as follows:

- Perform a basic color segmentation on the image pixels. Each pixel is determined to be part of either a robot, the ball or the background.
- Apply a median filter to the segmented image. This removes removes most of the pixel noise.
- Cluster the remaining pixels into blobs. Each blob is determined to be either one of our robots, one of the opposition robots, or the ball.
- Identify blobs. For our own robots, use pattern matching on the surrounding pixels to determine the identity of the robot (each robot has a unique black-and-white pattern). For opposition robots, use a track-continuation algorithm to assign an arbitrary identity to each robot.
- Feed the identity and location of the blobs into a set of Kalman filters to extract a filtered estimate of each object’s pose and velocity.

In addition to determining the state of the world, we also make predictions about future states. For the opposition robots and the ball, we use a Kalman filter to generate the prediction. For our own robots, we allow each robot to advertise its *intentions*; i.e. its planned trajectory. Using these predictions, we are able to implement behaviors which are both dynamic and cooperative, such as passing, interception and blocking. See [1] for a full description of this approach.

6 Strategy

MuCows team strategy is implemented through the concept of set-plays, in which each robot is assigned a particular role (such as striker, blocker, etc). The set-plays include the concepts of possession, passing, blocking and looking for openings, and can express quite complex team interactions. Unfortunately, they do not provide a mechanism for role *re-assignment*. During the contest, it quickly became evident that this was a major deficiency: robots executing fixed patterns of behavior would fail to exploit emergent opportunities. For example, if a robot was assigned the role of blocker, it would not attempt to score goals even when a scoring opportunity presented itself. Clearly, a much more flexible and opportunistic approach is required.

7 Special Team Features

The MuCows is one to two teams (Big Red, from Cornell being the other) to introduce dribbling mechanisms into the contest. Our mechanism is a rapidly rotating roller that applies significant back-spin to the ball. As a consequence, the robot is able to ‘kick the ball to itself’. In pre-contest experiments, we were able to achieve a very high degree of ball control: dribbling the ball all the way from the back line to the goal, for example. Under actual game conditions, however, we were unable to fully capitalize on this ability. The rules on contact are such that opposition robots could easily strip an attacking player of possession. Interestingly, Cornell was able to make much better use of their roller due to the superior maneuverability afforded by their omni-directional drive mechanism.

8 Conclusion

Analysis of the 2000 contest has revealed three major areas that need to be addressed: increased mobility through the construction of omni-directional drives, improved skills through better tuning of low-level controllers, and improved game play through the use of more flexible approaches to teamwork. Given the demonstrated robustness of our basic hardware and software architectures, we feel we are in an excellent position to address these issues and return the the contest in 2001 with a much improved team.

References

- [1] Andrew Howard, Alan Blair, Dariusz Walter, and Ed Kazmierczak. Motion control for fast mobile robots: a trajectory-based approach. In *Australian Conference on Robotics and Automation*. Australian Robotics and Automation Association, 2000.

Role-based Strategy in TPOTs RoboCup Team

Nadir Ould Khessal

Temasek Engineering School, Temasek Polytechnic
21 Tampines Avenue 1 Singapore
nadirok@tp.edu.sg

1. Introduction

This paper describes a role-based strategy implemented in TPOTs RoboCup team. A detailed work on the physical robots as well as the overall architecture can be found in [1,2]. Like most small size teams, TPOTs uses a global vision system. Ball position, opponent robots positions, team robots positions and orientations as well as landmarks on the field (e.g. goal keeper area) are captured using a global CCD camera.

2. Team Development

Team Leader: Nadir Ould Khessal

Team Members:

Nadir Ould Khessal

- Affiliation: Temasek Eng. School
- Country: Singapore.
- Position: Lecturer.
- Attended the competition.

Teng Jit Sin

- Affiliation: Temasek Eng. School
- Country: Singapore.
- Position: Technical Staff.
- Attended the competition.

Maung Yan Naing

- Affiliation: Temasek Eng. School
- Country: Singapore.
- Position: Student.
- Attended the competition.

Eric Ng Boon Hwee

- Affiliation: Temasek Eng. School
- Country: Singapore.
- Position: Student.
- Attended the competition.

Pyi Soe Oo

- Affiliation: Temasek Eng. School
- Country: Singapore.
- Position: Student
- Attended the competition.

Lui Hoi Shun Antony

- Affiliation: Temasek Eng. School
- Country: Singapore.
- Position: Student.
- Attended the competition.

Web Page <http://rag.tp.edu.sg/Tpots>

3. TPOTs Robots Architecture.

The new generation of TPOTs uses faster on-board computing board called the Vesta Board. The Vesta Board is based on MC 68332 processor with 16 MHz crystal, 1Mbyte of EEPROM and 512Kbyte of RAM. It has 16 TPU (Timer Processing Unit) channels used to produce Pulse Width Modulation- PWM signals to drive the motors and capturing position pulses from motor encoders.

The robots have been designed to run at approximately 2m/sec. Driven by a DC motor and controlled in a closed loop fashion. Robot positioning on the field is more reliable.

The global vision system consists of a CCD camera and a Cognachrome vision card. The vision board is an external standalone vision card with a processing speed of a maximum 60 frames per second. There are six channels used to train and track up to six different colors. Each of our team robots is equipped with two color pads, robot reference color and robot position color. For obstacle avoidance purposes we capture the opponent reference color, which can be either blue or yellow (see F180 rules).

The off-board computer communicates with the robots using radio frequency modules. Our robots were designed to support four different frequencies 418,433,900 and 800 MHZ. Depending on the frequency used by the opponent we could switch from one frequency to the other simply by plugging the suitable board.

4. Off-board Software Architecture.

There are three main parts in the off-board software namely robot tracking, strategy, and graphical user interface. The graphical User Interface (GUI), Figure 1, is used mainly to start and stop the game and real time monitoring of each robot behavior. The program was developed using Visual C multithread programming.



Fig. 1. TPOTs Graphical User Interface.

4.1 Robot Tracking.

Since we are using two colors for each robot, the user assigns robot ID by clicking on the robot reference color in the GUI. The program then searches for the nearest second color (position color). Starting from the next execution cycle, the program performs two nested searches by looking for the nearest reference color followed by the search for the second color. Even though looking for nearest color is logically correct, tracking errors occurred due mainly to errors from the vision card. A correction routine was then added to filter noisy data.

4.2 Role Based Strategy.

Our strategy is based on subdividing the field into attack, defense and goal area. There are a total of five zones, Left Attack, Right Attack, Left Defense, Right Defense and Goal Keeper Zone (figure 2). The strategy assigns the appropriate behavior for each robot depending on its current position and the ball position. Except the goalkeeper all robots could be assigned any of the following behaviors:

Intercept the ball, kick towards the opponent goal, following the ball, blocking the ball and homing.

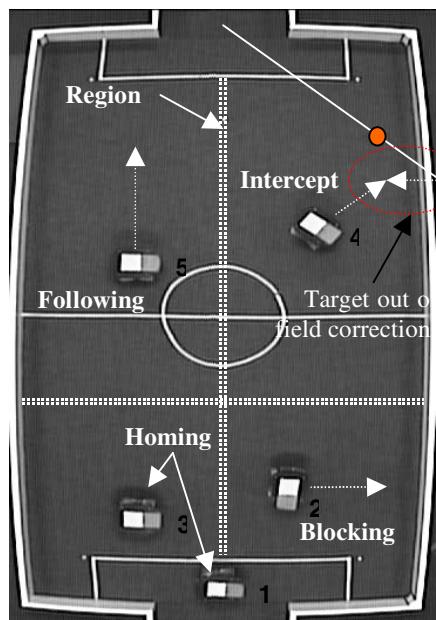


Fig. 2. TPOTs Graphical User Interface

As in Figure [2], if the ball is in the Attack Right Zone robot 4 will be assigned the intercept ball behavior for attacking purposes. Robot 5 will be assigned the Following ball behavior to assist robot 4 in the attack. Robot 2 will be assigned the ball block behavior ready to collect the ball in case an opponent bounces it back. Robot 3 and robot 1, they are both assigned the homing behavior. Since the ball is quite far from both goalkeeper and defense left zones the robots are prevented from performing any unnecessary movements.

5. Conclusion

In this paper we chose to give an insight into the off-board software architecture instead of a full description of the team. The main reason was to provide a simple starting platform for robotics community interested in pursuing the RoboCup challenge.

References

1. Nadir Ould Khessal, Maung Yan Naing, Eric Ng Boon Hwee, Lui Hoi Shun Antony and Pyi Soe Oo *Vision Based Autonomous Soccer Robots* TENCON 2000 Intelligent Systems and Technologies for the Next Millennium. 25th -27th September, 2000 Kuala Lumpur, Malaysia
2. Nadir Ould Khessal. *Towards a Distributed Multi-Agent System for a Robotic Soccer Team*. In Ed. Manuela Veloso, Enrico Pagello and Hiroaki Kitano, *RoboCup-99: Robot Soccer World Cup III*. Springer Verlag, 2000, in this volume.

LuckyStar II - Team Description Paper

Ng Beng Kiat (nbk@np.edu.sg), Quek Yee Ming, Tay Boon Hock, Yuen Suen Yee, Simon Koh

Ngee Ann Polytechnic, Singapore

1 Introduction

Our first robotic soccer team, LuckyStar, competed in Stockholm 1999 and was lucky enough to win third place. From that experience, we found that our team weaknesses were due to lack of vision reliability, smooth robot movement control, shooting capability and team cooperation. We decided to concentrate on areas with the most immediate impact on the game, i.e. vision, robot movement control and shooting mechanism.

2 System Description

We continue to use a global-vision-based system except that the vision system now resides in the host computer. The host computer computes the next move for each robot. The outputs, which are the robot's translational and rotational speeds, are sent to the robots via a Radiometrix RF transceiver.

3 Robot

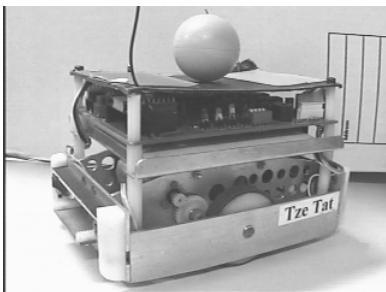


Figure 1. Picture of robot

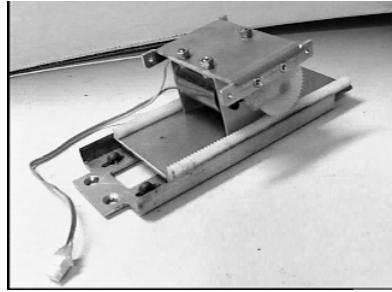


Figure 2. Kicking mechanism exposed

The new robots are similar to the old robot except that they have kicking mechanism. The kicking mechanism is based on a rack and pinion system driven by a DC motor. This allows the robot to kick both way and with varying amount of force. The robot is

able to shoot with full force at opponent goal and with just enough force when clearing the ball towards the sloped wall so as to avoid putting the ball off field.

The host system decides completely when to kick the ball. In general, once the ball is within kicking distance, angular error between the robot orientation and the ball-target-to-robot angle is used to determine shooting criterion. The further the robot is from the ball-target, the smaller the angle error allowance.

4 Vision system

One of the problems faced last year is the detection of opponent color. The ping pong balls are small and not evenly illuminated as they are not flat. In order to be able to use only simple vision algorithm, the camera and the frame grabber must be carefully chosen so as not to lose any ping pong ball color pixel unnecessarily.

4.1 Hardware

We tested some cameras and concluded that 3-ccd cameras are the best in term of resolution but they are extremely expensive. Not being able to afford one, we settled for the second best, a single-ccd camera with RGB output. With composite video, the color information are encoded at the camera end and decoded by the frame grabber board. The result is the degradation of signal-noise ratio, hence poorer color resolution.

It is also important that camera has electronic shutter capability in order to be able to capture high-speed object. The shorter the exposure time, the sharper the high-speed object. Of course, we must make sure the lighting requirement for short exposure can be met.

For the frame grabber, we chose the FlashBus MV Pro from Integral Technologies as it provides RGB input and field capture interrupt capability for fast vision processing response.

4.2 Software

We rely on color patches on top the robot to determine robot orientation. In order to play a game, the vision system must be able to detect the orange ball, the yellow and blue team colors and one other secondary colors. True enough, with proper choice of hardware, we are able to use HSI thresholding to detect 5 good colors without any post-processing. This enables us to process the vision data at 50 field per second on a Pentium 500MHz PC with spare capacity for the host software.

5 Robot Motion Control

As in all sport, speed is a major factor in determining the outcome of the game. But if you have speed without good skill and control, then the game will end up pretty rough and unpredictable. One of our major goals is to achieve high speed and smooth robot control.

5.1 Video lag

In general, by the time the robot receives the movement speed commands, there is a time lag of about 4 video field time between its current position and the processed vision data. The time lag is due to a) camera exposure and capture (1 field), b) frame grabber capture(1 field), c) vision and host processing (slightly less 1 field) and d) RF transmission (slightly less 1 field). To have smooth high-speed control of the robot, the time lag must be compensated. There are two ways to do it.

- 1) Use filtering (e.g. G-H filter[2]) to estimate the robot position, speed and orientation in 4 field time or
- 2) Use the last four speed commands sent to the robots to estimate the robot position, speed and orientation in 4 field time.

We use the second method, which gives a better estimate unless the robot is obstructed.

5.2 Moving Target

In order to block or kick a moving ball, the system must be able to move the robot to the predicted ball position. This must take into account the current ball speed and direction and current status of robot. For our system, an estimate of the time to get to the current ball position is calculated based on the robot-ball distance, current-robot-orientation and final-robot-orientation difference, and average robot translational and rotational speed. With the estimated time, we can estimate the new ball position. With the estimated new ball position, the process is performed for another 2 iterations in order to get a better estimate of the ball position. In general, this works well for only slow ball and is pretty useless for ball speed greater than 0.2 m/sec. We will be spending more effort in this area for our next team.

7 Conclusion

Our robotic soccer team is fairly reliable and able to play a match continuously without many problems. But it is still very lacking in the area of team cooperation.

Next year, we hope to develop ball-passing skills for our robots. With good ball-pass skills, team cooperation will be more feasible.

References

- [1] Manuela Veloso, Michael Bowling, and Sorin Achim. The CMUnited-99 small robot team.
- [2] E. Brookner, Tracking and Kalman Filtering Made Easy. A Wiley-Interscience Publication, 1998

FU-Fighters 2000

Raúl Rojas, Sven Behnke, Lars Knipping, and Bernhard Frötschl

Free University of Berlin, Institute of Computer Science
Takustr. 9, 14195 Berlin, Germany
{rojas|behnke|knipping|froetsch}@inf.fu-berlin.de
<http://www.fu-fighters.de>

1 Introduction

Our F180 team, the FU-Fighters, participated for the second time at the RoboCup Competition. Our main design goal for RoboCup'2000 was to build a heterogeneous team of robots with different shapes and characteristics, and to use new kicking devices. Different sets of robots can be selected to adapt to the strategies of other teams. Our team finished second, as the runner-up to Big Red.

2 Team Development

Team Leader: Raúl Rojas (college professor)

Team Members:

Sven Behnke (scientific staff): general design

Lars Knipping (scientific staff): behavior control

Bernhard Frötschl (scientific staff): web site, mechanics

Achim Liers (scientific staff, did not attend competition): electronics

Wolf Lindstrøt (student): behavior control

Mark Simon (student): computer vision, user interface

Kirill Koulechov (student): behavior control, communication

Lars Wolter (student): behavior control, user interface

Oliver Tenchio (student): mechanics, electronics

3 Mechanical and Electrical Design

We built three types of field players and goal keepers as shown in Figure 1. All robots have sturdy aluminum frames that protect the sensitive inner parts. They have a differential drive with two active wheels in the middle and are supported by two passive contact points. Two Faulhaber DC-motors allow for a maximum speed of about 1.5-2.5 m/s. The motors have an integrated 19:1 gear and an impulse generator with 16 ticks per revolution. One distinctive feature of some of our robots is a kicking device that consists of a rotating plate that can accumulate the kinetic energy produced by a small motor and release it to the ball on contact. Our smaller robots use a kicking device that is based on solenoids that can be activated faster than the rotating plate.

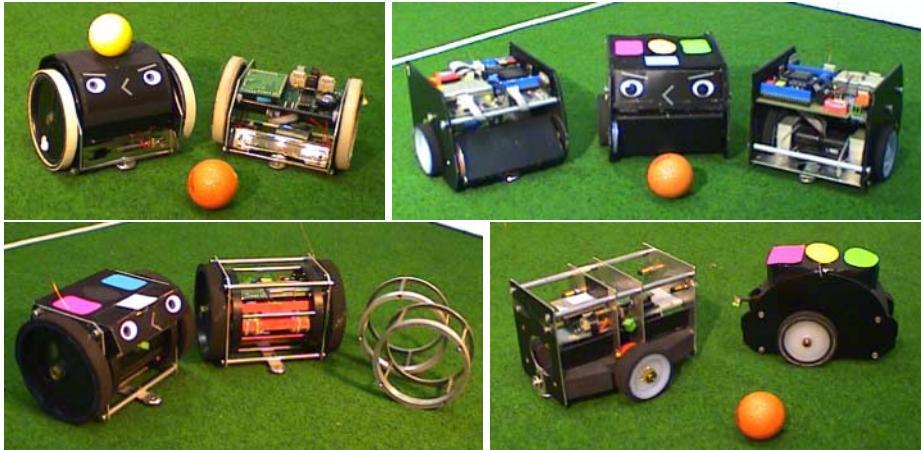


Fig. 1. Different generations of robots.

For local control we use C-Control units from Conrad electronics. They include a microcontroller Motorola HC05 running at 8 MHz with 8 KB EEPROM for program storage, two pulse-length modulated outputs for motor control, a RS-232 serial interface, a free running counter with timer, analog inputs, and digital I/O. The units are attached to a custom board containing a stabilized power supply, a dual-H-bridge motor driver L298, and a radio transceiver SE200 working in the 433MHz band that can be tuned to 15 channels in 100kHz steps.

The robots receive commands via a wireless serial link with a speed of 19,200 baud. The host sends 8-byte packets that include address, control bits, motor speeds, and checksum. The microcontroller decodes the packets, checks their integrity, and sets the target values for the control of the motor speeds. No attempt is made to correct transmission errors, since the packets are sent redundantly.

Our robots are powered by 8 Ni-MH rechargeable mignon batteries. To be independent from the charging state of the batteries, we implemented locally a closed loop control of the motor speeds.

4 Tracking Colored Objects in the Video Input

The only physical sensor for our behavior control software is a S-VHS camera that looks at the field from above and outputs a video stream in NTSC format. Using a PCI-framegrabber we feed the images into a PC. We capture RGB-images of size 640×480 at a rate of 30 fps and interpret them to extract the relevant information about the world. Since the ball and the robots are color-coded, we designed our vision software to find and track multiple colored objects. These objects are the orange ball and the robots marked with two colored dots in addition to the yellow or blue team ball.

To track the objects we predict their positions in the next frame and then inspect the video image first at a small window centered around the predicted

position. We use an adaptive saturation threshold and intensity thresholds to separate the objects from the background. The window size is increased and larger portions of the image are investigated only if an object is not found.

The decision whether or not the object is present is made on the basis of a quality measure that takes into account the hue and size distances to the model and geometrical plausibility. When we find the desired objects, we adapt our model of the world using the estimates for position color, and size. We also added an identification module to the vision system that recognizes the robots by looking at a black and white binary code.

5 Hierarchical Generation of Reactive Behavior

The Dual Dynamics control architecture, proposed by Jäger [3], describes reactive behaviors in a hierarchy of control processes. Each layer of the system is partitioned into two modules: the activation dynamics that determines whether or not a behavior tries to influence actuators, and the target dynamics, that determines strength and direction of that influence. The different levels of the hierarchy correspond to different time scales. The higher level behaviors configure the lower level control loops via activation factors that determine the mode in which the primitive behaviors are. These can produce qualitatively different reactions if the agent encounters the same stimulus again, but has changed its mode due to stimuli that it saw in the meantime.

A more detailed description of our control architecture, that is based on these ideas, is given in [1]. The robots are controlled in closed loops that use different time scales. We extend the Dual Dynamics scheme by introducing a third dynamics, the perceptual dynamics. Here, either slow changing physical sensors are plugged in at higher levels, or readings of fast changing sensors, like the ball position, are aggregated to slower and longer lasting percepts. Since we use temporal subsampling, we can afford to implement an increasing number of sensors, behaviors and actuators in the higher layers.

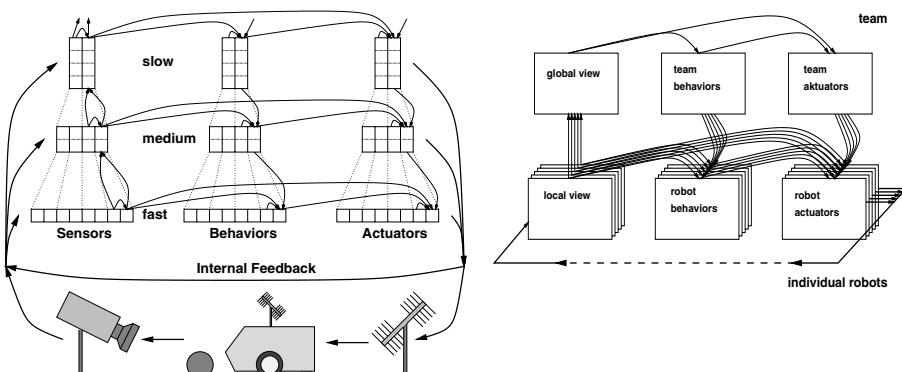


Fig. 2. Reactive control architecture.

Behaviors are constructed bottom up: First, processes that react quickly to fast stimuli are designed. Their critical parameters, e.g. a target position, are determined. When the fast behaviors work reliably, the next level can be added. This level can now influence the environment either directly by moving slow actuators or indirectly by changing critical parameters in the lower level.

In the lowest level of the field player only two behaviors are implemented. These are a parameterized taxis behavior and an obstacle avoidance behavior.

On the next level various behaviors use them. They approach the ball, dribble with the ball, kick the ball, free the ball from corners, home, and so on. Here, we also implemented a ball prediction that allows for anticipative actions. On the third level, we dynamically adjust the home positions of the players, such that they block opponent players or position themselves freely to receive passes.

Each of our robots is controlled autonomously from the lower levels of the hierarchy using a local view to the world. For instance, we present the angle and the distance to the ball and the nearest obstacle to each agent. In the upper layers of the control system the focus changes. Now we regard the team as the individual. It has a slow changing global view to the playground and coordinates the robots as its extremities to reach strategic goals.

In the first team level, we decide which robot should take the initiative and go for the ball. The remaining players are assigned to supporting roles. Passing and strategy changes would be implemented in higher team layers.

6 Future work

Our goal is to make the robots more autonomous. Therefore, we want to add a local omni-directional camera to the robots. Initially we want to transmit the images to an external computer, where they can be analyzed. Later, the vision algorithms should be ported to an on-board computer. We also plan to add other local sensors that can measure the robot's movement. One further step towards autonomy will be to implement the behavior control locally.

The other research direction, we are interested in, is learning. We plan to use reinforcement learning to adapt the parameters of the system. We also hope to improve team play to allow for an increased number of players on a larger field.

References

1. Behnke, S., Frötschl, B., Rojas, R., Ackers, P., Lindstrot, W., de Melo, M., Schebesch, A., Simon, M., Sprengel, M., Tenchio, M.: Using hierarchical dynamical systems to control reactive behavior. In: Veloso, M., Pagello, E., Kitano, H. (eds.) RoboCup-99: Robot Soccer World Cup III, 186–195, Springer, 2000.
2. Christaller, T.: Cognitive Robotics: A New Approach to Artificial Intelligence. In: Artificial Life and Robotics, Springer, 3/1999.
3. Jäger, H., Christaller, T.: Dual Dynamics: Designing Behavior Systems for Autonomous Robots. In: Fujimura, S., Sugisaka, M. (eds.) Proceedings International Symposium on Artificial Life and Robotics (AROB '97), Beppu, Japan, 76–79, 1997.

RoGi Team Description

Josep Lluís de la Rosa, Bianca Innocenti, Miquel Montaner, Albert Figueras,
Israel Muñoz and Josep A. Ramon

Institut d'Informàtica i Aplicacions, Universitat de Girona & LEA-SICA
Lluís Santaló s/n, E-17071 Girona, Catalonia
{peplluis, futbolistes} @ eia.udg.es

1 Introduction

RoGi team started up in 1996 at the first robot-soccer competition as the result of a doctorate course in multi-agent systems. In 1997, 1998 and 1999 it has participated at the international workshops held in Japan, Paris and Stockholm. The main goal has been always the implementation and experimentation on dynamical physical agents and autonomous systems. Through these years the platform has become a stable system, where experiments can be repeated and new theories regarding unstructured, dynamic and multi-agent world can be tested. This paper focuses on decision making and vision improvements carried out during this last year. This year results at the competition were: 0-4, 10-0, 2-0 and 0-14 at quarterfinals.

This year, the team has dealt mainly with the changes on rules. Also, it has improved some aspects as:

- To use robots with different dynamics.
- To correct the distortion introduced by the lens of the camera.
- To introduce roles in the decision-making system in order to co-ordinate the players.

2 Team Description

Team Leader: Josep Lluís de la Rosa

Team Members:

Josep Lluís de la Rosa

- Associate Professor
- did attend the competition

Bianca Innocenti

- PhD Student
- did attend the competition

Miquel Montaner

- PhD Student
- did attend the competition

Albert Figueres

- Associate Professor
- did attend the competition

Israel Muñoz

- PhD Student
- did attend the competition

Josep Antoni Ramon

- Associate Professor
- did not attend the competition

Team Web Page <http://rogiteam.udg.es>

3. Robots Description

The robots have on board 8 bit Philips microprocessors 80C552 and RAM/EPROM memories of 32kBytes. The robots receive data from the host computer by means of a FM receiver. The FM receiver allows working with two different frequencies 418/433 MHz in half-duplex communication. The information sent by the host computer is converted to RS-232C protocol. The two motors have digital magnetic encoders with 265 counts per turn. They need 10 V to work and consume 1.5W at a nominal speed of 12,300 rpm. Ten batteries of 1.2 V supply the energy. There are two power sources, one DC-DC switching regulator, which provides stable 10 V to motors, and another one that guarantees 5 V to the IC.

At present, the RoGi team is formed by robots with different dynamics, which is non-linear but linear piece-wise.

4. Vision System

A specific hardware has been designed to perform the vision tasks, merging specific components for image processing (video converters, analogue filters, etc.) with multiple purpose programmable devices (FPGAs). A real time image-processing tool is obtained, which can be reconfigured to implement different algorithms.

4.1 Camera Distortion

The lens of the camera produces a radial distortion on the acquired image. This effect introduces an error on the position measurements. To minimise this error, a filter is applied to data after processing the image. Also, there is an algorithm that excludes anomalous data, for instance, wrong measurements according to the size of the field. Figure 2 shows the complete treatment that receives the image before making available the data to the control system.

The filter to minimise the effects of distortion is based on the Tsai algorithm [2]. This algorithm converts this data through a set of parameters obtained by a previous calibration of the camera, achieved with the landmarks of the field.

In addition to this filter, a Kalman filter is used to get stable data, which also gives a prediction of the position of the ball some steps ahead.

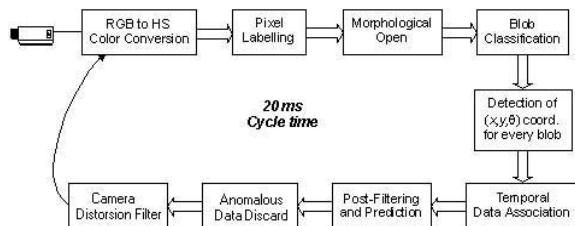


Fig. 1. Treatment of the image.

5. Control System

The control system is a multi-agent environment in which the agents decide the best action to do. Each agent takes first, a reactive decision based on data given by the vision system, and second, a deliberative one considering the teammates decisions. Once all the agents agree on the decisions, each one transforms the actions in orders understandable by the robot, and sends them via radio link. The vision and control systems are implemented in two different computers and they are connected by means of a LAN, using TCP/IP protocol.

5.1 Co-ordination

As each agent is capable of taking autonomous decisions in this multi-agent system, they need to be co-ordinated in order to avoid interference among them when executing the task and to plan the best way to carry out the task together.

In RoGi Team this feature is implemented in a very simple way: robots have roles. These roles have specific capacities. For example, the attacker moves near the opponent's goal and when it gets the ball it tries to score. A midfielder tries to drive the ball to the opponent's field and pass it to the attacker. With these different capabilities, a co-ordinated team behaviour is achieved and conflicts are significantly reduced.

Humans assign the roles of the robots before starting the game, and during it agents can change roles accordingly to the progress of the game. This co-ordination is improved using deliberative decision in the decision system

5.2 Decision System

This is the core of the system and the target of all our efforts. The decision is taken in a two-steps algorithm:

Reactive Decisions Step: In a first step of reasoning, every agent decides a private/local action. To take this private decision, agents have a global real world perception (see figure 2), and a set of capabilities (actions they can perform) that varies depending on the role. The world perception is the result of high level abstraction, using fuzzy using the data from the vision system. With this knowledge of the environment and its capabilities, all the players take the reactive decisions using fuzzy logic. The agent converts this reactive action into its actual belief.

Deliberative (Cooperative) Decision Step: Deliberative reasoning in the sense of [1] is implemented by communicating the former reactive belief. Every agent knows the beliefs of the other teammates. Beliefs contain the reactive decision, the certainty of this decision and the identification of the player-agent (*reactive_belief, certainty, ID_player*).

When two or more agents realise their beliefs are in conflict, they use this certainty coefficient in a consensus algorithm based on a positional method to resolve it. In this positional method players are specialised since they have assigned roles. One possible effect of their specialisation is that they prefer to stay in a certain position on the playground. Agents take advantage of this feature and modify their vision of the co-operative world according to the positions of team players. These modifications lead the agents to reinforce or to change the former reactive decision. As a result, collisions among playmates are significantly reduced comparing to non-adaptive perception of the co-operative world.

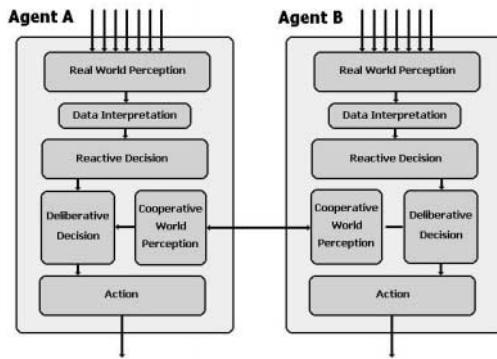


Fig. 2. Schematic structure of agents.

6. Conclusions

This year team goal has been to settle the platform with the aim of developing experiments that can be repeated and testing new theories in an unstructured, dynamic and multi-agent world, simulating a football game.

7. Future Work

We plan to participate at next year competition and improve our decision-making system. Also we will build new and robust robots with better control strategy.

References

1. Busetta P., Rönnquist R., et al., "JACK Intelligent Agents-Components for Intelligent Agents in Java", Agentlink Newsletter, No.2, pp. 2-5, January 1999.
2. Roger Y. Tsai, "A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, page 323-344.

UQ RoboRoos: Kicking on to 2000

Gordon Wyeth, Ashley Tews and Brett Browning

Department of Computer Science and Electrical Engineering
University of Queensland, Australia
{wyeth, tews, browning}@csee.uq.edu.au

Abstract. The UQ RoboRoos have been developed to participate in the RoboCup robot soccer small size league. This paper overviews the history of the team, and provides details of some key factors to the teams success in 2000: a new goalkeeper design, robust communications and smooth, fast navigation. The paper concludes with some thoughts on the future of the RoboRoos.

Overview

The RoboRoos are one of the longest serving teams in the RoboCup Small Size league. The robots competed at RoboCup '98 [7] where they were runner-up, and in RoboCup '99 [8] where they were eliminated in the quarter finals, despite a 56-1 goal difference during the round robin stage. In RoboCup 2000, the team was eliminated in the semi-finals after a 36 - 2 goal difference in the round robin stage. The field robots have remained mechanically and electronically the same through the years of competition, with significant improvements being made through software revision. The goalkeeper has been re-designed several times over this period with the latest revision detailed in this description.

Figure 1 shows the architecture of the RoboRoos system, which has remained constant since its inception. The vision system was extensively revised in 1999 with excellent results [9]. The team planning system, MAPS, has been the focus of ongoing development [4,5,6], and represents one of the most significant research results of the RoboRoos development. MAPS forms cooperative strategies by observing team agents at the current point in time and choosing appropriate actions to increase the likelihood of cooperation in the near future. These actions are transmitted to the robots, along with the current state of the field, over the RF communications link. The RF communications link has received significant attention to ensure reliability under adverse conditions. The robots use the information from the communications link to set the current goal for navigation, and to build representations of obstacle maps for path planning. The navigation module for the robots has been another area of significant research effort [2], and provides the smooth controlled motion that is the signature feature of the RoboRoos.

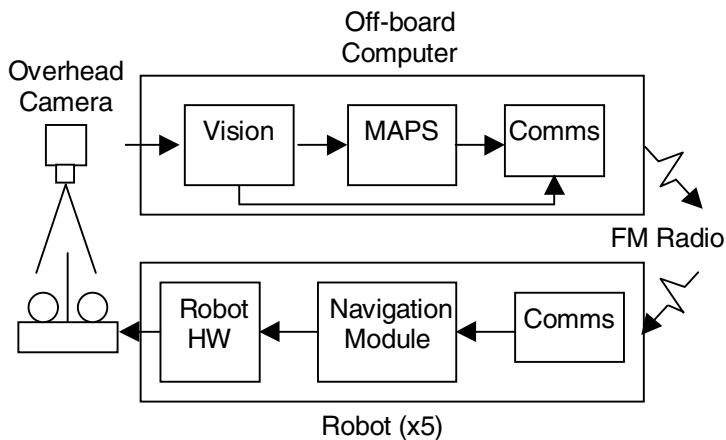


Fig. 1. The RoboRoos system in overview.

This team description will focus on the parts of the system that are not described elsewhere, and that have contributed significantly the team's success in RoboCup 2000 - the new goal keeper design, the communications link and the navigation system. The conclusion focuses on the future of the RoboRoos.

Goalkeeper Design

Past goalkeeper designs for the RoboRoos attempted to achieve maximum performance by having a four wheel drive design with slip steering. The justification for this approach was that the robot would mostly function on a single linear axis, with minor corrections for slip. In practice, this assumption was invalid. As we developed algorithms for our goalkeeper, it became clear that good rotational control was also important. The slip steering method was found to be very hard to control, and extremely hard on battery consumption.

In 2000, we adopted the approach that manoeuvrability was the key, and reverted to a simple wheelchair design. By keeping the centre of mass of the robot low, we were able to minimise "lost" down force on the casters, thus achieving greater grip on the driving wheels. The heaviest components were kept close to the turning centre of the robot, minimising the moment of inertia, leading to vastly improved angular acceleration over the previous slip steer design. The improvement in achievable turning acceleration lead to vastly improved control of the goalkeeper. The nett effect was that usable linear acceleration lifted from 0.3g in the old design, to 0.5g. The time to cover the goal mouth went from 0.9 seconds to 0.7 seconds with the new design, but most importantly the goal keeper remained under control at all times.

The adoption of this simple design for a goalkeeper also allowed us to have a second robot with a large side-on profile. In 2000, we employed this robot as a sweeper which operated in a similar fashion to the goal keeper immediately in front of the goal area. During game play this robot was able to stop a number of long range

shots, but was most effective as a tackling defender with its fast sideways motion making it very difficult to pass while dribbling.

Communications Link

The communication link is a one way design giving the controlling computer the ability to transmit field state and strategic commands to the robots. The hardware design of the link is centred around the use of a RadioMetrix TX2 radio transmitter and RX2 radio receivers. These modules offer a serial data rate of 40 kbps, a useful indoor transmission range of 30m and are able to be directly interfaced to a TTL level signal. One of the weaknesses with this product is that the data slicer is optimised for signals with a 50:50 mark to space ratio. Our investigations found that the error rates quickly rose to over 20% for a 60:40 or 40:60 mark to space ratio, with a massive 80% error rate at the specified outer limit of 30:70 and a slightly lesser 50% for 70:30. The key factor for the use of these modules is clearly a protocol that maintains a 50:50 mark to space ratio.

The simple protocol we adopted for obtaining a 50:50 mark to space ratio is to transmit every byte twice, once in the usual fashion and once inverted. Upon reception, each byte is checked for parity error and for a match between the two copies of the transmitted byte. Any errors cause the entire 48 byte packet for that visual frame to be discarded. Under this protocol, detected error rates were kept below 2% in playing conditions - even when other teams using the same communication modules were having total communication loss. In testing in the lab, we were unable to generate any errors that were not detected by the error detection protocol.

Navigation

The RoboRoos navigation system has been re-developed from those reported in [2] and [7]. The new module is based on the RanaC model of frog prey catching behaviour [1], which we have dubbed RanaR. The module is based on three radial potential field maps of the environment - Goal Direction (GD), Obstacle Map (OM) and Motor Heading Map (MHM). The GD representation is formed by the goal location received from the MAPS module. It takes the form of a Gaussian field centred on the goal direction. The OM representation is built from the robot position information received from the vision system, with each obstacle contributing to the field based on the heading and distance to the obstacle. The MHM is then built by subtracting the OM from the GD, and applying competitive attractor dynamics to smoothly select a distinct winner in the resulting potential field.

The distinguishing features of the RanaR model from the RanaC model are use of path integration to account for delay in sensory information, the use of pragmatic models of competitive attractor dynamics to achieve real time computation, and the inclusion of a speed module to determine the average speed of navigation for the given situation. The latter is essential for any wheeled system as the robot must slow

to approach goals without overshooting, and should travel slowly in crowded regions to allow faster response to potential collisions.

RoboRoos 2001

If the RoboRoos are to become champions of the Small Size league, it is clear that serious revision of the mechanical design is required. The addition of a mechanical kicking mechanism adds several benefits to a robot soccer team. Not only is there potential for more powerful kicks, as demonstrated by the FU-Fighters, there is the possibility of one-touch passing and deflections as employed by Cornell Big Red in 1999, and Lucky Star II in 2000. It is far simpler to move to a point where the ball may be intercepted and wait to employ the kicking mechanism, than it is to attempt to navigate the robot through the moving ball. For this reason, the RoboRoos have great difficulty with any opponent who can keep the ball moving. In 2001, the RoboRoos will investigate kicking and dribbling mechanisms.

It is likely also that the RoboRoos will take advantage of new offerings in communication modules to improve communications bandwidth, use a faster off-board computer to improve vision resolution and explore new playing strategies using MAPS. While much of our research focus is now centred around our new ViperRoos [3] team, the RoboRoos will continue to offer technical and personal challenges to the undergraduates at the University of Queensland.

References

1. Arbib, M.A. and Lee, H.B. (1993) Anuran visuomotor coordination for detour behaviour: From retina to motor schemas. *From Animals to Animats 2: Proc of SAB*, MIT Press.
2. Browning B., Wyeth G.F. and Tews A. (1999) A Navigation System for Robot Soccer. *Proceedings of the Australian Conference on Robotics and Automation (ACRA '99)*, March 30 - April 1, Brisbane, pp. 96-101.
3. Chang M., Browning B. and Wyeth G.F. (2001) ViperRoos 2000, RoboCup 2000: Robot Soccer World Cup IV. LNAI, Springer Verlag, in this volume.
4. Tews A. and Wyeth G.F. (2001) MAPS: A System for Multi-Agent Coordination. *Advanced Robotics*, VSP / Robotics Society of Japan, accepted for publication.
5. Tews A. and Wyeth G.F. (2000) Thinking as One: Coordination of Multiple Mobile Robots by Shared Representations, *Proceedings IROS 2000*.
6. Tews A. and Wyeth G.F. (1999) Multi-Robot Coordination in the Robot Soccer Environment. *Proceedings of the Australian Conference on Robotics and Automation (ACRA '99)*, March 30 - April 1, Brisbane, pp. 90-95.
7. Wyeth G.F., Browning B. and Tews A. (1999) UQ RoboRoos: Preliminary Design of a Robot Soccer Team. *Lecture Notes in AI: RoboCup '98*, 1604.
8. Wyeth G.F., Browning B. and Tews A. (1999) UQ RoboRoos: Ongoing Design of a Robot Soccer Team. *Team Descriptions: RoboCup '99*.
9. Wyeth G.F. and Brown B. (2000) Robust Adaptive Vision for Robot Soccer, Mechatronics and Machine Vision, Research Studies Press, pp. 41 - 48.

ART'00 - AZZURRA ROBOT TEAM FOR THE YEAR 2000

Giovanni Adorni ¹, Andrea Bonarini ², Giorgio Clemente ³, Daniele Nardi ⁴,
Enrico Pagello ⁵, Maurizio Piaggio ⁶

¹ DII-Università di Parma, ² DEI-Politecnico di Milano, ³ Consorzio Padova Ricerche, ⁴ DEI-Università di Padova, ⁵ DIST-Università di Genova, ⁶ DIS-Università di Roma "La Sapienza"

1 INTRODUCTION

Robotic soccer is a challenging research domain that can be used to explore new problems and test new techniques/ solutions in the fields of Artificial Intelligence and Autonomous Robotics, as well as for training and educational purposes.

Azzurra Robot Team (ART) is a national project launched in 1998 in the framework of RoboCup Initiative with the aim of education and training of students in the fields of Artificial Intelligence and Autonomous Robotics as well as developing and testing new research ideas useful for indoor robotics. ART is the result of a joint effort of several Italian research groups. The goal of the project is to exploit the expertise and ideas from different research groups around Italy in order to build a team where players have different features, both hardware and software, but retain the ability to coordinate their behaviour within the team. Therefore, ART is an heterogeneous team where each player synthesizes experiences/solutions of different research laboratory.

ART achieved the second place in RoboCup '99 in Stockholm, and ranked second, first among the European teams, in the First European RoboCup Championship held in Amsterdam in May 2000. In this paper we describe the main characteristics of ART'00, the team participating in the RoboCup Initiatives during the year 2000. More precisely, next section describes some features of the players; section 3 discusses the problem of coordination among the team; section 4 presents a card for each player participating in RoboCup 2000; and last section thanks all the ART-ists involved in this project.

2 ROBOT FEATURES

Each player differs from its team mates in many ways: mechanics, sensors, computer hardware, control software and, most of all, design. The design in particular is not only different (each research group develops its own robot) but also operates in a substantially independent way. Presently, we are using different robotic bases, all of them characterized by a conventional PC, wireless high bandwidth connection, low cost frame grabber(s), CCD camera(s), odometry and kicking device.

The vision systems, the most important sensor of ART'00 players, rely on three different camera setups: omnidirectional, binocular, single (fixed or pan-tilt) camera. Omnidirectional vision systems consist of an upwards-oriented camera that acquires the image reflected on a concave mirror hanging above it, obtaining a field of view of 360 degrees. In single camera systems the field of view is quite narrow, limiting the distortions introduced by the lens and making object localization in the acquired image quite reliable. If pan-tilt devices are present, the camera motion can compensate for the limited field of view. Multi-camera vision provides a wide field of view with little distortion requiring no moving devices. However, either as many frame grabbers as the number of cameras or a multi-input frame-grabber is required. Camera calibration may be critical as it must ensure that information coming from different cameras is homogeneous and not contradictory in the regions where the camera fields of view overlap.

Vision algorithms (VAs) have been designed to tackle: object detection, obstacle avoidance, player self-localization. In light of the different geometry of the vision systems, VAs have one class independent of the acquisition system, concerning object and landmark detection, based on fast segmentation techniques by means of adaptive thresholding in the color space. A second class, device-dependent, has to take into account the different geometry in the interpretation of distances and orientations of the objects as they appear in the acquired image. In that case, VAs may involve not only vision but also vision/control interaction. In omnidirectional vision setups, object/landmark detection can be done all around the robot with the analysis of a single frame. In pan-tilt camera systems, the robot control system has to drive two systems, the robot and the camera, and has to analyze their relative motion in real time and accurately enough to follow the ball motion at any time. Finally, in binocular systems, even if it is possible to acquire images from all cameras at the same time, real-time constraints suggest that only the relevant ones be processed. Therefore, VA must include some kind of spatial reasoning to choose the right buffer to process.

3 TEAM COORDINATION

Coordination of players in ART'00 is particularly challenging because of its significant heterogeneity. A first level of coordination (low-level communication framework) is strictly related to the chosen software architecture (ETHNOS) and its message based communication protocol (EIEP - Expert Information Exchange Protocol); a library has been also developed to allow the communication with other architectures (i.e., SAPHIRA). The EIEP deals transparently both with inter process communication within a single robot and with inter-robot communication. In the EIEP the different robots are allowed to subscribe to communication clubs in which messages are exchanged with a publish/subscribe technique. Whenever a message is published ETHNOS transparently and dynamically distributes the messages to the appropriate subscribed receivers.

In ART'00 we are allowing the robots to communicate in a single club (the team) and with an external monitoring station (the coach) which monitors the

activity of all the players for displaying and debugging purposes during a match. In this club robots can be added and removed dynamically without explicit programming. A second level of coordination is related to the role the players: goalkeeper (role_0), main attacker which demands ball possession (role_1), helping attacker which supports the main attacker in the action (role_2), defender which attempts to recover the ball (role_3). Three field zones are associated to a specific role. The zones generically describe the regions in the field (attack, defense, field center) in which the respective role identified behaviour should be carried out. Two utility functions are used in the robot/role association. The goalkeeper is always associated to the designated robot unless a malfunction forces another robot to temporarily act as a substitute. For the remaining (MF-Middlefield) players the role/player association based on the above structure is: « Every cycle (100 ms) each robot transmits a message containing the computed value for the two utility functions based on its relative perceptions of the external world. Every robot also subscribes to the utility value message type and therefore also receives the utility values of all the other robots in the field. Every cycle each robot compares its first utility value with the one of the other robots. The robot with the lowest value takes role_1. The remaining two robots compare the second utility value. The one with the lowest value takes role_2, the remaining takes role_3».

To avoid role association oscillations a double threshold mechanism has also been introduced to facilitate role keeping by the robots in the presence of similar utility values. The type of roles in the structures and the utility functions used determine the personality of the team and can be changed from match to match.

4 THE PLAYERS

#1 *Galavrón*(Goal Keeper) by DII - Università di Parma: Self made base, frontal/lateral pneumatic kickers, motors in central position with four spherical wheels keeping the robot in balance, home developed control and power cards, binocular vision to allow 220 front view and perform geometric reasoning for self-localization, back/lateral infrared sensors, ETHNOS environment.

#2 *RonalTino* (MF Player) by DIS-Università di Roma"La Sapienza": Pioneer 1 base, pneumatic kicker allowing for left/right kick, Pioneer 1 microcontroller card with modified basic control hw and sw, monocular vision, sonars and infrared sensors, SAPHIRA environment.

#3 *TotTino* (MF Player) by DIS-Università di Roma"La Sapienza": Replica of Ronaltino with pan/tilt camera to reduce the time to locate the ball and to improve precision and speed of the actions approaching the ball.

#4 *Bart* and #5 *Homer* (MF Players) by DEI-Università di Padova: Pioneer 1 base, pneumatic kicker with an effector allowing for frontal/lateral kick, Pioneer 1 microcontroller card, home developed power card, monocular vision, ETHNOS environment.

#6 *Relé* (MF Player) by DIST - Università di Genova: Pioneer 1 base, pneumatic kicker with lever system able to lift the ball, Pioneer 1 microcontroller card with modified basic control hw and sw, monocular vision, ETHNOS environment.

#7 *Rakataa* (MF Player) by DEI-Politecnico di Milano: Self made base, two independent traction wheels, mechanical kicker, home developed control and power cards, omnidirectional vision, ETHNOS environment.

#8 *Lisa* (Goal Keeper) by DEI - Università di Padova: Modified Pioneer 1 base, pneumatic kicker with an effector opening two lateral wings while moving ahead, motors in a central position with two spherical wheels keeping the robot in balance, Pioneer 1 microcontroller card, home developed power card, omnidirectional vision, ETHNOS environment.

#9 *NakaTino* (MF Player) by DIS-Università di Roma "La Sapienza": Pioneer 2 robot prepared as RonalTino.

5 ART-ists

Team Leader: Giovanni Adorni (adorni@ce.unipr.it)

Dept. of Computer Engineering, Università di Parma
Parco Area delle Scienze 181A, 43100 Parma, Italy

Team Members:

M.Airoldi, D.Baldassarri, M.Barbon, M.Bellini, C.Bernardi, M.Bert,
E.Bocelli, A.Bonarini, A.Brambilla, M.Bresciani, S.Cagnoni, F.Cappelli,
M.Carletti, A.Castaldo, C.Castelpietra, M.Cefalo, R.Cipriani, G.Clemente,
G.Colombo, E.Corbetta, U.Cugini, D.Daniele, A.Farinelli, C.Ferrari,
G.Franceschini, L.Garbarino, M.Giardino, A.Gilio, M.Grillo, G.Invernizzi,
T.H.Labella, L.Luminari, F.Marando, A.Marangon, F.Marchese,
D.Mastrantuono, M.Matteucci, E.Menegatti, P.Meriggi, M.Mordonini,
C.Moroni, D.Nardi, E.Pagello, C.Pellizzari, M.Piaggio, G.Pucci,
G.Remondini, C.Rota, A.Scalzo, A.Sgorbissa, D.Sorrenti, D.Spagnoli,
S.Suriano, M.Tamburini, R.Zaccaria, S.Zaffalon, M.Zoppi

Web page <http://RoboCup.CE.UniPR.IT/ART2000/>

References

- G.Adorni, S.Cagnoni, M.Mordonini, M.Piaggio, Team/goal-keeper coordination in the RoboCup mid-size league, *In this Volume*.
- C.Castelpietra, L.Iocchi, D.Nardi, M.Piaggio, A.Scalzo, A.Sgorbissa, Communication and coordination among etherogeneous mid-size players, *In this Volume*.
- M.Piaggio, A.Sgorbissa, R.Zaccaria, A programming environment for real time control of distributed multiple robotics systems, *Advanced Robotic J.*, 14, 2000.
- A.Bonarini, P.Aliverti, M.Lucioni, An omnidirectional sensor for fast tracking for mobile robots, *Procs. IEEE Instr. & Meas. Tech. Conf.*, Piscataway, 1999.
- E.Pagello, A.D'Angelo, F.Montesello, F.Garelli, C. Ferrari, Cooperative behaviors in multirobot systems through implicit communication, *Robotics and Autonomous Systems*, 29(1), 1999.

RMIT United

James Brusey, Mark Makies, Lin Padgham, Brad Woodvine, and Karl Fantone

RMIT University, Melbourne, Australia
united@rmit.edu.au

1 Introduction

The RMIT United team is entirely based on custom-made robots of two types, one goalie robot and three field player robots. The field players are distinguished by a powerful kicking device and are largely the same as our previous year's entry. The goalie robot is designed to move from side to side, rather than forward and back, and has tank tracks rather than wheels. The robots mainly use vision for perception, and have a dedicated digital signal processor (DSP) to interpret what it sees. The goalie robot has two cameras and two DSPs.

The strategy component is a hybrid, combining a commercial agent development system, JACK¹ [2], with a behaviour-based control mechanism [1]. Robots cooperate with each other using a mechanism that works in a completely distributed manner and without using specific request or acknowledgement messages.

Our game scores were 2–0, 8–1, 7–0, 8–0, 3–5, 1–3, 0–3, 0–2. We made the finals but were eliminated in the preliminaries.

2 Robots

The robot's main chassis is milled from a single piece of 12.5mm structural U-section aluminium. With all components, including batteries and laptop computer, the field robots weigh around 15 kilograms and the goalie weighs about 25 kilograms. Figure 1 summarizes the system architecture.

The robot is controlled via a serial link operating at 9600 baud. To attempt to reduce control latency, encoder and status packets are returned by the robot every 50 milliseconds, even if no command is issued.

The field robots have a special kicking device based around a heavy brass cylinder that slides along a rod for the length of the robot. The cylinder is pulled back against two elastic cords by a worm drive motor (actually a windscreens wiperblade motor). A solenoid actuated clutch holds the cylinder until the robot is ready to kick. When the kicker is armed (via a command from the laptop), it will kick when a touch sensor is pressed. This system is capable of kicking the ball about 20 metres along a carpeted surface and is probably the most powerful kicker in the league. The goalie robot has a solenoid actuated kicker that is hinged so the sides swing out.

¹ JACK is a Java-based agent programming system developed by Agent Oriented Software, <http://www.agent-software.com.au>

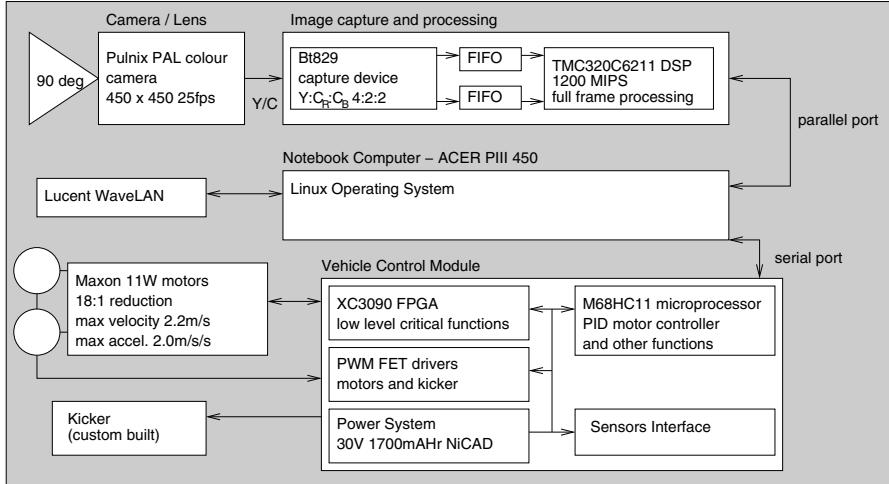


Fig. 1. Hardware architecture of soccer robot.

3 Perception

The robot's perception of the world is almost entirely based on vision. For this competition, we developed a digital signal processing board (DSP) that converted each frame to a list of identified objects. We used a 90 degree field-of-view lens, however this gave substantial distortion. Therefore, the first step in processing the frame is to correct for the fish eye distortion. The image is segmented based on colour thresholds. A Hough transform [3] is used to detect straight lines and then the colours that lie either side are used to work out if the line corresponds to a wall, a goal post or a field line. Some contextual rules are used to eliminate noise. The DSP easily keeps up with the Pulnix camera, which produces 25 frames per second.

The object data is sent to the laptop's parallel port, at which stage the data is in terms of pixel positions. From this, object distances are estimated. The relative positions of the ball and robots are used to update the world model. The relative positions of the walls and goal posts are used for localisation.

4 World Model

The world model is generated in terms of absolute field coordinates. On each perception cycle, some simple rules are used to make sure that the model is realistic. For example, old objects that should be in view but can't be seen are deleted.

The consistency and accuracy of the world model is dependent, in part, on the accuracy of the robot's position and heading estimates. We use wheel encoder

feedback to update our position and heading. Also, a filter is used to correct position estimates based on visual identification of walls and goal posts. Goal posts can be identified uniquely, but since we do not estimate their distance, just their relative angle, yielding infinite possible poses for the robot. We used the simple approach of assuming that the heading contained the most error, and only adjusted the heading.

Wall information supplied by the vision includes the distance to the wall and its relative angle. Therefore, only 4 possible hypotheses need to be considered. We choose the most probable one based on the current heading. Actually, the vision system detected the corners and so there were really 8 possible hypotheses. We filtered corner information by ignoring walls with angles more than $\pi/8$ radians from that expected.

In practice, we had some bugs that caused the localisation to be out. We hope to improve on the accuracy of the localisation in future.

5 Communication

The robots communicate via a radio ethernet LAN (Lucent WaveLAN). All messages are in the form of Multicast IP packets. An off-field laptop is used to stop, start and position the robots. Each robot broadcasts its position and current percepts. Each robot (and the off-field laptop) combines the received packets and forms a global world model. This model is kept separately from the locally derived world model. The global model is used to help find the ball if it's been lost and to determine the robot's role.

Role assignment is performed using a series of heuristics. The heuristics determine which robot is in the best position to take a particular role. Due to timing, the global model on each robot may be slightly different, and it is possible, therefore, for two robots to take the same role. We found that we could minimize this by using continuous functions for heuristics and including a small bonus to the robot previously owning the role (to make the roles slightly "sticky").

Calculating role assignment on every robot and at every cycle seems like a waste, however we found that the heuristic functions were generally very simple and so there was minimal cost. Also, it allowed us to make the system highly fault tolerant without having to design complicated negotiation protocols.

6 Skills

To effect smooth turns, motion control is expressed in terms of velocity and angular velocity. With careful tuning we found that we were able to turn with the ball held on the fingers. The basis of this move is to approach the ball slowly, and then perform a smooth turn at about 0.5 m/s. To help hold the ball, the fingers were curved and their front edges were coated with a silicon gel.

Getting the ball off the wall or out of the corners was difficult for most teams. Our approach was to approach the ball from a point at a normal to the

wall surface and flick the ball. Also, if the ball is lost, to continue around in the same direction. Localisation is important to getting this skill right.

7 Strategy

The strategy module is based on a hybrid deliberative and reactive model, with the deliberative part controlled using JACK [2]. In the deliberative part, predefined plans are triggered by *events* (i.e. certain changes detected in the environment). The plans control the reactive part by switching behaviours on and off. Otherwise, the reactive part runs independently.

8 Conclusions

RoboCup is not just about having the best kicking device. We have made great strides in improving our teams ability to get a shot at the goal. Our robots cooperated well and generally kept out of each others way. In future, we look to make better use of cooperation by passing. The ability of the robots to turn the ball around worked well but required much fine tuning. We hope to use learning techniques to make this skill more robust. Our motion control was much improved but we still need to reduce the control latency in order to cope with the velocities that these robots are capable of.

Acknowledgements

We'd like to thank the following staff members and students from various parts of RMIT University: Joseph Antony, Frank Barbaro, Daniel Bradby, Adrian Bruch, Lavindra deSilva, K Do Duy, Ferry Ferdinand, Jason Garbutt, Steven Garcia, Francis Goh, Matthew Hallsworth, James Harland, Erwin Hendriks, Vance Heredia, Marcus Holt, Antony Iorio, Sabu John, Thamanoon Kamtui, Chris Keen, Yeoh Keat Keng, Anthony Kendall, John Kneen, Deanne Koelmyer, Rob Lugton, Prem Mariappan, Attila Marton, Andrew May, Phillip Musumeci, Richard Pece, David Poutakidis, Sy Dinh Quang, Raja Riqtadar, Luke Robins, Malcolm Robins, Josie Ryan, Tony Sarhanis, Dhirendra Singh, Salim Shamsuddin, Naree Song, Jeyaprakash Srikantha, Olof Szymczak, Rahul Thomas, Laura Thomson, Thanh Trinh, Vasu Uppu, Adrian Volpe, Abhinav Vora, Michael Winikoff, Nigel Yeo, and Geoffrey Yu.

References

1. Ronald Arkin. *Behavior-Based Robotics*. MIT Press, 1998.
2. P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. JACK intelligent agents – components for intelligent agents in Java. *AgentLink Newsletter*, January 1999.
3. R. Davies. *Machine Vision : theories, algorithms, practicalities*. Academic Press, 1997.

Agilo RoboCuppers: RoboCup Team Description

Sebastian Buck, Robert Hanek, Michael Klupsch, Thorsten Schmitt

Forschungsgruppe Bildverstehen (FG BV) – Informatik IX

Technische Universität München, Germany

{buck,hanek,klupsch,schmittt}@in.tum.de

http://www9.in.tum.de/research/mobile_robots/robocup/

Abstract. This paper describes the *Agilo RoboCuppers*¹ team of the image understanding group (FG BV) at the Technische Universität München. With a team of four Pioneer 1 robots, equipped with CCD camera and a single board computer each and coordinated by a master PC outside the field we participate in the Middle Size League of the fourth international RoboCup Tournament in Melbourne 2000. We use a multi-agent based approach to represent different robots and to encapsulate concurrent tasks within the robots. A fast feature extraction based on the image processing library HALCON provides the data necessary for the on-board scene interpretation. All robot observations are fused to one single consistent view. Decision making is done on this fused data.

1 Introduction

The purpose of our RoboCup activities is to develop software components, frameworks, and tools which can be used flexibly for several tasks within different scenarios under basic conditions, similar to robot soccer. Our work is also used for teaching students in vision, machine learning, robotics and last but not least in developing large dynamic software systems. For this reason, our basic development criterion is to use inexpensive, easy extendible standard components and a standard software environment.

2 Hardware Architecture

Our RoboCup team consists mainly of four Pioneer 1 robots [1] each equipped with a single board computer. They are supported by a master PC (coach), which is also used in order to display the robot's data and states. The single board computers are mounted on the top of the robots. All robot computers are linked via a 10 Mbps radio Ethernet network [4, 5]. A master computer is located outside the soccer field and is linked to the radio Ethernet, too. It can also be used for debugging purposes, monitoring the robots' decision states and feature extraction processes. The operating system of the computers is Linux. Figure 1 demonstrates the hardware architecture.

¹ The name is derived from the Agilolfinger, which were the first Bavarian ruling dynasty in the 8th century, with Tassilo as its most famous representative.

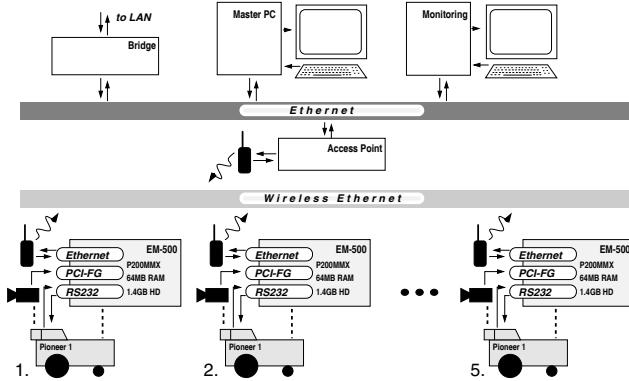


Fig. 1. Hardware architecture.

Figure 2 (a) shows one of our Pioneer 1 robots. Each of them measures 45 cm × 36 cm × 56 cm in length, width, and height and weighs about 12 kg. Inside the robot a Motorola microprocessor is in charge of controlling the drive motors, reading the position encoders, for the seven ultrasonic sonars, and for communicating with the client. In our case this is a single board computer (EM-500 from [2]) which is mounted within a box on the topside of the robot. It is equipped with a Pentium 200 MHz processor, 64 MB RAM, 2.5" hard disk, on-board Ethernet and VGA controller, and an inexpensive BT848-based [7] PCI video capture card [3]. PC and robot are connected via a standard RS232 serial port. A PAL color CCD camera is mounted on top of the robot console and linked to the S-VHS input of the video capture card. Gain, shutter time, and white balance of the camera are adjusted manually. For better ball guidance we mounted a simple concave-shaped bar in front of each robot. A kicking device enables the robot to push the ball in direction of the robot's current orientation.

3 Fundamental Software Concepts

The software architecture of our system is based on several independent modules. Software agents control the modules, they decide what to do next and are able to adapt the behavior of the modules they are in charge for according to their current goal. For this, several threads run in parallel. The modules are organized hierarchically, within the main modules basic or intermediate ones can be used. The main modules are image (sensor) analysis, robot control, information fusion, and decision making. The latter runs on the master PC outside the field, the others on the single board computers on the robots.

Beside the main modules there are some auxiliary modules (monitoring the robots' states). For the communication between different modules, we strictly distinguish between controlling and data flow. One module can control another by sending messages to the appropriate agent. Data accessed by various modules is handled in a different manner. For this, a special sequence object class was

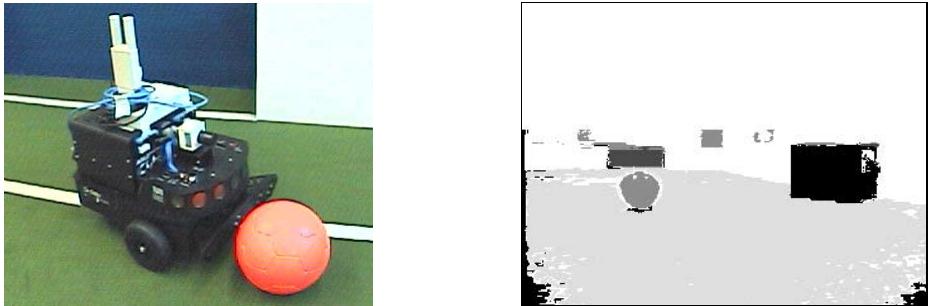


Fig. 2. (a) Odilo – one of our Pioneer 1 robots – and (b) what he perceps of the world around him.

defined. This offers a consistent concept for exchanging dynamic data between arbitrary components [11].

4 Vision

The vision module is a key part of the whole system. Given a raw video stream, the module has to recognize relevant objects in the surrounding world and provide their positions on the field to other modules. This is done with the help of the image processing library HALCON (formerly known as HORUS [9, 6]). The frame-grabber interface was extended to features for capturing gray scale images and color regions at the same time. For this we use the RGB-image data provided by the video capture card. The color regions can be achieved very fast by a three-dimensional histogram-based classifier. Gray scale images, color regions and the extracted data are provided by sequence objects as described in section 3. As a compromise between accuracy and speed we capture the images with half the PAL resolution clipping the upper 40 percent. This results in a resolution of 384×172 with a frame rate of about 12 images per second.

In general, the task of scene interpretation is a very difficult one. However, its complexity strongly depends on the context of a scene which has to be interpreted. In RoboCup, as it is defined in the present, the appearance of relevant objects is well known. For their recognition, the strictly defined constraints of color and shape are saved in the model database and can be used. These constraints are matched with the extracted image features such as color regions and line segments [10] (see Fig. 2 (b)).

Besides recognizing relevant objects with the help of the color regions, a second task of the image interpretation module is to localize the recognized objects and to perform a self-localization on the field. To localize objects we use the lowest point of the appropriate color regions over the floor in conjunction with a known camera pose relative to the robot. Self-localization is performed by matching the 3D geometric field model to the extracted line segments of the border lines and – if visible – to a goal. A sub-pixel accurate edge filter performed on the gray-scale image supplies contours from which, after removing radial distortions, straight line segments are extracted. Both, an absolute initial

localization as well as a successive refinement, compensating the error of the odometric data have been implemented.

5 Decision Making

Decision making is done on the fused world model. Choosing an appropriate action relies on this data. A number of possible actions such as *go2ball*, *shoot2goal*, *dribble*, *pass*, *cover*... are defined. For all robots and each of those actions a_i success rates $P(a_i)$ and gains $G(a_i)$ are estimated (see [8] for details). This is done by manually defined feature-based functions so far. But we are working on more sophisticated parametric learning techniques. From all promising actions ($P(a_i)$ exceeds certain threshold) the one assigned to the highest $G(a_i)$ is chosen to be executed on a robot. All known global environment information is exploited for the computation of P and G .

To execute an action a planning algorithm creates a sequence of states which leads from the actual state over to the target state. Another algorithm maps the state changes to low level robot commands using neural networks. Therefore we measured state changes of robots according to different executed low level commands.

Naturally this approach depends on a stable connection between all robots. In case of an unstable interconnection the robots may also work properly but use less good environment information and therefore can only achieve a limited degree of cooperation.

References

1. ActivMedia Robotics, <http://www.activmedia.com/robots/>
2. Lanner Electronics Inc., <http://www.lannerinc.com/>
3. Videologic Inc., http://www.videologic.com/ProductInfo/capt_pci.htm
4. RadioLAN Inc., <http://www.radiolan.com>
5. Delta Network Software GmbH, <http://www.delta-net.de>
6. MVTEC Software GmbH, <http://www.mvtec.com>
7. Brooktree: BT848 Single-Chip Video Capture Processor, <http://www.brooktree.com/intercast/bt848.html>
8. Buck S., Riedmiller M., *Learning Situation Dependent Success Rates of Actions in A RoboCup Scenario*, submitted to PRICAI2000.
9. Eckstein W., Steger C., *Architecture for Computer Vision Application Development within the HORUS System*, Electronic Imaging: 6(2), pp. 244–261, April 1997.
10. Hanek R., Schmitt T., *Vision-Based Localization and Data Fusion in a System of Cooperating Mobile Robots*, submitted to PRICAI2000.
11. Klupsch, M., *Object-Oriented Representation of Time-Varying Data Sequences in Multiagent Systems*, 4th International Conference on Information Systems and Synthesis – ISAS '98, pp. 33–40, 1998.

WinKIT

Kosei Demura¹, Nobuhiro Tachi, Noriharu Kubo, and Kenji Miwa²

¹ Matto Laboratories for Human Information Systems

² Factory for Dreams and Ideas

Kanazawa Institute of Technology

7-1 Ohgigaoka Nonoichi Ishikawa 921-8501, JAPAN

Abstract. This paper describes our research interests and technical information of our team for RoboCup-2000. Our robots have been developed to have a capability of pass-based tactics. That is, the capability of position estimations of the robots and the ball, the distinction between our team and opponent team, and passing the ball to a desired direction. To achieve the capability, robots have kicking devices and omnidirectional vision systems.

1 Introduction

The performance of robots was quite different between the F180 (small size) league and the F2000 (middle size) league in RoboCup-99. Many robots of F180 moved very quickly. Some of their average speed reached about 2 m/s. And many pass plays were observed in a real game. In contrast, many F2000 robots moved like tortoises, and some of them never moved. Few pass plays were observed only in the finals. The main reason is the difference of the vision system. F180 permits a global vision system, and F2000 does not.

Our robots for RoboCup-99 were able to move at speeds of up to 2 m/s, and their maximum acceleration speed was 4m/s². They were easily stand comparison with the F180 robots.

In the RoboCup-99¹ competition, however, their average speed was less than 0.5m/s because of our vision processing capability. Thus, in the F2000 league, we study the vision system mainly. Our current research topics are as follows:

- Object Detection
- Localization

For the robust object detection, our vision system uses the shape information besides the color information of the ball, and determines the color threshold automatically using the geometric constraints. For the robust localization, an omnidirectional sensor is equipped on the top of the robot.

¹ Our team name was Matto in RoboCup-99.

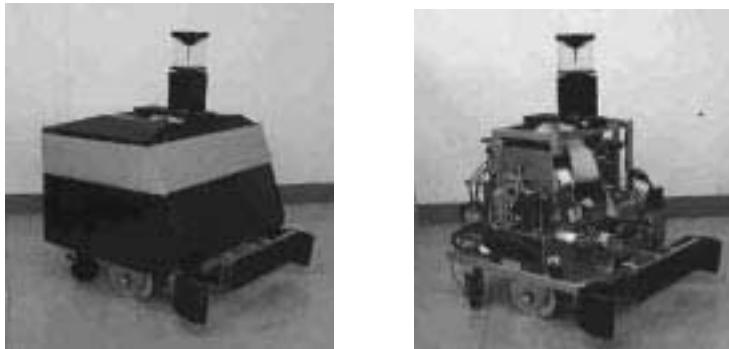


Fig. 1. The WinKIT Robot

2 Architecture

2.1 Hardware Architecture

Overview

Fig.1 shows one of our new soccer robot system that is designed to have a capability of a pass-based tactics. To realize the tactics, it has a pneumatic kicking device and an omnidirectional vision sensor for the position estimation.

It is equipped with an industrial PC, infrared position sensors, an omnidirectional vision sensor, a wireless Ether, a BT848-based PCI video capture cards, a powerful kicking device, and a fast mobile system.

Processing System

We have learned from RoboCup-99 that the reliability of the robot system is the most important. To make the robot system reliable, our processing system is an industrial computer.

Our robot system does not have any image processing boards, the high level processor needs for the image processing. Thus, our processor is a Pentium III 650MHz with 256MB RAM.

Sensor System

The vision system is the most important of all sensorial systems. To realize a robust vision system, we use omnidirectional vision sensors from ACCOWLE. They have hyperboloidal mirrors that can generate an image taken from a single view point [1]. And the infrared position sensors are used for the collision avoidance.

Specifications:

Dimension: Length 42cm x Width 42cm x Height 60cm

Weight: 23kg

Motion: Speed: 5 m/s Acceleration: 4 m/s²

Battery Power: 14 Ah

Motor: Maxon F2260 40W 24V

Gear: Maxon Planetary Gearhead GP62, Reduction 5.2:1

Processor: Pentium III 650MHz

Hard Drive: 2.5" 3.4GB

Memory: 256 MB

Wireless LAN: WaveLAN Turbo

CCD Camera: SONY EVI-330

Omnidirectional Sensor: ACCOWLE hyperboloidal mirror, middle size

2.2 Software Architecture

Vision

The vision system is one of the most important part in the F2000 league. Because the main difference between the F180 league and the F2000 league is that the global vision systems are not permitted in the F2000 league.

Limited in the vision system, the position estimation and the object recognition are very difficult compared to the F180 league. We make an effort to study the vision system.

In RoboCup-99, object detection was based on the object color. Visual information received from the CCD camera is converted to YUV from RGB. Because YUV is more stable than RGB when light conditions are changed. All colors (red, blue, yellow, green, white, black, light blue, purple) designated in RoboCup can be discriminated by threshold values of each color. The threshold values were manually determined.

This year, we also use the shape information of the ball to increase the accuracy of the ball direction and the ball distance. The method is based on the XY profile projection of the ball.

And, we have developed the method of determining the threshold values automatically [3]. The threshold values are calculated based on the physically-based approach. The position and the size of the ball on the captured image are determined by the actual ball size and the camera parameters. The threshold



Fig. 2. Left: Raw image, Right: The result of the image processing

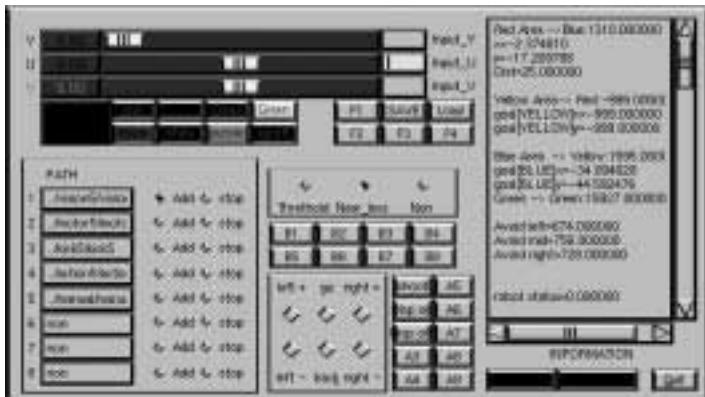


Fig.3. The vision system user interface

values are determined from the position and the size of the ball on the captured image. Fig.2 shows the result of the image processing, and Fig.3 shows our vision system user interface.

3 Conclusions

This article presents the details of our team. RoboCup-2000 is our second challenge. This year, we study the vision system mainly to accomplish the pass-based tactics, and have developed the new robots that have the kicking devices and the omnidirectional vision sensors.

References

- [1] S. K. Nayar and S. Baker, Catadioptric image formation, In Proceedings of 1997 DARPA Image Understanding Workshop, pp. 1431-1437, 1997.
- [2] N. Kubo and K. Demura. The Vision System for RoboCup. In Proceedings of ROBOMECH'00, 2P2-27-032, 2000

CMU Hammerheads Team Description

Rosemary Emery, Tucker Balch, Rande Shern, Kevin Sikorski, and Ashley Stroupe

Robotics Institute, Carnegie Mellon University
Pittsburgh, PA, 15213

{remery, trb, rande}@cs.cmu.edu, {kws, astroupe}@andrew.cmu.edu
<http://www.cs.cmu.edu/~coral/minnow>

1 Introduction

In this paper the design of the CMU Hammerhead middle-size robot soccer team is presented. The team consisted of 4 fully autonomous robots with wireless communication and color vision. The robots' color segmentation algorithm, CMVision, provided reliable, real-time ball tracking and was tolerant to changes in lighting conditions. The robots' software architecture was implemented using TeamBots, a Java-based environment for behavior based robot control, and Clay, a TeamBots module for creating motor-schema based control systems. The use of behavior-based control allowed to the robots to react quickly to the dynamic environment despite non-holonomic platforms. The team included some teamwork by sharing information about the ball's location and the states of other team members. The CMU Hammerheads made it to the quarter finals of the competition.

2 Team Development

The CMU Hammerheads were lead by Rosemary Emery. Rande Shern, Kevin Sikorski and Ashley Stroupe were the other graduate students on the team. Tucker Balch was the faculty advisor, with additional strategy advice provided by Manuela Veloso. Jim Bruce, Scott Lenser and Zia Kahn helped with the integration of CMVision with TeamBots.

3 Hardware Platform

Each CMU Hammerhead robot is identical, and is constructed from readily available commercial components. The robots are non-holonomic and composed of two main parts, a differential drive Cye robot from Probotics, Inc., and a passive trailer. Each robot covers an area of 30cm x 40cm, weighs 8kg, and costs under \$3000.00 in total. The trailer, which carries the robot's on-board computer, batteries and wireless ethernet, is free to rotate around the Cye. A color video camera is mounted on a shaft that rotates with the Cye. Wireless communication is via an 11 Mbps Lucent waveLanPC wireless Ethernet converter box.

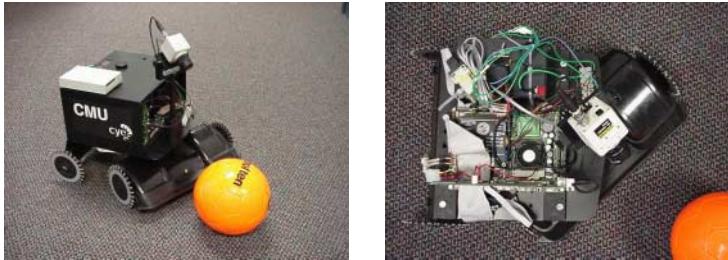


Fig. 1. A CMU Hammerhead robot and the on-board computer.

The Cye robot is a two-wheeled differential drive robot with an on-board 16 bit microcontroller with a serial interface. The Cye's innovative wheel design produces exceptional odometry on soft surfaces such as carpet. The unit's on-board computer keeps track of X, Y, and heading information for dead-reckoning and motion control. The robot's re-chargeable battery allows up to 3 hours of continuous running time.

The computer on the trailer is built from off-the-shelf components. It is a single board, P2 300 MHz Biscuit PC with 256 Megs of RAM and a 2 Gig HD. We run Linux RedHat 6.1 on the computer. A Winnov video capture card is plugged into the single PCI slot and the wireless Ethernet radio is connected via a 10-baseT connector. Power is supplied by an iron core battery and is regulated by a PC/104 DC to DC converter. This 8lb battery allows the computer to run continuously for at least 6 hours before being recharged.

3.1 High-Level Vision System

The vision system of the Minnow robots played an important role in the success of the team. Because the robots are able to reliably detect targets in real-time, they are able to track the ball and acquire it with high accuracy. The color filtering was remarkably tolerant to changes in lighting conditions during the competition.

The vision system in the Minnow robot uses a color camera with a 103 degree horizontal field of view lens. The camera/lens pairs were run through Flatfish - a camera calibration routine [5] that enables distortion to be removed from the images. During run time, images are captured and segmented using CMVision [4], a color blob detection package. This package is able to segment the image into regions describing up to 32 colours at 30Hz. When a request for a target location is made, the largest blob of the target's color is found. The pixel in that blob that most likely represents the point of contact between the object and the ground is then extracted and a ray constructed from the camera to this pixel in the undistorted image. Once the point of intersection of this ray with the ground plane is known, its coordinates in 3-space are returned to the control system. Therefore, at any given time, the entire image does not have to be de-warped, only the pixels of interest. This keeps the image processing fast.

4 Team Behaviors

The robots are controlled using the Clay architecture of Teambots [2, 3], a Java-based collection of application programs and packages for multiagent robots. Clay is a group of Java classes that can be easily combined into motor-schema based control systems, generally represented by finite-state machines. In this approach, each state corresponds to a suite of activated behaviors for accomplishing that state of the task [1]. Transitions between states are initiated by real-time perception. The activated behaviors create vectors to represent the desired trajectory of the robot. Clay is therefore similar to potential field navigation; however, it differs in that only the vector at the robot's current position is calculated, not the entire vector field. The resultant control systems are tolerant to use with non-holonomic robots. The use of a behavior-based approach allows the control system to quickly locate and track the ball without having to explicitly deal with the position of the trailer. A planning approach would require the control system to plan with non-holonomic constraints which is computationally difficult and time-consuming and, therefore, inappropriate to the highly dynamic environment of robotic soccer.

Due to the non-holonomic nature of the robots, our goalie cannot easily move back and forth in front of the goal. Instead, we implemented an aggressive goalie that, if the ball comes close to it, lines itself up behind the ball and then dribbles the ball to the half-way point of the field before returning back to the goal box. For all four of the robots (the goalie, halfback and two forwards), lining up at a homebase location is accomplished by a Going-Home behavior suite that swirls the robot to a position behind its homebase and then drives it forward to the homebase. All robots return to their homebase position if they cannot see the ball. This is done because the referee will remove the ball and replace it on a penalty location should a player commit a foul or no progress be made. Thus, if a forward cannot see the ball, moving to the center of the field and re-facing the opponent's goal allows it to see the ball in any of these penalty locations.

4.1 Teamwork

Teamwork in the CMU Hammerheads is limited to the sharing of information. The robots communicate to each other the location of the ball and their current state. Communication between multiple robots can improve the performance of the team in several ways. Communication of collected data can expand the effective visual range of individual team members and increase the accuracy of position estimates. The greater coverage of the environment and greater certainty in measurements possible with communication allows for more appropriate responses to rapidly changing situations.

The communication philosophy of our robot team was minimization. A few types of flexible messages were designed. The ball information, gathered via object position messages, is used to help direct the robot's attention before it performs a blind search for the ball; the robot will first look where its teammates see the ball and, if it cannot see it there, will go into a searching behavior.

For our robots to effectively share information about the ball, gaussian distributions of the robots' beliefs about the ball's location are communicated. The ball is identified through the vision system and its position located. The probability distribution of its location is then calculated and its mean and covariance matrix are converted to global coordinates. These parameters are transmitted to teammates and any received transmissions multiplied with existing distributions [6]. Finally, the most probable location of the ball is identified and transformed into the robot's local coordinate frame.

The state of other players, gathered through behaviour messages, is used by the forwards to help them avoid a teammate who already has control of the ball. Thus, if the halfback or a forward broadcasts that it has the ball, the other robots will avoid that player, minimizing interference. Control messages are used to start and stop the team and to restart robots removed from the field.

4.2 Conclusions

We were very happy with the performance of the CMU Hammerhead team during Robocup 2000. The team played six round robin games of which it won two, tied one, and lost three, scoring nine goals over the six games and sustaining ten goals. This performance allowed the team to progress to the quarter finals. The players were able to reliably sense the ball but were hindered by the speed at which they could acquire the ball compared to other, holonomic or virtually holonomic, players. Highlights of the games included the only aggressive goalie of the middle-sized competition and goals in which a forward avoided opponents while maintaining control of the ball.

References

1. R. Arkin and D. MacKenzie. Temporal coordination of perceptual algorithms for mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 10(33):276–286, June 1994.
2. T. Balch. Clay: Integrating motor schemas and reinforcement learning. Technical Report GIT-CC-97-11, College of Computing, Georgia Institute of Technology, March 1997.
3. T. Balch. *Behavioral Diversity in Learning Robot Teams*. PhD thesis, College of Computing, Georgia Institute of Technology, December 1998.
4. J. Bruce, T. Balch, and M. Veloso. Fast and cheap color vision on commodity hardware. In *Workshop on Interactive Robotics and Entertainment*, pages 11–15, Pittsburgh, PA, April 2000.
5. H. Moravec. Robot spatial perception by stereoscopic vision and 3D evidence grids. Technical Report CMU-RI-TR-96-34, Robotics Institute, Carnegie Mellon University, September 1996.
6. R. C. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, June 1994.

GMD-Robots

Ansgar Bredenfeld, Thomas Christaller, Horst Guenther, Jörg Hermes,
Giovanni Indiveri, Herbert Jaeger, Hans-Ulrich Kobialka,
Paul-Gerhard Plöger, Peter Schoell, Andrea Siegberg

*GMD - Institute for Autonomous intelligent Systems (AiS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany*

e-mail: bredenfeld@gmd.de

1 Introduction

The overall research goal of GMD's RoboCup team is to increase both, (1) the speed of mobile robots acting as team in a dynamic environment and (2) the speed of design for behavior-based robot control. Therefore, we started in 1998 to develop a proprietary fast robot platform and the integrated Dual Dynamics Design Environment.

In Melbourne, our middle-size league team achieved the second rank during round-robin. In the Quarter Finals our Robots were beaten by Sharif CE, the World Champion 1999 and current European Champion 2000.

2 Team

Ansgar Bredenfeld (DD-Designer, team leader)
Horst Guenther (technical support)
Jörg Hermes (ball guidance)
Giovanni Indiveri (control engineering of goalie)
Herbert Jaeger (Dual Dynamics)
Hans-Ulrich Kobialka (beTee)
Paul-Gerhard Plöger (Hardware)
Peter Schoell (DDSim)
Michael Severin (kicking device)
Andrea Siegberg (technical support)
Web Page: <http://ais.gmd.de/BE>

3 Hardware Platform

Our robot hardware is a custom-built platform. We use two 20 Watt, high-quality Maxon motors that are mounted on a very solid, mill-cut aluminium frame. A piezo-gyroscope senses the angular velocity of the robot. Obstacle avoidance is supported by four infrared-based range detectors and standard bumper ring sensors. Our robots kick the ball with a pneumatic device. The camera of the Newton Lab's Cognachrome vision system is mounted on a 360 degree panning unit.

The computer system of the robot consists of a Pentium PC notebook connected to two C167 micro controller subsystems for sensor drivers and actuator interfaces. The

communication between the PC and the micro-controllers is via CAN bus and between the PC and other robots or a remote-logging PC is via WaveLAN.

4 Software Architecture

Our approach to robot programming is based on Dual Dynamics (DD) [1], a mathematical model for robot behaviors which we developed. It integrates central aspects of a behavior-based approach, robust control, and a dynamical systems representation of actions and goals. Robot behaviors are specified by differential equations. In the whole they build a dynamical system consisting of subsystems which interact through specific coupling and bifurcation-induction mechanisms. Behaviors are organized in levels where higher levels have a larger time scale than lower levels. Since the activation of behaviors (activation dynamics) is separated from their actuator control laws (target dynamics), we named our approach "Dual Dynamics". An important feature of DD is that it allows for robust and smooth changes between different behavior modes, which results in very reactive, fast and natural motions of the robots.

5 World Model

Dual dynamics behavior systems use symbolic sensors to represent the locally perceived environment of the robot. We do not maintain a global world model shared by all robots. Self localisation of the robot is performed based on odometry and gyroscope data. Since these data is noisy and subject to be disturbed, we compensate odometry errors by improving the self-localization of our robots using *weighted* Monte Carlo sampling [2]. This approach re-adjusts the pose of the robot using the bearing of the goals measured by our vision system.

6 Communication

Our team communication mechanism allows to establish real-time point-to-point connections between all robots of a team. During robot software development all behaviors are kept in a common central source code repository (CVS). Thus we can determine the data flow network within a team of robots in advance. This allows to communicate shared variables between the robots efficiently without any protocol overhead. The variables shared among different behaviour systems are simply tagged in the specification tool DD-Designer. The technical implementation of the team communication mechanism is based on TCP/IP and uses WaveLAN.

7 Skills

Our vision system relies on the well-known Newton Lab's Cognachrome system for ball and goal detection. Since it is mounted on a 360 degree panning unit, we are able to perform "radar-like" optical scans of the robots surrounding. The angle encoder of the panning unit delivers a precise relative angle of each camera picture, which is used in the target dynamics of our behaviors.

Kicking is done with a pneumatic device which is mechanically integrated in the ball guidance of the robot. Kicking is triggered by the behavior system dependent on the pose and the mode of the robot. A neural network is used to anticipate whether the ball will be lost in near future.

The goalie has a slightly different sensor configuration (infrared range detectors pointing to the back of the goal) and of course a specific behavior system. It is essentially a two-dimensional controller, which maintains a fixed distance to the back of the goal and a certain angle to the visible ball. If the robot should be hit by opponent robots thus losing its position in front of the goal, a homing behavior is activated in order to recover the correct position in front of the goal and to re-start the keep goal behavior.

8 Special Team Features

The successful design of robot software requires means to specify, implement and simulate as well as to run and debug the robot software in real-time on a team of physical robots. The integrated Dual Dynamics design environment [2][3] we develop allows to specify DD-models on a high-level of abstraction and to synthesize all code artifacts required to make these models operative in practice: a simulation model, a control program for a real robot, a team communication layer and set-up parameters for real-time monitoring and tracing. In this environment the following steps are performed iteratively in order to design a behavior system for a robot.

Specify. The specification and code generation tool *DD-Designer* comprises a graphical editor to enter the specification of a DD-model in terms of sensors, actors, sensor filters and behaviors. Sensor filters and behaviors are further detailed using the equation editor of DD-Designer. We use multi-target code generation to refine the DD-model to code artifacts required by the simulator, the robot and the real-time monitoring tool. DD-Designer continuously evolved from a first shot prototype [4] to a full-fledged design tool. This development process was an ideal test case to investigate the evolution of software prototypes in design environments [5].

Simulate. The Java simulator *DDSim* is specifically tailored to simulate a team of robots with different behavior systems. The simulator is capable of simulating the whole sensor equipment of the robots including the vision system. The implementation of the behavior system, i.e. the robot control program, is a Java class generated by DD-Designer.

Run. The code for the real robot implements the behavior system in C/C++. This code is directly derived from the high-level specification edited in DD-Designer. Since both artifacts, simulation model (Java) and robot control program (C++), are derived from the same specification, we avoid all problems that occur if a migration from a simulation model to a robot control program has to be performed manually.

Test/Analyse. The real-time trace tool *beTee* allows to capture and analyse internal variable states of an implemented behavior system in real-time [6]. The configuration of beTee is carried out by a Java class generated by DD-Designer. It contains a dictionary of all variables including their dependencies.

9 Future Work

Future work will further focus on our main research goals.

In order to further increase the speed of our mobile robots, the sensor information flow needs to be raised. This will be achieved by adding optical flow sensors to the robot. We will investigate these sensors in close connection with the behavior systems of the robot in order to make them more reactive even at higher speeds than that we already achieved. In addition, we focus on extensions to the Dual Dynamics scheme and on "Observable Operator Models" [7]. Both emphasize the dynamical systems nature of behaviors, with OOMs additionally capturing the stochastic nature of a robot's experience and acting.

In order to further increase the speed of our behavior development process, we have to further narrow the still existing gap between simulation and the real robot. At present, we are able to simulate the robot with its complete sensor equipment in our simulator DDSim. Therefore, we are able to design functional correct behavior systems using simulation only. Nevertheless, parameter tuning is left to real experiments with real robots on the field. We will investigate approaches allowing to adapt the sensor models in the simulator to the precise sensor behavior as measured on individual robots. This will further decrease the number of time-consuming experiments with the robots on the field.

In 2001, it is planned to participate in the 1st GermanOpen in Paderborn and in the 5th RoboCup World Championship in Seattle. Results achieved up to these events will be demonstrated on the GMD-robots.

References

- [1] H. Jaeger and Th. Christaller. 'Dual dynamics: Designing behavior systems for autonomous robots', *Artificial Life and Robotics*, 2:108-112, 1998
- [2] A. Bredenfeld, T. Christaller, H. Jaeger, H.-U. Kobialka, P. Schoell, 'Robot Behavior Design Using Dual Dynamics', *GMD Report 117*, accepted for KI-Zeitschrift, 2000
- [3] A. Bredenfeld, T. Christaller, W. Goehring, H. Guenther; H. Jaeger, H.-U. Kobialka, P. Ploeger, P. Schoell, A. Siegberg, A. Streit, C. Verbeek, J. Wilberg, 'Behavior engineering with "dual dynamics" models and design tools', in *Veloso, M.; Pagello, E.; Kitano, H. (Eds.): RoboCup-99: Robot Soccer World Cup III*, Springer, LNCS 1856, 2000
- [4] A. Bredenfeld, 'Co-design tool construction using APICES', in *Proc. of the 7th IEEE Int. Workshop on Hardware/Software Co-Design (CODES'99)*, 1999.
- [5] A. Bredenfeld, 'Integration and Evolution of Model-Based Prototypes', *Proc. of the 11th IEEE International Workshop on Rapid System Prototyping (RSP 2000)*, Paris, France, June 21-23, 2000
- [6] H.-U. Kobialka and P. Schoell, 'Quality Management for Mobile Robot Development', *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, Venice, Italy, July 25-27, 2000
- [7] H. Jaeger, 'Observable operator models for discrete stochastic time series', *Neural Computation* 12(6), 1371-1398, 2000

A goal keeper for middle size Robocup

M. Jamzad¹, A. Foroughnassiraei², T. Hadji Aaghai¹, V.S. Mirrokni¹,
R. Ghorbani², A. Heydar Noori¹, M. Kazemi¹, H. Chitsaz¹, F. Mobasser¹,
M. Ebraahimi Moghaddam¹, M. Gudarzi¹, and N. Ghaffarzadegan²

¹ Computer Engineering Department
Sharif University of Technology
Tehran, Iran.

jamzad@sina.sharif.ac.ir
<http://www.sharif.ac.ir/~ceinfo>

² Mechanical Engineering Department
Sharif University of Technology
Tehran, Iran.
<http://www.sharif.ac.ir/~mechinfo>

Abstract. In real soccer goal keeper has a completely different behavior than the players. Thus we designed our goal keeper with some basic ideas taken from real soccer, which is moving mainly in goal area and using arms to take the balls. Our goal keeper has a moving mechanism based on 8 motors which enables it to move forward/backward, straight left/right and rotate around its geometrical center, and a sliding arm which moves toward the direction of ball faster than the robot body. Using 3 CCD cameras in front and rear left and right provides the goal keeper with a view of about 210 degrees.

1 Introduction

Since the basic design of our player robots remained unchanged from that of previous year, [1], we decided to focus on the goal keeper. As in real soccer we think a goal keeper robot should have a completely different mission from the player robot. It should be able to move in any direction and use its hands to get the ball and also use its hands and feet to kick the ball. With these mechanisms in mind we designed and constructed our goal keeper in such a way that it provided us with all necessary movements for the robot body and also a sliding arm with a special design to move left or right much faster than the robot body for getting the ball and also kicking it. Since in general, mechanical movements are relatively slow in robots, thus if the goal keeper makes angular movements to look for the ball, not only it will be time consuming but also due to inexact movements after some short period of time the goal keeper might find itself in an unsafe position. Therefore, to cover a wide angle of view, it is better to use multiple cameras than angular movements of robot body itself.

2 Goal Keeper Motion Mechanism

Because the goal keeper should keep the goal, it seems that fast and deviation less horizontal movement in front of goal area is a great advantage for it. Therefore, in order to guarantee a nearly perfect horizontal movement for the goal keeper, 4 drive units are installed in the robot (the castor wheel which is used in our player robots [1] has been eliminated because it causes deviation in the robot movements). However, even using 4 drive units, in practice the robot will be displaced after some movements, thus it should have the ability to adjust itself when displaced. Horizontal movements and self adjustment can be done by a combination of the following three basic movements:

1. Move forward and backward (Fig. 1-a).
2. Rotate around its geometrical center (Fig. 1-b).
3. Move straight towards left and right (Fig. 1-c).

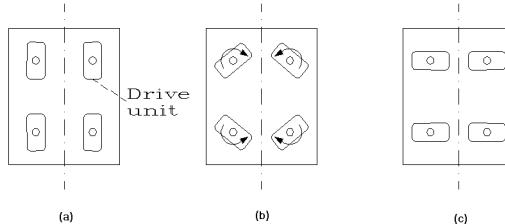


Fig. 1. Goal keeper drive units

In order for the robot to perform these movements, 4 drive units and two steer units are installed in the robot. One steer unit rotates two front drive units round their vertical axes simultaneously in opposite directions and the other steer unit does the same on two rear drive units. The drive units wheel has a diameter of 8 Cm and a gearbox of 1/15 ratio. Measurement of rotation angle for drive units are done by encoders installed on steer unit motor shafts. The equations to calculate velocity vectors and angles are given in [2].

To minimize the robot body movement, its adjustment movements and also increase goal keeper performance, we installed a fast moving sliding arm on it. This arm can slide in left or right direction before the robot body itself moves in these directions. In practice it moves much faster than the robot body and is used to get the ball and kick it from either left or right sides. This arm can slide to its leftmost or rightmost position within less than 0.1 of a second. Considering the front body size of goal keeper which is 23 Cm, and the arm size is 45 Cm (when fully extended in one direction), the robot can cover 68 Cm which is approximately 1/3 of the goal area, within less than 0.1 of a second. Compared to goal keeper maximum speed which is 75 Cm/sec, this arm gives a good protection of goal area from very fast moving balls.

Sliding arm movement is carried out by a rack and pinion mechanism, as seen in Fig. 2. To control the amount of arm sliding, an encoder is mounted on

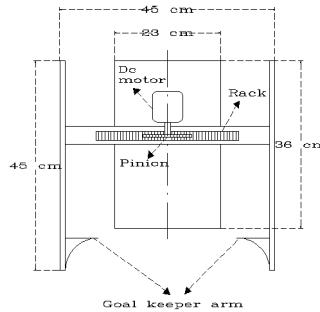


Fig. 2. Sliding arm movement

the shaft of pinion motor. It is necessary to fix the arm when goal keeper is in a stuck situation with other robots. This is done by using a solenoid which can lock the arm in its present position. In addition, a kicker installed in front of the robot, uses a solenoid with controllable kicking power. The power of kicking is controlled by duration of 24 DC voltage applied to the solenoid.

2.1 Goal Keeper Vision System

To cover a wide range of view for the robots, we used multiple cameras installed in front, rear left side and rear right side. We used three Computar CCD camera Model CY-200 with a f2.8 wide lens each covering about 75 degrees. Therefore, these 3 cameras give a view of about 210 degrees, for the goal keeper. We use a commercially available standard analog multiplexer CMOS 4041 which has 8 inputs with 3 address lines. It can switch between input addresses in 15 nano seconds. The software of the vision system consists of three main parts, real time object recognition, motion control and decision making. Software which is based on deterministic algorithms is written in C++ using DJGPP (DJGun Plus Plus) compiler in MS/DOS[3].

The object oriented design of software has 4 classes covering functions for working with frame grabber, machine vision system, interface between software and hardware and finally all robot playing methods and algorithms. For color classification we used HSI [4] color model . In practice we replaced the "I" component of HSI with "Y" component of YIQ color model [4] which gave much better results. The reason for this replacement was that, the "Y" component resembled a real monochrome image where the "I" component gave a faded unrealistic monochrome view of the scene. So we call our color model HSY.

2.2 Goal Keeper Basic Algorithm

Initially the goalkeeper is located in middle of goal area. It moves in horizontal direction towards left or right following the ball, but this horizontal movement is

limited until certain position which is the goal left most and right most positions. However, during this horizontal move, there will be some angular displacement in robot body related to the horizontal white line in front of the goal area. In such cases the robot adjusts itself in the proper horizontal position by appropriate movement of its drive units and the steer units.

The goal keeper tries to locate the ball right in the middle of its front camera. However, when the ball moves diagonally, first the robot moves its sliding arm in that direction, and then moves its body, until it reaches to the displacement limits. For example, if the ball is seen by the left rear camera, the algorithm will keep on using only that camera and no multiplexing is done. The robot will stay to its left most limit position, but if the ball comes near, it will use its sliding arm to kick the ball. The same approach is done on the right side. The front kicker is used when the ball comes within its kicking area.

3 Conclusion

In Robocup, a goal keeper misses the ball, basically in two situations, not seeing the ball and seeing the ball but not being able to move fast enough to catch it. To solve the first problem we think the best approach is to have the widest view possible from sides and front, process the scene and find the ball at least in a commercial frame grabber rate which is 1/25 of a second. That is why we used 3 fixed position wide angle CCD cameras linked to a multiplexer and one frame grabber which can cover about 210 degrees. Our fast object finding algorithm gives us a near real time rate for ball detection. In this approach with almost no mechanical movement, the goal keeper can find the ball position. Our approach for the second problem was to design a sliding arm which can move toward the direction of ball much faster than the robot body itself. In addition a kicker in front acts as a leg which shoots the ball when necessary.

One main problem with small front size of our robot (23 Cm) was that we had to make the robot tall enough (58 Cm) to install all equipment in it. This caused some instability in robot body movements, specially in fast movements. However, by using special components to make robot short enough, we can reduce or eliminate this problem. One disadvantage of this goal keeper is that it does not communicate with our player robots. Our team Sharif CE took the 3rd place in middle size Robocup 2000.

References

1. Manuela Veloso, Enrico Pagello, Hiraoki Kitano, *it RoboCup-99: Robot Soccer, World Cup III*, Springer, 2000.
2. Meriam, J.L., *Dynamics*, John Wiley, 1993.
3. M.Jamzad, A.Foroughnassiraei, et al, ARVAND, A Soccer Played Robot, *AI Magazine*, Fall 2000.
4. Gonzalez, R.C., and Woods, R.E., *Digital Image Processing*, Addison-Wesley, 1993.

The Dutch Team

Pieter Jonker¹, Will van Geest², Frans Groen³

¹ Applied Physics Department, Delft University of Technology

² Electrical Engineering Department, Delft University of Technology

³ Computer Science Department, University of Amsterdam

pieter@ph.tn.tudelft.nl

Abstract. This paper describes the layered and modular hard- and software architecture of autonomously soccer playing robots used in The Dutch Team. The schemes of the hard and software architectures are presented and the software functional modules are described with their relations and interfaces. With this software architecture we attempted to merge the distributed computing aspect of real-world intelligent autonomous robots, with the client-server approach of the soccer simulator.

1 Introduction

Real-world multi agents will play an important role in the near future. There are many application areas, such as cleaning, public safety, pollution detection, fire fighting, traffic control, etc. where these agents may become important. Real world agents should be able to co-operate together, coming up with an optimal sensing and action strategy, which can be adapted based on the current situation. Using multiple systems with local intelligence and shared communication makes such a multi-agents system robust in case of erroneous perception or malfunctioning of one of the systems. Research on these items is performed in the context of the Autonomous Interacting Robotics (AIR) project by groups from the universities of Amsterdam (UvA, VU), Delft (TUD) and Utrecht (UU) in The Netherlands. To provide a standard problem the robot soccer game was chosen.

This paper is devoted to the hard and software architecture of soccer playing autonomous robots for the “Dutch Team” of the co-operating universities.

2 Hardware Architecture

The type of robot that will mainly be used (another type of robots used in the Pioneer) in our project is the Nomad Scout robot [1]. See Figure 1. It is a mobile robot with vision system, 16 ultrasonic sonar sensors, a tactile bumper ring, odometry sensors, a control processor for low-level tasks and an onboard industrial Pentium PC for high-level tasks. The effective range of the ultrasonic sensors is from 15 cm to 6.5 m. Our hardware architecture is drawn in Figure 2.



Specifications	
Diameter:	41 cm.
Height:	35 cm.
Weight:	25 kg. (incl. batteries)
Payload:	5 kg.
Battery Power:	300 watt-hour
2 wheel differential drive @ geometric center	
Omnidirectional motion	
Ground Clearance:	1.5 cm
Speed:	1.0 m/sec
Acceleration:	2 m/s ²
Encoder Resolution:	
Translation:	756 counts/cm
Rotation:	230 counts/degree

Fig. 1. A typical robot of the Dutch Team, without top cover, based on a Nomad Scout

The high-level processor is a Pentium II 233MHz, 64MB, 4GB, on an Advantech Multi-media Single Board Computer PCM5862 [2]. It communicates with the low-level processor, a MC68332, through a serial port. Additionally, a TMS320C14 DSP is responsible for high-bandwidth motor control at 2 KHz control rates. The robot has a differential drive system, with two independent drive motors. It can operate about one hour between charges on removable batteries if all on board systems are running.

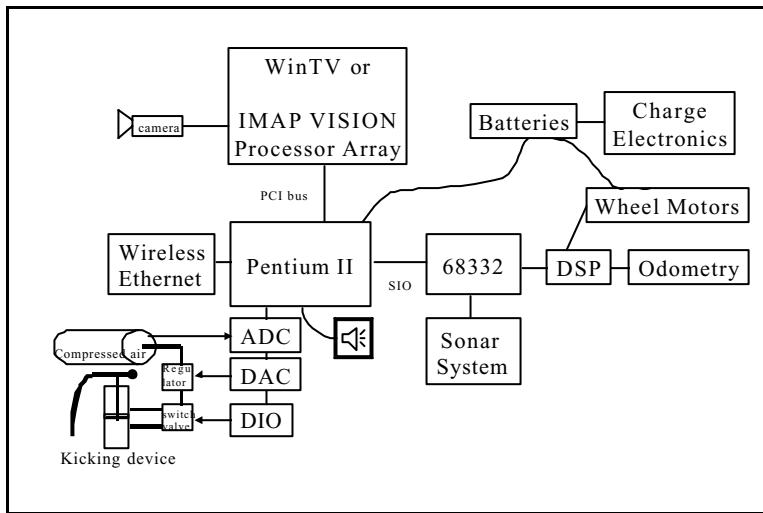


Fig. 2. The Hardware Architecture

For communication with other robots, a wireless Ethernet system is used, the BreezeCom SA-10 PRO [3], with a data rate of max. 3 Mbps, giving the robots an action radius of up to 1 Km. It uses the 2.4GHz ISM band and is compliant with IEEE 802.11. An AX10410 AD/DA/DIO board [4] is used to control a ball kicking mechanism. This mechanism is a pneumatic device, of

which the pressure and hence the kicking and catching force can be controlled by a FESTO proportional pressure regulator MPPE3-1/8-6/010B [5] through the DA converter. The pressure in the air container can be read out with the AD converter. The container is good for about 50 full force kicks. The robots run RedHat Linux 6.2. The standard robots are equipped with a WinTV PCI framegrabber card. In this case, the Pentium will need to do the image processing tasks. As tracking the ball, team-mates and competitors was one of the most difficult tasks in past RoboCup matches, we will equip at least one of the robots (the keeper) with an IMAP-VISION system from NEC's Incubation Center [6]. The cameras used are Chugai YC02B cameras [7] with a Santec TC2814M-1/3, fixed zoom manual iris lenses, with diameter 1/3", focus 2.8 mm, aperture 1.2, opening angle 94° (!).

The IMAP-VISION system is a Linear Processor Array with 256, 8 bits data processors in SIMD mode, colour framegrabbing hardware and a RISC control processor on a PCI board. It is a parallel architecture specially made for real-time image processing, where a single column of an image is mapped onto one data processor. The system is programmed with 1DC, a version of C extended for parallel data processing. See Figure 3



Fig. 3.. The IMAP-Vision System board.

3. Software Architecture

The implemented software architecture of the soccer robots is shown in Figure 4. It shows three identical robots connected to each other via a communication module. The architecture has three layers. The basic layer consists of virtual devices, the second layer of basic skills of the robot and the top layer consists of the mission strategy module. The software of the middle layer contains the intelligent basic skills of the system. Within the top level, decisions can be made to grossly change the character of the game. Normal interface function calls from modules higher in the hierarchy to modules lower in the hierarchy are implemented as strings for the task queue of the lower module. In the implementation of the modules triggers and subscription mechanisms are implemented using the same message passing techniques. For a trigger, a lower module deposits a message in the queue of a module higher in the hierarchy. Higher modules can subscribe to triggers or measurements of lower modules. This technique can even be used between robots in the team using the wireless Ethernet. The modules are:

- Motion Module:** Controls the robot movement using odometer, touch and acoustic sensors.
- Kick Module:** Controls the kicking, catching and holding (goalkeeper) of the ball.
- Sound Module:** Controls the shouting of messages to entertain the audience.
- Vision Module:** Tracks objects in the camera's field of view
- Communication Module:** For sending and receiving messages over the wireless Ethernet.

World Knowledge Module: Autonomously maintains a consistent view on the world, in accordance with the other robots

Player Skills Module: Controls autonomous skilled ball manipulation.

Team Skills Module: Keep the position in the team.

Mission Manager Module: Autonomously fulfil the mission and role as communicated by the coach, while listening to the referee. A decision tree with priority / confidence is used [8].

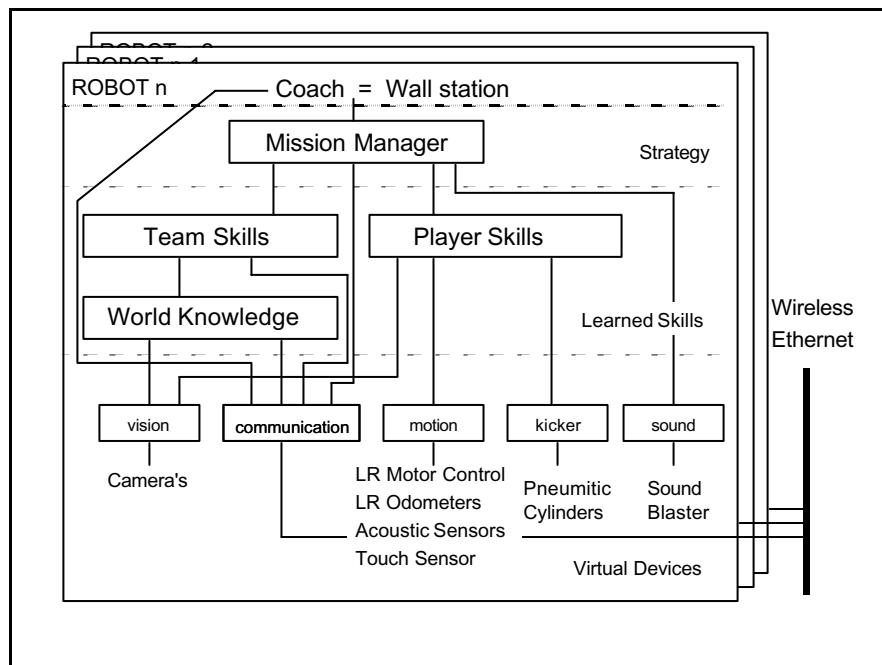


Fig. 4. Software architecture of the autonomous soccer playing robots

This work was partially sponsored by the Dutch Foundation for Pure Research NWO.

- [1] http://www.robots.com/nsuperscout.htm
 - [2] http://www.advantech-usa.com/prselect/index.htm
 - [3] http://www.breezecom.com
 - [4] http://www.axiomtek.com/Products
 - [5] http://www.festo.com
 - [6] Y. Fujita et. al. 'A 10 GIPS SIMD Processor for PC based Real-Time Vision Applications, Architecture, Algorithm Implementation and Language Support', Proc. of the IEEE int. Workshop on Computer Architectures for Machine Perception (CAMP'97), pp22-32, Cambridge, MA, USA, 1997
 - [7] http://www.chugai.com/cctv/cam
 - [8] J. Lubbers, R.R. Spaans, E.P.M. Corten, F.C.A. Groen, 'AIACS: A Robotic Soccer Team Using the Priority/Confidence Model', in 'Proceedings Xth Netherlands/Belgium conference on AI', 19-19 November 1998, p. 127-135.

The RoboCup-NAIST

T. Nakamura
M. Oohara

H. Takeda
T. Yamamoto

T. Terada
M. Takeda

Graduate School of Information Science, Nara Institute of Science and Technology
Takayama-cho 8916-5, Ikoma, Nara 630-0101, Japan

1 Introduction

Through robotic soccer issue, we focus on “**perception**” and “**situation and behavior**” problem among RoboCup physical agent challenges [1]. So far, we have implemented some behaviors for playing soccer by combining four primitive processes (motor control, camera control, vision, and behavior generation processes)[2]. Such behaviors were not sophisticated very much because they were fully implemented by the human programmer. In order to improve the performance of such behaviors, we have applied a kind of learning algorithm during off/on-line skill development phase. For example, to acquire purposive behavior for a goalie, we have developed a robot learning method based on system identification approach. We also have developed the vision system with on-line visual learning function [3]. This vision system can adapt to the change of lighting condition in realtime.

This year, we constructed a new robot equipped with an omnidirectional camera in addition to an active vision system so as to enlarge view of our soccer robot. Using this omnidirectional camera, our robot could estimate its location in the soccer field. However, such result is not used for accomplishing cooperative tasks between robots.

In the RoboCup00 competition, our team had 7 games in roundrobin. We won 2 games and lost 5 games. Finally, our team could not reach the quaterfinals.

2 Team Development

Team Leader: Takayuki Nakamura

Team Members:

- Hideaki Takeda, Associate Professor, who attended the RoboCup since 1998.
- Kazunori Terada, Ph.D candidate student, who attended it since 1998.
- Masamichi Oohara, Master course student, who attended it for the first time.
- Tetsuya Yamamoto, Master course student, who attended it for the first time.
- Masanori Takeda, Master course student, who attended it for the first time.

Web page <http://robotics.aist-nara.ac.jp>

3 Robots

We have developed a compact multi-sensor based mobile robot for robotic soccer as shown in **Fig.1**. As a controller of the robot, we have chosen to use a Libretto 100 (Toshiba) which is small and light-weight PC. We utilize a wireless LAN PC card (WaveLAN(AT&T)) for communication on our soccer robot.

Motor control system

A motor control system is used for driving two DC motors and is actually an interface board between a portable PC and motors on the chassis of our soccer robot. This control board is plugged into a parallel port on the portable PC. The motor speed is controlled by PWM. To generate PWM pulses, we use a PIC16C87 microcontroller. The motor control command is actually 8 bits binary commands for one motor. This board can receive control commands from the portable PC and generate PWM signals to right and left motors.

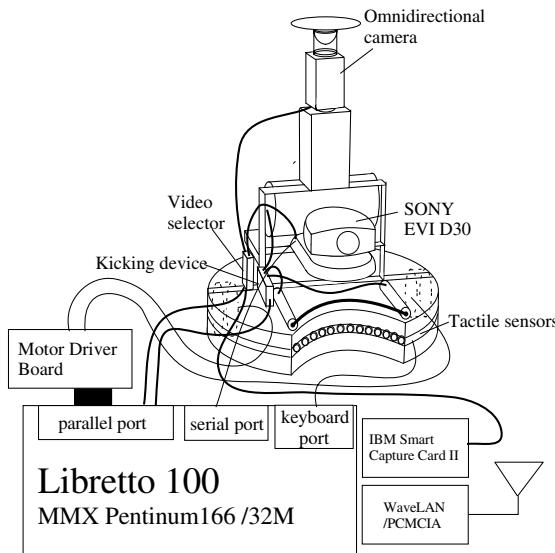


Fig. 1. Our soccer robot.

Kicking device control system

Our kicking device consists of an air tank, an air cylinder and a solenoid valve. We make an interface board between a portable PC and a relay for controlling this solenoid valve. The control command is actually 1 bit binary command. This control board is also plugged into a parallel port on the portable PC.

4 Perception

Tactile sensing system

We constructed a cheap tactile sensing system [2] by remodeling a keyboard which is usually used as an input device for PC. A keyboard consists of a set of tactile sensors each of which is a ON/OFF switch called a key. If a key is pressed, the switch is ON. If not, the switch is OFF. By using these sensors which are set around the body of soccer robot, a tactile sensing system can detect contact with the other objects such as a ball, teammates, opponents and a wall.

Visual sensing system

We use an active vision system and an omnidirectional camera system. As an active vision system, we have chosen a color CCD camera (SONY EVI D30, hereafter EVI-D30) which has a motorized pan-tilt unit. An omnidirectional camera system consists of a hyperbolic mirror and a color CCD camera of which optical axis is aligned with the vertical axis of the mirror. In order to capture two images from both vision systems by one video capture PCMCIA cards (IBM Smart Capture Card II, hereafter SCCII), we make an video selector device which can be controlled through the parallel port from a portable PC.

Vision Module

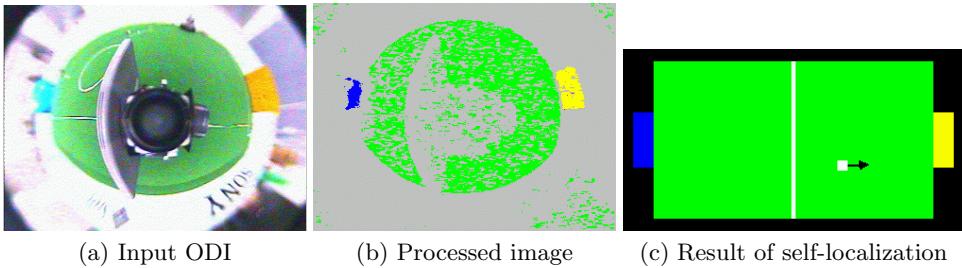
The vision module provides some information about the ball, goal and teammates in the image. The teammate is recognized by a colored marker on each robot. The vision module provides the area of the targets(ball, goal and a colored marker), the coordinates of their center and the both maximum and minimum horizontal coordinates of the goal and so on.

In the omnidirectional image (hereafter, ODI), the vision module also provides observed directions of the targets. Based on such observed direction of the targets (goals) whose location is known in advance, the vision module also estimates the location of the robot in the soccer field.

5 World Model

In order to control our hardware systems, we use a shared memory and 5 software components which are the motor controller, camera controller, tactile sensor module, vision module and behavior generator. All software components read and write the same shared memory in order to acquire and give the states of our hardware systems. Using this shared memory, they can communicate each other asynchronously. For example, the behavior generator takes the state of camera, vision, tactile and motor in the shared memory as input vectors. Then, it combines these information with programmer's knowledge and decides the robot's action at next time step. Finally, it writes the motor command for the motor controller on the shared memory.

Using the ODI and prior knowledges of the landmarks, our robot can estimate its location and orientation. **Fig.2** (a), (b) and (c) shows the input ODI, the processed ODI and the result of self-localization, respectively. In this case, own and opponent goals are treated as two landmarks. The distance between these two goals are known in advance. In **Fig.2** (c), a white rectangle and a black arrow shows the robot's position and its orientation, respectively. As shown in this figure, the result of self-localization is almost correct. In this experiment, the average error of positions and orientations is $(\bar{\Delta}X, \bar{\Delta}Y, \bar{\Delta}\theta) = (174(mm), 208(mm), 4(^{\circ}))$, where $(\bar{\Delta}X, \bar{\Delta}Y)$ and $\bar{\Delta}\theta$ shows the average error of estimated positions and orientation with respect to 21 locations. In our current system, it takes 66 ms to do this processing for one frame. Due to this long processing time, our robot do not estimate its location and orientation in the competition.

**Fig. 2.** Experimental result

6 Communication

There is no communication between our robots in the competition.

7 Strategy

The behavior generator decides the robot's behavior such as avoiding a wall (called avoiding behavior), shooting a ball into a goal (called shooting behavior) and defending own goal (called goalie behavior). We make a simple strategy for shooting the ball into the goal. To shoot the ball to the goal, it is important that the robot can see both ball and goal. Therefore, the robot must round the ball until the robot can see both ball and goal with the camera toward the ball. Finally, the robot kicks the ball strongly. Avoiding behavior is implemented in a reflex way based on the tactile information.

For preventing a ball from entering a goal, our goalie moves left/right with the center of robot body toward a ball, when a ball is approaching to our goal. The home position of the goalie is the center of a line close to our goal. The goalie only moves along that line.

8 Conclusion

An accurate localization method is a key technology for successful accomplishment of tasks in cooperative way. Next year, we will develop a method for estimating position and orientation of multiple robots using multiple ODIs based on geometrical constraints.

References

1. M. Asada, Y. Kuniyoshi, A. Drogoul, and et.al. “The RoboCup Physical Agent Challenge:Phase I(Draft)”. In *Proc. of The First International Workshop on RoboCup*, pages 51–56, 1997.
2. T. Nakamura, K. Terada, and et al. “Development of a Cheap On-board Vision Mobile Robot for Robotic Soccer Research”. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1998*, pages 431–436, 1998.
3. T. Nakamura and T. Ogasawara. “On-Line Visual Learning Method for Color Image Segmentation and Object Tracking”. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1999*, pages 222–228, 1999.

KIRC: Kyutech Intelligent Robot Club

Takeshi OHASHI, Shuichi ENOKIDA, Junich MINATODANI, Kazuhiko KAWAMOTO, Youji SHIGEOKA, Takaichi YOSHIDA and Toshiaki EJIMA

Kyushu Institute of Technology

1 Introduction

Autonomous soccer robots should recognize the environment from the captured image from a video camera and plan to a proper behavior. Furthermore, when some robots play cooperatively, communication system between robots is important. Because of simple vision and actuator system, the gap between the real world and simulation environment is considered to be small. Our research target is to accomplish multi-agent system using reinforcement learning based on a simulation environment.

2 Team Development

Team Leader: Takeshi OHASHI, Research associate

Team Members:

- Shuichi ENOKIDA, Junich MINATODANI, Kazuhiko KAWAMOTO, and Youji SHIGEOKA, Graduate student
- Muneichi KOUNO and Hisato NODA, Student
- Takaichi YOSHIDA, Associate Prof.
- Toshiaki EJIMA, Prof.

Web page <http://www.mickey.ai.kyutech.ac.jp/KIRC/kirc.html>

3 Hardware configuration

Our robot is designed simply. Special devices are not necessary to make it. Our robot can be utilized for a general platform of an autonomous robot including a soccer robot. A field player and a goal keeper are shown as Fig.1. The robot has a radio control tank chassis, a note type personal computer and a video camera. Main parts of the robot are listed in Fig.2. Its chassis based on a M4 Sherman that consist of independent left and right motors, gear-boxes, and tracks. It can easily pivot on the spot and run over rough field.

The field players mainly keep attention to their front view. This is because their tasks are to find the a ball and goals and at the same time avoiding other players and walls. They are equipped with a wide lens video camera toward front, which is tilted slightly down to remove the noise from outside of the field.

The goal keeper should watch not only front but also left and right side. There are some solutions to this problem. Either they use panoramic movable

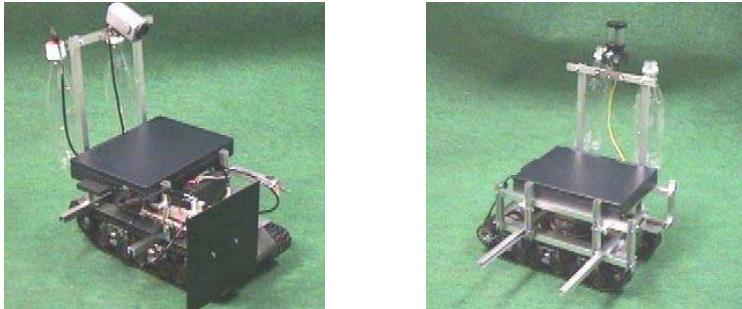


Fig. 1. Our field player and goal keeper

camera, or using two or three cameras, or using a fish-eye lens or a wide mirror. If it uses a panoramic movable camera, it should keep consistency of its body direction and the camera direction. By doing this the computational cost in the action planning will increase. If it uses more than one camera system, time taken for image processing will increase as due to image capturing and processing cost are in proportion to the number of cameras. If it uses a fish-eye lens or a wide mirror, viewable seen area is wider, but valuable image area in the image plain is reduced. We considered these as merit and demerit. We choose the goal keeper to have an omnidirectional vision system, because increasing the computational processing cost is more serious than reducing the image resolution.

3.1 Motor controller and kicking device controller interface

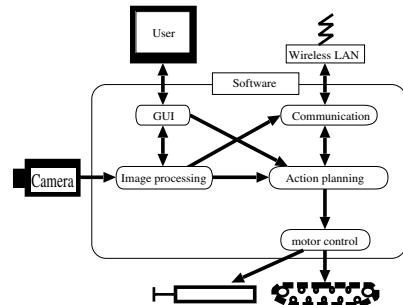
This radio control tank kit has independent left and right motors and gear-boxes. The digital control unit has two control modes. Mode 1 is to simulate steering wheel system and mode 2 is to directly control each motor separately. Our robot uses mode 2, each motor is controlled separately. This controller is connected in parallel port of the computer through a CPLD(Complicate Programmable Logic Device) interface unit.

The digital speed controller receives digital pulse signals. The duty ratio is variable from 6.8% to 11.4%. If the duty ratio is higher than 9.1%, the motor rotates forward, else if the ratio is lower, it rotates back. The interface logic is written in Verilog-HDL and download to Xilinx CPLD XC95108 board. The interface board has a parallel port adapter and DMD controller connectors and a serial port for rewrite the internal logic.

The kicking device includes two air cylinders, two electromagnetic valves, and an air tank. The valves are controlled by the CPLD unit independently. It is expected that explicit cooperative behavior (for example, in such cases as the pass) becomes possible by using the kicking device.

Fig. 2. Main components

Main components	Product name
Chassis	Tamiya M4 Sherman
Note PC	IBM ThinkPad-235
Video camera	Sony CCD-MC100
Capture card	IBM Smart Capture Card
Control interface	CPLD board
Wireless LAN	NCR WaveLAN/IEEE
Omni vision	Accowl
Kicking device	Kuroda air cylinder

**Fig. 3.** Software modules

4 Software configuration

Operating system of the robots are FreeBSD 3.4-RELEASE with the PAO3 mobile package. An application program is written in C++ language. Base robot class has fundamental common API for primitive behavior and communication. Each robot program is written as a derived class. It is easy to prepare many kind of players, for example, the variation is depending on its position and policy.

Software modules diagram is shown in Fig.3. Most modules are written as POSIX threads and the motor control module is written as another process, because the motor controller use timer alarm to archive a fail safe.

Image processing: The image processing module captures an image from a video camera and extract a ball, goals, walls, and robots by their template colors. First, the captured image is converted from RGB color space to TSL color space. Next, each pixel is calculated for the distance from the target objects' color. If the value of the distance is lower than a threshold value, the pixel is labeled as the target object. Last, the labeled image is scanned same labeled area, if the area size is larger than a threshold value then the target is found out. Same processes are applied for the other targets.

Motor and Kicking device control: FreeBSD 3.4-RELEASE has a special device, for example /dev/ppi0, which is used for direct access from user land process to a parallel port. The robot has ten primitive actions, slow forward, fast forward, forward left, forward right, pivots left, pivot right, back left, back right and back. A primitive action selected by the action planning module is sent to the motor control module with a valid interval. The motor control module sets an alarm for valid interval of the action. The alarm is override for each control message. If the alarm interrupt is invoked, the motor is stopped. This mechanism continues the same or similar actions without any stop, but it avoids going out of control when new actions are lost or late.

Communications: The communication module is consisted by two threads, one is to send the processed result as a broad cast packet to other robots over wireless LAN, another is to receive any data from other robots or a start/stop controller. Communication protocol uses UDP packet and does not wait ACK,

because sometime during the competition the wireless LAN would break down, it avoids the dead lock state. This protocol is similar to human players shouting protocol.

Behavior planning: Each robot selects same primitive actions, but it should choose a proper action that depends on the situation.

5 Future works

From the result of our experience in attending RoboCup-99, fundamental components, which are recognition of the environment from image processing, agent communication over wireless LAN and prior knowledge based action planning, worked well. However, we have not applied the reinforcement leaning method to the real and complicated tasks. Our current research interest is how to estimate to the actual action value function using continual base functions[2] and multi layered reinforcement learning[3]. The first approach will be realized by EQ-learning. In EQ-learning, it is assumed that an action value function is able to be replaced with the summation of weighted base functions. The weight of the function is adjusted in learning phase. EQ-learning can get the allocation of the base function which is appropriate for learning automatically and adjust the weight of base functions based on temporal difference learning. The second approach, multi layered learning, partitions behavior into a number of modules which are the fundamental behavior to accomplish a task. These behaviors are learned by using primitive actions, and more complicated behavior is realized by using them. In addition, we will study about multi-agent cooperative work. When a robot accomplishes a task (perform a soccer game) cooperatively, it is desirable that there are some common information. So, we design the robot of KIRC to recognize a global position which is an absolute position in the environment as the common information. The global position is acquired by using a landmark (goals, lines, and so on) from the image inputted from one video camera. We expect that cooperative behavior is produced more easily by using the global position. We would apply these approaches to real robot competitions.

References

1. Jean-Christophe Terrillon and Shigeru Akamatsu: Comparative Performance of Different Chrominance Spaces for Color Segmentation and Detection of Human Faces in Complex Scene Images Vision Interface '99, Trois-Rivieres, Canada, pp.18-21, 1999.
2. Shuichi Enokida, Takeshi Ohashi, Takaichi Yoshida, Toshiaki Ejima: Stochastic Field Model for autonomous robot learning, 1999 IEEE International Conference on Systems, Man and Cybernetics, 1999.
3. Masato Fukuda, Shuichi Enokida, Takeshi Ohashi, Takaichi Yoshida and Toshiaki Ejima Behavior Acquisition of Soccer Robot Based on Vision Sensor Information, Technical report of IEICE, PRMU99-82, pp. 31-38, (in Japanese), 1999-09.

CoPS-Team Description

N.Oswald, M.Becht, T.Buchheim, P. Burger, G. Hetzel, G. Kindermann,
R.Lafrenz, M.Schanz, M.Schulé, P. Levi

Institute for Parallel and Distributed High Performance Systems (IPVR)
Applied Computer Science - Image Understanding
University of Stuttgart, Breitwiesenstr. 20-22, 70565 Stuttgart, Germany
robocup@informatik.uni-stuttgart.de

Abstract. This paper presents the hardware and software design principles of the medium size RoboCup Team *CoPS Stuttgart* which are developed by the image understanding group at the Institute for Parallel and Distributed High Performance Systems (IPVR) of the University of Stuttgart. By adapting already successfully tested multiagent software concepts by our group to the domain of robotic soccer we intend to improve those concepts at the field of realtime applications with uncertain sensory data.

1 Introduction

Multiagent theory has become a popular research area in the context of artificial intelligence. Although applied to many domains, the full potential of this paradigm develops especially in situations where decisions have to be made upon uncertain data or partial information as e.g. in robotics. Deficiencies of a single agent in perceiving its environment and having only partial knowledge about its surroundings shall be compensated by its ability to exchange information with other agents.



Fig. 1. Our team of soccer robots

Furthermore multiagent systems support the idea of the whole being more than the sum of its parts, meaning that the problem solving potential of a multiagent system exceeds by far the capabilities of the agents within the system. This can be achieved by providing the agents with proper communicational abilities to negotiate and coordinate their actions.

The research group **Image Understanding** of the **Institute for Parallel and Distributed High Performance Systems** has been working for several years in the field of multiagent-systems developing a multiagent architecture [1], a theoretical framework for cooperating agents [2] and applying cooperative concepts in computer vision [3]. With our medium size robotic soccer team *CoPS* (Cooperative Soccer Playing Robots) those developed concepts are tested and adapted to the real world application of a soccer match.

2 Hardware Equipment

Our RoboCup Team consists of 6 Nomad robots of type *Super Scout* [4] equipped with additional sensorics and a kicking device. The onboard computer system of the Super Scouts consists of a Pentium 233 MMX with 64 MB and a hard disc of 4 GB capacity. The two wheel differential drive allows robot motion with a maximum speed of $1m/s$ and an acceleration of $2m/s^2$.

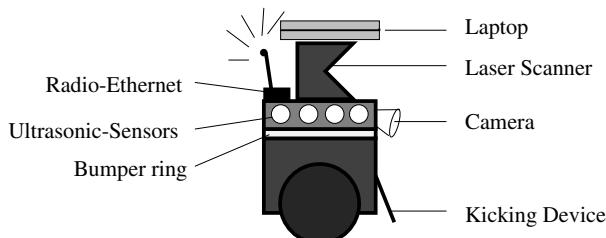


Fig. 2. Equipment of a soccer robot

Each of these robots is already equipped with odometric sensors, a tactile bumper ring of 6 independent sensors and 16 ultrasonic sensors arranged around the outside perimeter of the robot. The 6 tactile sensors allow a coarse localization of physical contacts with the environment in the front-left/center/right or the rear-left/center/right of the robot.

2.1 Modifications and additions

We installed a $1/3''$ -Chip CCD camera with a 752×582 resolution delivering a video signal to a MATROX frame grabber on the Pentium board. By image processing algorithms the ball and the two goals are extracted from the grabbed pictures and their current position relative to the robot is estimated.

As the precision of the odometric sensors degrades constantly with every movement of the robot, we equipped each robot with a SICK laser range finder¹ for a more robust and exact self localization. Within an angle of 180° the range finder provides depth information with a resolution of 0.5° and an accuracy of 5cm. To derive the current position from the laser data the measured laser image is iteratively rotated and shifted until it fits best a predefined model of the football field.

The Pentium board is connected to a radio-ethernet HUB enabling communication with other robot agents.

For passing and kicking the ball we constructed a special kicking device which is driven by two solenoids. The camera, laser range finder and kicking device are mounted towards the front of the robot players. Only the goalkeeper's sensors are shifted by 90° as it only moves unidirectionally forwards or backwards to defend the goal.

3 Software Concepts

3.1 Agent Architecture

In our software model each robot consists of a set of concurrent software modules, so called elementary agents (EA). Each EA has special plans to perform tasks which can be requested by other elementary agents. We classify EAs into three categories according to the level of abstraction of the tasks they perform.

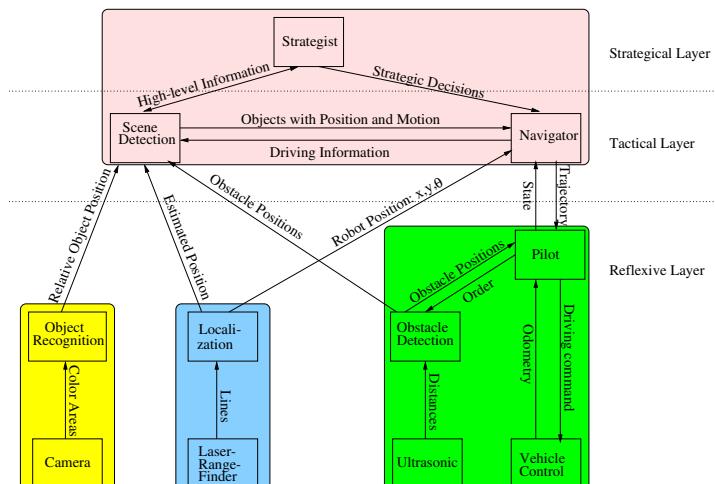


Fig. 3. System architecture

¹ We'd like to thank the SICK AG,Germany for lending us three Laser Range Finders

EAs concerned with simple control tasks of the sensors and actuators are situated at the lowest layer within the architecture, **the reflexive layer**. Reflexive tasks generally involve only simple planning or no planning at all, guaranteeing an immediate response on a request. The Pilot-EA e.g. performs tasks like driving a given route straight ahead and rotation by a given angle, while the Localizer-EA determines the current position by odometric and laser range finder data.

More complex tasks are performed by the EAs at **the tactical layer** of our software architecture. An EA concerned with scene-detection aggregates and evaluates information from the sensors of a robot and builds a model of the environment which can be requested by other agents. A navigator-EA, also located at this level and responsible for vehicle control, is concerned with tasks like *getting the ball* or *moving towards the goal*. Planning such tasks requires information about the environment which is provided by the scene-detection.

Finally, at the strategical layer, the Strategist-EA is concerned with long term goals and team coordination. It generally has to react only on external signals like e.g. *start of the game*, *goal* and *end of the game*. All EAs can send task-requests to each other within the same robot as well as to elementary agents of other robots. Thus, data-uncertainties concerning the environment can be compensated and a means for coordinating joint actions is provided.

For a more detailed description of our software architecture we refer to our technical paper [5].

3.2 Implementation Issues

The elementary agents were all designed as separate multi threaded processes on a LINUX system, a communication thread waiting for task requests from other agents and further executional threads processing requested tasks. For interagent communication we used the freely available CORBA [6] implementation MICO [7] which facilitated the distributed agent modelling a lot.

References

1. P. Levi, M. Becht, R. Lafrenz, and M. Muscholl. COMROS - A Multi-Agent Robot Architecture. In *DARS 3*. Springer-Verlag, 1998.
2. M. Becht, M. Muscholl, and P. Levi. Transformable multi-agent systems: A specification language for cooperation processes. In *Proceedings of the World Automation Congress (WAC), Sixth International Symposium on Manufacturing with Applications (ISOMA)*, 1998.
3. N. Oswald and P. Levi. Cooperative vision in a multi-agent architecture. In *LNCS*, volume 1310, pages 709–716. Springer-Verlag, 1997.
4. <http://www.robots.com>.
5. R. Lafrenz, N. Oswald, M. Schulé, and P. Levi. A cooperative architecture to control multi-agent based robots. Submitted to : Proceedings of PRICAI, 2000.
6. <http://www.corba.org>.
7. <http://www.mico.org>.

Golem Team in Middle-Sized Robots League

Golem Team

Paolo de Pascalis, Massimo Ferrareso, Mattia Lorenzetti,
Alessandro Modolo, Matteo Peluso, Roberto Polesel, Robert Rosati,
Nikita Scattolin, Alberto Speranzon, Walter Zanette
PDP, MF, ML, AM, MP, RP, RR, NS, AS, WZ undergraduate students
of University of Padua, Italy

golemteam@wappi.com,
WWW home page: <http://gteam.browse.to>

Abstract. Golem is an holonomic robot designed to be compliant with Robocup regulations for the middle-sized league. The project consists in three parts: mechanics and hardware, vision system and software. We adopted the universal three wheels model to achieve a great freedom of movement. In order to take advantage of this particular feature, the vision system has to give a full sight of the environment. This objective is achieved by a mirror mounted on the mobile base. Decision making and planning are based on the knowledge of the relative position of all objects in the field (ball, walls, robots, goals) and especially on the recognition of teammates and opponents. Every robot can play as goalkeeper, defender or attacker.

1 Introduction

Golem Team is a team compounded by some students of University of Padua who participated to Robocup 1999 World Cup, held in Stockholm, as members of ART - Azzurra Robot Team. After this important experience we have decided to start this new project that tries to cover all parts of a robot building process from the mechanics to the high level software.

In this paper we analyze these some main topics. In the following section we give an overview of this holonomic mobile base focusing on the mechanics and hardware. In the third section we describe the vision system from the point of view of the hardware and software. In the fourth section we explain how the software is embedded in the system and in the last we conclude with some general consideration about our team.

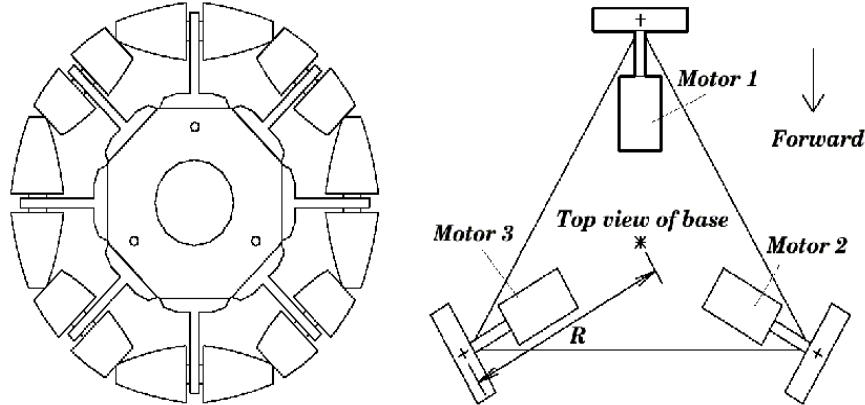
2 Mechanics Architecture

The motion unit consists of three universal wheels placed at the vertexes of an equilateral triangle. A brush motor drives each wheel. Our group designed a controller, based on the Hitachi H8/3334, to calculate and adjust the speed of each wheel. An optical shaft encoder is mounted on every motor and its resolution is such that one pulse represents 0.18 degrees of wheel rotation. Linear and rotational velocities may be independently

set by means of the mechanical architecture adopted, which allows the robot to follow the desired path without any constraint on its orientation, simplifying the playing algorithms.

Let r be the wheel radius, R the distance between a wheel and the center of robot and $\omega_1, \omega_2, \omega_3$ the angular speeds of wheels; it follows that :

$$\begin{aligned} V_x &= \frac{1}{3} r \omega_3 - \frac{1}{3} r \omega_2 \\ V_y &= \frac{1}{3} r \omega_3 + \frac{2}{3} r \omega_1 - \frac{1}{3} r \omega_2 \\ J_{\text{og}} &= \frac{r}{3R} \omega_3 - \frac{r}{3R} \omega_1 - \frac{r}{3R} \omega_2 \end{aligned}$$



We observe that if $\omega_3 = \omega_2$ and $\omega_1 = 0$ then $V_x = 1.5 r \omega_3$ that is greater than the velocity which could be achieved by a unicycle robot.

Every robot is equipped with a pneumatic kicking device which can be used to kick in different directions and to dribble the opponents.

3 Vision System

To take advantage of the physical structure of the mobile base, we adopted an omnidirectional vision system. The hardware solution consists of a camera looking at a custom-made mirror. In this way we obtain a 360 degree perception of the field in a single image. The loss of resolution, typical of this kind of structure, is balanced by its efficiency in the tracking of evolution of the game.

The camera is a commercial product, the Hitachi VK-C78ES, an high-resolution camera whose lens gives a visual of 45 degrees; it is set up vertically pointing to the overlooking mirror.

The mirror was designed by Golem Team and was optimized for Robocup applications. It has a convex surface consisting of three different curves each dedicated to a particular kind of information, creating in this way three different zones of analysis. The inner zone reflects the whole view of the field: the profile of mirror is designed to

minimize the reflected robot image. The middle zone is tailor-made to allow searching of markers placed on robots; the teammates recognition is a fundamental step to obtain a coordinated game and it is performed considering the color of the marker worn by the players. The middle zone of the mirror reflects the objects located between 30cm and 80cm of height from the field plan. The external zone, finally, is dedicated to the analysis of objects in proximity to the robot. A good ball control is possible only knowing its exact position, so the mirror shape increases the resolution of the image in this part of the visual area.

The equations of the curves composing the mirror were calculated upon a maximum acceptable error, function of the distance from the focus of the camera system.

The camera sends an S-video signal to a frame grabber based on the Conexant BT848 Chip. This is a simple and cheap system to capture high quality images from a camera.

The main problem to solve was the good real-time extraction of information from the frames captured. Every kind of object (ball, robots, field components) needs a dedicated approach to be recognized. Some data are needed with high frequency, while others can be refreshed less frequently. In order to reduce the computational load of the CPU, the vision software spawns several processes each of which evolves in parallel with its own frequency. For instance, we need to know the position of the ball with the highest possible frequency, so the searching process must be quick since it has to work at a frame rate of 25fps but, moreover, it must be robust in order to extract the ball position in a noisy environment. On the other hand walls do not move too far in the entire game, thus we can sporadically track them. This architecture let us use heavy algorithms blended with simpler ones yielding a system more flexible and capable to adapt to the on-board computer computational resources.

4 Planning and Behavior-Based Control

The whole set of software modules for controlling the players has been developed over the ETHNOS¹ kernel. ETHNOS is a real-time programming environment which exploits the LINUX RT multithreaded operating system and provides communication facilities. It provides support for the real-time execution of periodic and sporadic tasks (called Experts).

Vision Experts, Planning Experts and the Arbitration module are realized as periodic Experts while each behavior is a sporadic Expert. At every decision step the Arbitration Expert activates a single basic behavior (such as 'kick ball' or 'intercept ball' or 'follow teammates', etc.). Every behavior tries to perform its task according to planning directives (for instance the preferred direction to avoid an obstacle). Communication of raw data is not used. Every robot communicates 10 times per second the value of a function (Q function) which quantifies its priority in the action. Q is a function of ball distance and velocity, relative positions of ball, robot, goal and precedent results in performing the action. The hierarchy introduced by Q prevents a robot from interfering with the action of a teammate which could be carrying the ball.

¹ ETHNOS : Expert Tribe in a Hybrid Network Operating Systems developed by DIST of University of Genoa, Italy. <http://www.ethnos.dist.unige.it>

Since players have the same planning system, they can make assumptions on teammates intentions. This is a fundamental capability for obtaining an effective coordination without using a global positioning system or explicit communication. Every robot can play as defender or attacker, according to situation, score and its own disposition.

5 Conclusions

In this paper we presented the Golem Team for Robocup 2000 contest and we described the hardware and software of the robots. In our opinion this mobile base permits to perform a good play in the field and enhances coordination possibilities.

It should be clear that we want to reduce the explicit communication of information, let any single robot to act only using information obtained from the vision system that is the main sensor of these players.

From the point of view of the high level software, that represents the reasoning module of the robot, it can be considered as a layered module where the internal layers are the basic behaviors. Our work is now focused on the external layers that allows to improve the general capabilities of this kind of autonomous systems.

References

1. A. Blake, M. Isard, Active Contours : The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion, *Springer Verlag*, Mar. 1999.
2. J. Borenstein, H. R. Everett, L. Feng, Where am I? Sensors and Methods for Autonomous Mobile Robot Positioning - Available from <ftp://ftp.eecs.umich.edu/people/johannb/>, 1996.
3. M. Piaggio, R. Zaccaria, Distributing a Robot System on a Network: the ETHNOS Approach - *Advanced Robotics Journal*, Vol. 12, N. 8, VSP, 1998.
4. M. Barabanov, V. Yodaiken, Introducing real-time Linux - *Linux Journal*, (34), Feb. 1997.
5. Ronald C. Arkin, Behaviour Based Robotics - *MIT Press*, Cambridge, MA, 1998.
6. R. Cassinis, A. Rizzi, Design issues for a Goalkeeper Robot - *Robocup-99 Workshop*, Stockholm 1999.
7. F. Marchese, D.G. Sorrenti, "Omni-directional vision with a multi-part mirror", *Presented at 1st workshop eurobocup2000*, Amsterdam; submitted to workshop robocup2000 Melbourne.

Osaka University “Trackies 2000”

Yasutake Takahashi, Eiji Uchibe, Takahashi Tamura, Masakazu Yanase,
Shoichi Ikenoue, Shujiro Inui and Minoru Asada

Dept. of Adaptive Machine Systems,
Graduate School of Engineering Osaka University,
Suita, Osaka 565-0871, Japan

{yasutake,uchibe,tamtam,yanase,ikenoue,inui,asada}@er.ams.eng.osaka-u.ac.jp

Abstract. This is the team description of Osaka University “Trackies” for RoboCup2000. The hardware and software architecture are presented.

1 Introduction

The Robot World Cup Soccer Games and Conferences (RoboCup) are a series of competitions and events designed to promote the full integration of AI and robotics research. The robotic soccer provides a good test-bed for evaluation of various research, e.g. artificial intelligence, robotics, image processing, system engineerings, multi agent systems.

Osaka University “Trackies” has participated the RoboCup since the first one, held in Nagoya, Japan, in 1997. We have changed our robot system from a remote controlled vehicle to a self-contained robot because the remote brain system often suffered from much noise of camera image and motor command transmissions. On the other hand, we adopt an idea that the cooperative behaviors without any planning to emerge cooperation in the dynamic environment cased by multi robots in a hostile environment.

This paper presents the employed hardware, namely robots and computers, and a fundamental approach to control robots.

2 Hardware

Fig.1(a) and (b) show pictures of our mobile robots we designed and built. Fig.1 (c) shows an overview of the robot system. The system consists of a motor driving unit, a vision system, and a control unit.

2.1 Control Unit

We use an on-board computer as a controller for the robot. It is a standard PC parts, and commercially available. Each PC has a 233 MHz Intel Pentium MMX CPU, 128 MB RAM and 160MB Silicon Disk Drive (as a hard disk drive). The operating system is Debian GNU/Linux, and we install the minimum system in the hard disk drive in order to communicate with external file server. The

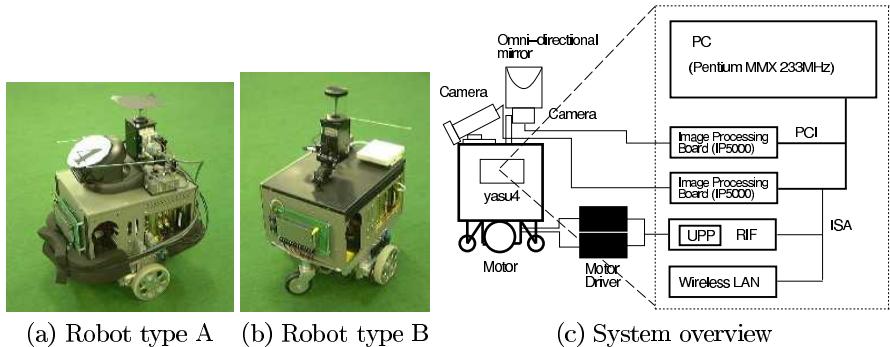


Fig. 1. Our mobile robots and an overview of the robot system

software development of our mobile robot is done on the external file server. For the communication between a robot and an external file server, wireless PCMCIA Ethernet cards (AT&T WaveLAN/IEEE) in PCMCIA to ISA adaptors are used. Each computer has one or two image processing board (Hitachi IP5005) or frame grabber for vision system, and one Robot Interface Board which has an Universal Pulse Processor for motor control.

2.2 Vision System

The robot has an omni-directional camera system and/or a normal camera system.

Some robots have SONY EVI-D30 which has a motorized pan-tilt unit. An omni-directional camera system consists of a color CCD camera and a omni-directional mirror and is mounted on the robot as the camera optical axis is aligned with the vertical axis of the mirror. A simple color image processing is applied to detect the ball, goals, field, and obstacle areas in the image in real-time (every 33ms).

2.3 Motor Driving Unit

The driving mechanism, which is originally designed at Tsukuba University and now commercially available as a vehicle unit with motor, wheels, and control drivers, is a PWS (Power Wheeled System); the vehicle is fitted with two differential wheels. The wheels are driven independently by separated DC motors, and two extra free wheels ensure the static stability. Robot Interface Board generates PWM pulse, which controls the speed of the wheel. The pulse are sent to the motor drivers, and drive the wheel.

3 Software

Our robot team consists of four identical robots. They all share almost same basic hardware, but they differ in their behavior programming. The basic idea for

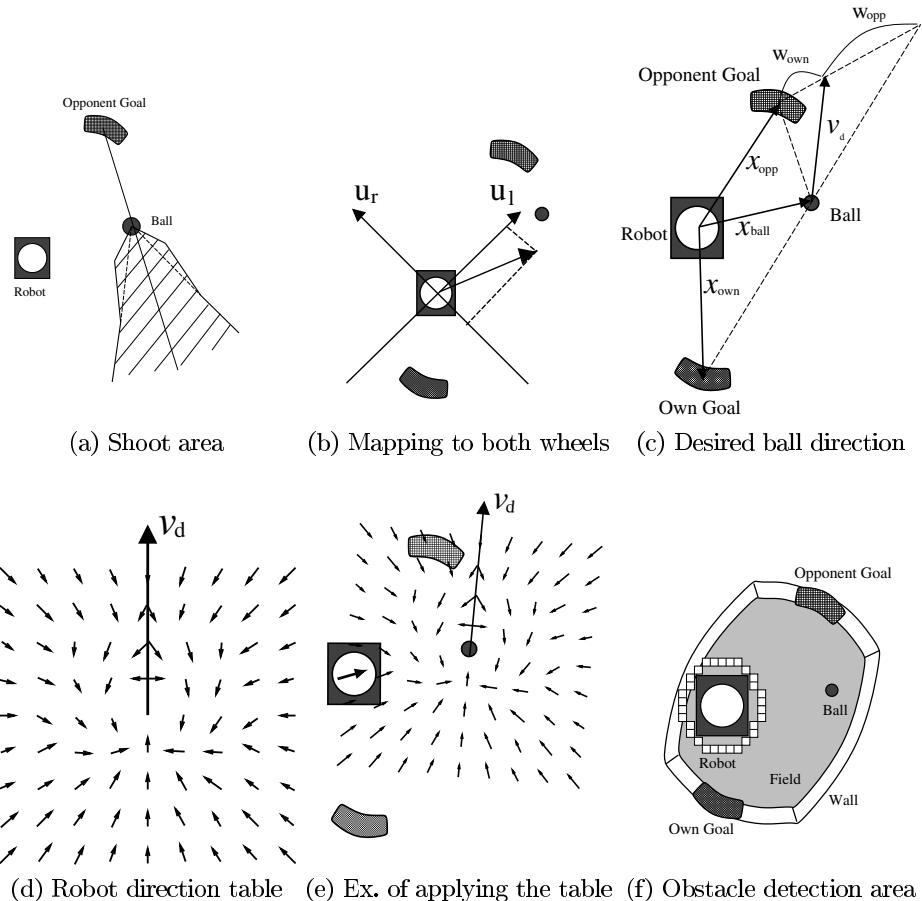


Fig. 2. A typical strategy of our robot

cooperative behaviors among the teammates is that cooperative behaviors without any planning or intention of cooperation emerges in the highly dynamic, hostile environment provided by RoboCup. We design each robots' behaviors such as "use the environment directly," "replace computation with rapid feedback," and "tolerate uncertainty before trying to reduce it." [1] We describe a typical one here.

All robots of our team have a camera with an omni-directional mirror, and capture the omni-directional camera image. They decide their all behavior based on the information in the image plane.

First of all, we prepare two submodules; One is a shooting module and the other is a turning around module. The robot switches the two modules according to the situation. It turns on the shooting module when the robot is in the situation that the ball is in the same direction to the opponent goal. A robot can

detect this situation on the camera image as that the robot is in the shadowed region in the Fig.2(a).

The shooting module drives the robot toward the ball; the module defines the vector to the ball on the omni-directional image plane. We convert the vector to the speed of the both wheels in a very simple way; We define a left wheel axis and a right wheel axis on the omni-directional image plane as shown in Fig.2(b). The projection from the vector to the left and right wheel axis leads the robot to the desired direction.

The turning around module is a little bit complicated. We divide this policy decision process into the following two states.

1. Definition of the desired ball direction
2. Mapping from the ball location and the desired ball direction to the desired robot direction

The definition of the desired ball direction \mathbf{v}_d is given by (see Fig.??(c)):

$$\mathbf{v}_d = [w_{own}(\mathbf{x}_{ball} - \mathbf{x}_{own}) + w_{opp}(\mathbf{x}_{opp} - \mathbf{x}_{ball})]/(w_{own} + w_{opp}) \quad (1)$$

where \mathbf{x}_{ball} , \mathbf{x}_{own} , and \mathbf{x}_{opp} are the vectors of the ball, the own goal and the opponent goal on the omni directional image plane, respectively. And w_{own} and w_{opp} are the weights for the own goal and the opponent goal, respectively. Each weight is the area of the each goal region on the image, and will be zero if the robot cannot detect the goal. The definition of these weights means that if the robot is near to the own goal, the ball should be moved against the own goal, and when the robot is near to the opponent goal, ball should go to the opponent goal.

In order to carry the ball into the desired direction, we define a table that maps sections of the visual field to motion commands. Fig.2(d) shows the table; the center of the figure indicates the ball and the vertical axis indicate the desired ball direction \mathbf{v}_d . Arrows indicate the desired robot direction as response to the robot position in the segments of the visual field. Fig.2(e) shows an example.

The obstacle avoidance behavior is implemented as follows: Fig.2(f) shows the obstacle detection areas. The robot recognizes that the area is free, if the green color occupies the area on the image. If the other colors (for example, black (the robots) and white (walls)) occupied the area and the desired robot direction crosses the area, the robot changes the desired direction next to the area.

Acknowledgment We would like to thank Dr. Shoji Suzuki, Tatsunori Kato, and Hiroshi Ishizuka for their assistance in designing and building the initial version of our robot hardware and software.

References

- [1] Barry Brian Werger. Principles of minimal control for comprehensive team behavior. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pages 3504–3509, 1998.

Essex Rovers Team Description

Huosheng Hu, Dongbing Gu, Bo Li

Department of Computer Science, University of Essex

Wivenhoe Park, Colchester CO4 3SQ, UK

Email: {hhu, dgu, bli}@essex.ac.uk

Abstract. This article introduces our current research efforts to build a multi-agent system for cooperation and learning of multiple Sony legged robots in the RoboCup domain. A behaviour-based hierarchy is proposed for our Essex Rovers robot soccer team to achieve intelligent actions in real time, which includes both a neural network based color detection algorithm and a fuzzy logic controller.

1. Introduction

RoboCup (Robot World Cup Initiative) provides a challenging environment for research in a team of robotic systems that need to achieve concrete objectives, particularly in the presence of an adversary team. In the Sony Legged Robot League [2], there are three Sony legged robots in each team. According to the competition rules, the field size and other teams' experience [9], our team, Essex Rovers, has adopted the team formation with two robots playing as attackers and one as goalie. For the first time to take part in the competition, it was too much to set down for our team. Fortunately we have manuals and support from Sony [8] and the reports from other teams competed in RoboCup99 [1]. All these were really helpful.

Our preparation for the competition can be divided into three stages. At the 1st stage, we read every piece of information available to see how to program and tested each example that Sony provided in order to see how the robot operates. Then we came to the 2nd stage to construct our team strategies. We targeted two goals under the pressure of time limit: to quickly set up everything for the robot to play soccer and to gradually modify the program that appeared not reasonable. This strategy does reduce the stress we faced and it took us about three months to have a fully working team. In the 3rd stage, we refined our sensing and control strategies gradually. Our team showed the abilities to play the match and challenges during the last few days before leaving for Melbourne. Our research is focused on learning and evolving of the Sony legged robots based on the behaviour structure.

2. Architecture for building the intelligent robot

We have designed agent-based control architecture for our robots, which consists of three modules: Perception, Cognition, and Action [3]. As shown in figure 1, three modules are placed under the OPEN_R structure and communicate with each other through their connections. The Perception module takes all sensor information as inputs, including images from a CCD camera, ranges from an infrared range finder, postures from gyros, pressures from a head touch sensor. It maintains a local map relative to the robot centre and a global map relative to the left corner of the field by updating the localization of different objects defined in the map based on different sensory information. Both the local map and the global map are C++ objects shared

through their points. The Cognition and Action modules form hierarchical behaviour structure [4].

The high-level behaviours in the Cognition module behave as the states of a non-linear dynamic process actuated by the two maps, which will be stabilized either in one state such as FINDBALL behaviour or in the limit cycle states formed by different behaviours. The low-level

behaviours in the Action module are called basic behaviours, each of which can complete a primitive action such as APPROACH_TO_BALL according to the information from the maps. The fuzzy logic controller is used here to implement the action while handling uncertainty [6].

3. Vision

We developed a tool called YUVC by using Microsoft Visual C++ to generate the Colour Detection Table (CDT) for image thresholding based on the built-in threshold hardware of the robot. The CDT has 8 tables for 8 colours, each of which has 32 levels of luminance value (Y) for 4 min-max threshold colour values of U, V in the YUV colour space. YUVC is a window-based tool and makes manual colour separation easy by moving mouse only. Furthermore it also provides the ability to simulate the image processing with same C++ code employed in real robots.

4. Object recognition

The procedures for object recognition [3] are:

- Image capture – AIBO provides the colour images in YUV space and each pixel in the image is represented by 3 bytes of Y, U, and V values. It also provides 8 colour images after each captured image being threshold by hardware. The hardware threshold makes use of 32 Y levels and there are 2 thresholds for U and other 2 thresholds for V at each level. This will lead to 32 rectangles in U and V frames for each colour. In order to select the thresholds for the hardware, we developed a software tool that can manually label the different colour and find the thresholds among the labeled pixels.
- Image segmentation – The threshold image may contain noise due to the luminance condition. We use morphology filters to de-noise the image since the detected object's shape is known a prior [7]. For a binary image, there are two operators normally used in a morphology filter, namely dilation and erosion. Morphologically filtering an image by an opening or closing operation corresponds to the ideal non-realizable band-pass.

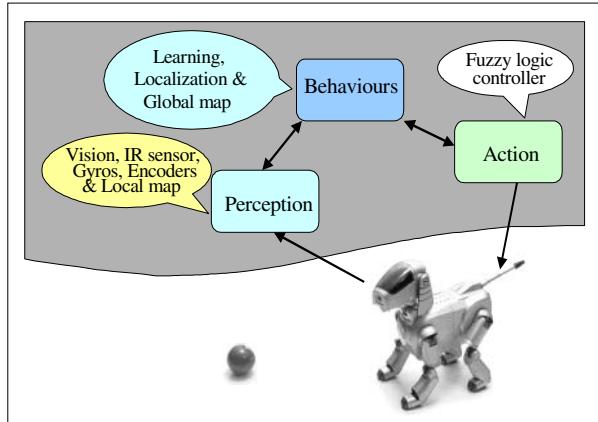


Figure 1 Agent architecture for Essex Rovers

- Image representation – Before object understanding the similar adjacent pixels have to be grouped into the connected regions. This is typically expensive operation that severely impacts real time performance. We calculate the run length encoding (RLE) to represent the image in order to make our next image operation based on RLE not on individual pixels. Region identification can be performed in two passes, see [3] for more details.

5. Localization

There are two maps for localization in our team, namely the local map relative to the robot center and the global map relative to the left corner of the field, both of which maintained by the sensory information. The object features are kept in the maps, for example, the distance, view angle, pixel size in the local map and 2-D co-ordinations and orientation in the global map. A certainty value (CV) that is exponentially decayed is assigned to each object in the map. The purpose of CV is to equip the robots with memory in order for them to make decisions when the desired objects are lost from their sight. For the attackers, the triangulation that is calculated by recognizing three poles is used for localization. The condition for activating the localization is limited in only a few cases, such as losing ball for a certain time, in order to save time to react with the dynamic environment. For the goalie, a set of heuristic rules is developed for localization. The rule set takes the goals, poles, ground, and white lines in both the local map and global map as inputs and is more efficient than triangulation in some situations in which some poles are perhaps obscured by other robots or not recognized due to the changing lighting condition. The localization will be triggered when the robot has no ball or cannot see the ball.

6. Behaviours

There are two level behaviours in our robots. The high-level behaviours contained in the Cognition module as the states of a non-linear dynamic process actuated by the two maps, which will be stabilized either in one state such as FINDBALL behaviour or in the limit cycle states formed by different behaviours. The low-level behaviours in the Action module are called basic behaviour, each of which can complete a primitive action such as APPROACH_TO_BALL according to the information from the maps. Both the attackers and the goalie have their different Cognition modules, and these modules then form team behaviours such as localization, co-operation, etc.

A reactive control scheme [3] is employed in the low level for each behaviour with the sensory data from the local map as inputs and moving command from OPEN-R moving modular as outputs. The sensor data from the local map is corrupted with noise, which forces us to use a fuzzy controller to complete the behaviour issued from the Cognition module.

7. Action/Execution

After testing for different walking styles provided by MoNet object of Aperios, we use the normal walking command as our team's motion styles. The two custom walking commands, SideWalking and Rolling, are designed by trial and error to make the robot move more flexibly. The kick command of MoNet is not efficient in testing

due to its slow moving and inaccurate angle. Therefore we use the fast walking style for kicking instead. Head motion for finding the ball is also difficult to control since there is a time delay between the time when the robot finds the ball and the time when the command is executed. The head angles are recorded when the ball is founded and then head is moved back to that position after the current command is finished.

8. Conclusion

In this article, we present our effect on training Sony legged robots for the competition in RoboCup-2000. The vision processing and the behaviour construction are two key aspects being focused by our team. A morphology filter is employed for image pre-processing. The behaviours serve as basic building blocks for our modular software system. The structure of two level behaviours can de-composite the task into contextually meaningful units that couple perception and action tightly. Fuzzy logic implementation further enhances the abilities of the system in face of the uncertain situation [10].

Based on the current system, we will focus on the robot's learning and evolving abilities for the next competition, particularly on the vision system and cooperative behaviours. The main shortcomings for our team in RoboCup-2000 are the slow motion and inefficient kicking. We will try to modify the walking style to improve its mobility and accuracy, and add effective kicking behaviours like the UNSW team.

Acknowledgements: We would like to thank Masahiro Fujita and other staff members in the Digital Creatures Laboratory, Sony Corporation, Japan for their technical support. This research is financially supported by the Royal Society Research Grant 574006.G503 and the Essex University Research Promotion Fund DDPB40.

References:

1. M. Asada, et al, RoboCup: Today and tomorrow what we have learned, *Artificial Intelligent*, 110, pages 275-292, 1999.
2. M. Fujita, H. Kitano, K. Kageyama, A reconfigurable robot platform, *Int. Journal of Robotics and Autonomous Systems*, Vol. 29, No. 2, pages 119-132, 1999.
3. D. Gu and H. Hu, Towards Learning and Evolving of a Team of Sony Legged Robots, *The Workshop on Recent Advances on Mobile Robots*, Leicester, 29 June 2000
4. H. Hu and D. Gu, A Multi-Agent System for Cooperative Quadruped Walking Robots, *IASTED Int. Conf. Robotics and Applications*, pages 182-186, Honolulu, Hawaii, USA, August 2000
5. H. Kitano, et al, Sony Legged Robot for RoboCup Challenge, *Proc. of the 1998 IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, pages 2605-2612, May 1998.
6. A. Saffiotti, E. Ruspini, and K. Konolige, Using Fuzzy Logic for Mobile Robot Control, *Practical Applications of Fuzzy Technologies*, In H-J. Zimmermann, editor, Kluwer Academic Publisher, pages 185-206, 1999.
7. M. Sonka, V. Hiavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, PWS Publishing, 1998.
8. Sony Corporation, *Entertainment Robot ERS-110: Operation Manual*, 1999
9. M. Veloso, E. Pagello, H. Kitano (Eds.), *Lecture Notes in Artificial Intelligence*, 1856, *RoboCup-99: Robot Soccer World Cup III*, Springer, 2000.

French LRP Team's Description

Vincent Hugel, Patrick Bonnin, and Pierre Blazevic

Laboratoire de Robotique de Paris, 10/12, avenue de l'Europe,
F-78140 Vélizy, France
hugel@robot.uvsq.fr

Abstract. This paper describes the problems encountered in vision by the French team in RoboCup2000. Since the participation of LRP in 1998, the team has focused on the 3 following topics: locomotion, vision and strategy. In RoboCup 1999 in Stockholm, Sweden, we carefully designed the locomotion patterns [1][2][3] to make the robots walk as fast as possible. We also implemented some trajectory correction while walking. In RoboCup 2000 in Melbourne, Australia, our team tried to improve the vision system to see better and farther. We also focused on adding some special strategy features to improve the robots' behavior.

1 Introduction

From the beginning (RoboCup 98, exhibition of legged league in Paris), the French team has decided to code its own algorithms for locomotion and vision. This has proved very useful because it is possible to master every detail of the computation. The French team got the 2nd place in 1998, became champion in 1999 and finished second in 2000. After the victory in 1999 against the UNSW team, it was clear that it would be very difficult to defend our title the following year. We did not improve our walking patterns very much. The vision system designed in 1999 was enhanced thanks to special high-level filtering. However lighting conditions were not satisfactory for our vision system to show its best performances. In fact, that is the strategy that played a significant role in our second place [4]. In this paper, the first section deals with locomotion. The second section is devoted to the vision system and the last section describes the special behaviors designed to play the final.

2 Locomotion

Since our locomotion patterns were successful in 1999, why not reuse them in RoboCup 2000? We reused them and tried to add some improvements to avoid situations of falls. Falls may occur when the robot starts to walk on the inclined wall that makes the border of the field, when it bumps into another robot, or when he crosses the separating line between the goal and the green carpet (the goal floor is made of plastic and is more slippery than the carpet). To prevent from falling, we tried to detect the collision and trigger some reflex movements.

However it appeared that detection of collision was very difficult to achieve. As a matter of fact, either there were some non-desired detections, or detection was not so reliable (50% of falls remaining). In case of non-desired detections, the robot was often suddenly interrupted in its strategy of attacking the ball and consecutively lost a lot of time, which was not acceptable for a soccer game. The case of non reliable detection did not bring a significant improvement in the behavior. We decided not to use it in the matches after the training tests.

3 Vision

In 1999, our vision system was fairly reliable, and our robots were able to see the ball within the range between one third and one half of the field length. Detection was based on the hardware color detection table (CDT) that equips the Sony quadrupeds. Thanks to optimized computation, the image processing rate was 20 images per second. This was possible by implementing low level filtering similar to an opening procedure. The main objectives for RoboCup 2000 were to increase the range of sight of the ball and the markers around the soccer field, and to run at video rate. For this purpose we focused on the two following points: how to automatize the tuning of the CDT (selection of threshold parameters), and how to design new low level filtering.

3.1 The different confusions

Paradoxically we encounter more problems in vision than last year. The ball was smaller, shiny. In addition, lighting conditions were bad compared to RoboCup 1999. We suspect that a lot of confusions come from the fact that the soccer field was unequally illuminated.

1. The first type of confusion is the confusion between two landmarks (a landmark is composed of two piled blocks of different colors among yellow, blue, pink and green) when the ball is near one of them with pink bottom (see Fig. 1). In fact the orange of the ball is seen as a yellow color, and all the markers with pink bottom on one side can be mistaken for the pink-yellow marker on the corner of the opposite side. A very bad consequence of this is that the robot can take the wrong direction and push the ball towards its own field.
2. The second type of confusion is most classical. The orange of the ball is confounded with the yellow color of one of the goal. The first consequence of this is that the robot sometimes sees a ball inside the side walls of the yellow goal. This comes from shadows of side walls that darken the yellow color of part of the side walls. The second consequence is that the robot may see the yellow goal inside the ball ! This may be due to the brightness on the top of the ball.

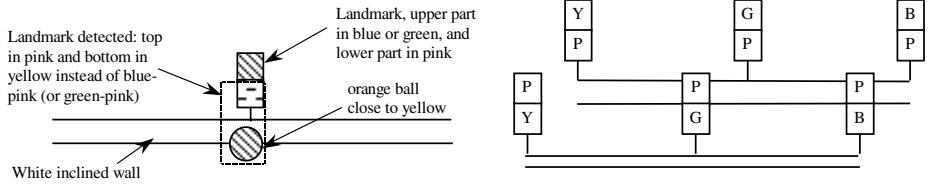


Fig. 1. **Left:** marker-ball colour confusion. **Right:** right and left length sides of the soccer field. The yellow goal is located between the yellow-pink (Y/P) and pink-yellow (P/Y) markers. The blue goal is the opposite one. The pink color at the bottom of one of the marker and the orange of the ball near the marker are confounded with the pink-yellow marker on the other side of the field. This is due to orange-yellow confusion.

3.2 On improving the vision system to avoid confusion

In 1999, the tuning of the CDT parameters required some expertise from the user. For RoboCup 2000, we designed new functionalities for our tuning interface to allow users who were not experienced in image processing to do the job of tuning. The new features added were based on cooperative multi-spectral edge/region segmentation. However due to bad lighting conditions and brightness on the ball we had to restrict the areas inside the objects to detect in the image. Tuning of parameters appears to be more difficult than expected. We also developed low level filtering procedures with various degree of filtering, similar to *mathematic morphology* procedures. The advantage of these filters is that they can be processed at video rate together with the extraction of connected components during the same image scanning. However, because these filters reduced the range of sight of the ball, they were replaced by high level filtering procedures during the competition. Unfortunately, two many cases of confusion appeared, and it was not possible to deal with all of them. In the same time, we carried out some segmentation procedures in simulation without using the CDT. They gave better results. One conclusion we can draw from this experience is that teams who used the CDT got more problems than teams who decided not to use it.

4 Behaviors

4.1 Special features

In RoboCup 2000, our team added some special features to the behaviors of the different players. Like other teams we developed a kind of shoot where the robot clears the ball by plunging forward with both front legs stretched. Moreover, the goalkeeper was given the capability to drop on the floor with legs stretched sideways to stop a ball that would go too close and too fast towards the goal. However this technique was not the best one against robots that made pressure near the goal. In addition, it was not reliable since it was difficult to evaluate the speed of the ball because of vision confusion. We opted for a better strategy that consisted for the goalkeeper in clearing the ball as soon as the ball crossed the penalty line. The robot should shoot into the ball and keep on clearing it as

long as it was in its field of sight. The goalkeeper was a kind of *flying* goalkeeper. The technique was fairly efficient but the robot sometimes missed the ball when shooting because of bad implementation.

4.2 Localization

In case the robot was lost and could not spot the ball after a certain time, the strategy module was designed to run a localization procedure. This procedure made the robot halt, scan and turn-in-place until capturing 3 different markers for triangulation localization. We used 3 landmarks because we thought that information of distance was not so accurate. However, during the final we disabled this feature since it took too much time. Instead of running the localization procedure, the robot was designed to go back to its own goal to defend. This was very useful since an attacker could assist the goalkeeper. However the rule of *no more than one defender inside the penalty area* penalized us very much. We did not count how many times the referee picked up our defender robots that were coming back to help defending !

5 Conclusion

A lot of work remains to be done in vision. Without significant improvements in this domain, we think that it is not possible to design cooperation between robot partners, unless wireless communication inside the same team is allowed in next RoboCup. Some other teams like CMU have made some significant developments in vision that allow them to achieve very good performances in terms of localization on the field [5]. However, one robot spotting the other ones on the field with enough accuracy is still very difficult to achieve.

References

1. Hugel, V., Bonnin, P., Bouramoué, J.C., Blazevic, P.: Integration of vision and walking skills together with high level strategies for quadruped robots to play soccer. Advanced Robotics, the Int. Journal of the Robotics Society of Japan, Special Issue on RoboCup.
2. Hugel, V.: On the control of legged locomotion applied to hexapod and quadruped prototypes. PhD thesis (in French), November 30, 1999.
3. Hugel, V., Blazevic, P.: Towards efficient implementation of quadruped gaits with duty factor of 0.75. Proceedings of ICRA 1999, Robotics and Automation, Detroit, May 10-15, 1999.
4. Hugel, V., Bonnin, P., Blazevic, P.: Using reactive and adaptive behaviors to play soccer. AI magazine. Vol. 21 – 3. Fall 2000.
5. Lenser, S., Veloso, M.: Sensor resetting localization for poorly modeled mobile robots. Proceedings of ICRA 2000, Robotics and Automation, San Francisco, 2000.

Team ARAIBO

Yuichi KOBAYASHI

The University of Tokyo

1 Introduction

Last year, we introduced heading and diving action to the robots. But these actions were actually carried out few times. There were two main problems. First, the measurement of the ball position was not accurate enough to stop in front of the ball. Second, it took much time to actuate these actions because the motion arbitration was not successfully achieved. Thus, we focused on developing ball recognition, walking motions, and ball operations. We proceeded to the quarter final.

2 Team Development

Team Leader: Tamio ARAI (Professor)

Team Members:

Hideo YUASA (Assistant professor)

Yuichi KOBAYASHI (2nd degree of doctor course)

Masahiro YOKOI(2nd degree of master course)

Takeshi FUKASE(1st degree of master course)

Ryuichi UEDA (graduate student)

– The University of Tokyo

– Japan

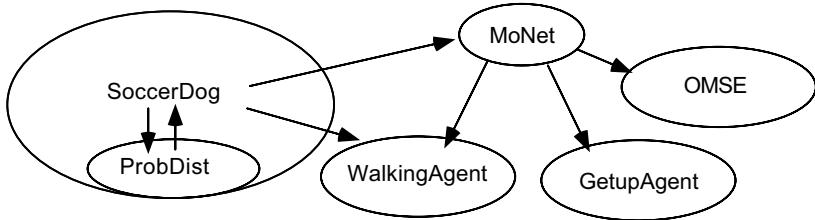
– Attended the competition except for YUASA and UEDA

Web page <http://www.arai.pe.u-tokyo.ac.jp/>

3 Architecture

The main idea of our robot architecture is , to approach to the ball as fast as possible, and to hit the ball to the goal. **Fig.1** shows the summary of objects. The sensory interface module is based on the original object ‘SoccerDog’. It has a smaller object called ‘ProbDist’, which restores the information to decide actions. The main informations are probability distribution of the ball and the position of the robot itself, and expressed by 2D-grid.

When action is decided by the object SoccerDog, it gives the commands to the object ‘Walking Agent’, ‘Getup Agent’ and ‘OMSE’. These objects deal with motion of the legs(walking), all of the body(heading and getting up), and the head, respectively. In case of WalkingAgent, command signals are send not only from MoNet but also directly from SoccerDog. This information flow was generated in order to control walking motions speedily. This function is important especially when the robot needs to stop in front of the ball before touching it.

**Fig. 1.** Summary of object

4 Vision

Color calibration We used the color detecting hardware for color recognition. We must give threshold values for dividing U,V space for each Y value. Our approach consists of two stages.

1. Trim the object region by edge detecting.
2. Optimize the threshold value that maximize the evaluation function.

Here, we defined the evaluation function

$$f_Y(U_{\min}, U_{\max}, V_{\min}, V_{\max}) = N - \alpha M, \quad (4.1)$$

where N denotes the number of pixels which succeeded to recognize as the correct color and M denotes the number of pixels which failed. Optimization of threshold values is achieved by controlling the parameter α .

Edge detection needs some parameter tuning that depends on the lighting conditions, but that is much easier than tuning threshold parameters directly. With the note PC(PentiumIII 500MHz), the optimization algorithm needs about 5 minutes per one color.

Object recognition When the ball is far enough, the result of Color detector can be used to measure the position of the ball. But when the ball is near and it covers much part of the image, the result of the color detector is not accurate any more. We adopted the least square method for quadratic curve to infer the center and radius of the circle. The least square method falls into the eigen value problem of 4×4 matrices in this case. With this calculation and the transformation of coordinates, the position of the ball in front of the body can be measured by the error of 5[mm] within 0.2[sec].

Goal recognition is applied for goles. The robot must get back to the own goal when it has cleared the ball from neighborhood. For the recognition of the own goal, the edge between the field and the goal is extracted. The robot aims to the midpoint of the line segment, and turns to the front when the line has got near enough.

5 Localization

We adopted the 2D grid based localization(**Fig.2**). The robot memorizes the relative positions of landmarks, while updating the dead reckoning information. The robot localizes itself when it gets near enough to the ball. In other words, self localization is needed only before the heading actions. Existence probability of each grid, which corresponds to the (x, y) position, is calculated by means of these landmark positions. Among grids with highest likelihood, one grid is selected based on the last seen landmark. The posture of the robot, which corresponds to θ , is also calculated by the last.

If it has not enough information of landmarks, it takes observation action. The information of the observed landmarks is lost when the inconsistency between landmarks is detected or when the information gets old.

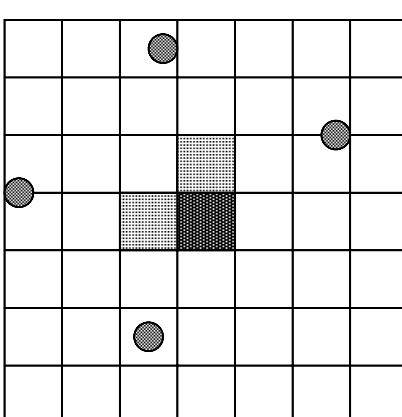


Fig. 2. 2D grid for localization

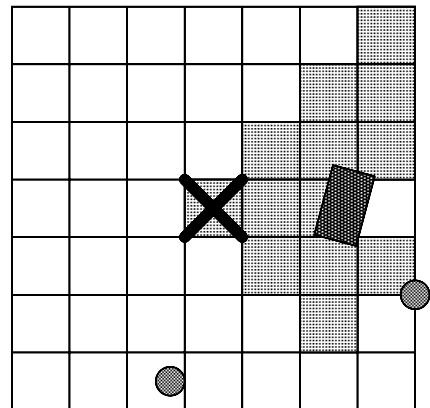


Fig. 3. Ball estimation

Fig.3 shows how to estimate the position of the ball. When the robot does not see the ball, it turns at the same place. While turning, it estimates the region without any ball. The center of the grids indicates where the robot is. In this case, the robot turned 90[deg]. If the robot has seen another robot, it could not specify the region over it. The landmark information is also used to specify the region where the ball apparently does not exist.

6 Behaviors

We implemented two roles for the robots, goalie and attacker. The algorithm for a defender was not implemented. Awareness of other robots was implemented just for the estimation of the ball, not for obstacle avoidance.

7 Action/Walking

We implemented three kinds of walking motions, walk forward, turn right, and turn left. The outline of the motion is informed by the walking module given by SONY. We improved these motions in two ways. First, we tuned control gain parameters and trajectories of feet so as to improve stability and speed. Second, transition from one walking motion to another made the robot motion difficult to estimate. We implemented transition motion from one walking motion to another that has little influence on the unexpected change of the robot posture. Though it took some time to proceed when walking command frequently changes, expected error of dead reckoning was reduced. Walking speed 4.4 [m/sec] and rotation speed 23[deg/sec] were achieved.

Heading actions were also improved by means of inverse kinematics. The main point is to make the trajectory of the head top as forward as possible. If the trajectory is too forward, the robot falls to the front. Our heading action is tuned to hit the ball of the far position as possible without falling down. **Fig.4, Fig.5** shows the outline of the controllable region. The radius of the circle (actually an ellipse) is 210[mm], which indicates that the robot can operate the ball within 210[mm] front of itself to the requested direction. As indicated in **Fig.4**, the ball within 110[mm] can be operated with heading without diving action. Diving action enables to touch the further ball, but less accurate. Kicking actions were not actually implemented because heading actions were more efficient.

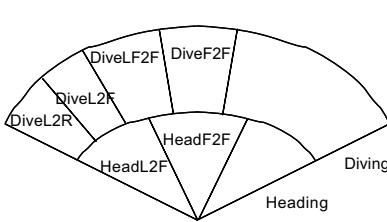


Fig. 4. Forward heading

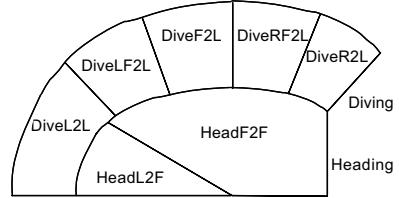


Fig. 5. Left heading

8 Conclusion

Finally, we improved the ball operation ability and realized many heading actions in the game situation. But heading actions were often made in the wrong direction, which brought disadvantage for scoring goals as a result. The reason is that the self-localization was quite simple and not perfect. Next year, we will improve individual and team strategy based on the recognition of other robots, after establishing a proper self-localization ,including line recognition.

This article was processed using the L^AT_EX macro package with LLNCS style

CMPack '00

Scott Lenser, James Bruce, and Manuela Veloso

Carnegie Mellon University, SCS
Pittsburgh, PA 15213-3891
`{slenser,jbruce,mmv}@cs.cmu.edu`

1 Introduction

This is a description of Carnegie Mellon University's entry in the Sony legged league of RoboCup 2000. See our web page for more details [4]. The main components of our system are: vision, localization, behaviors(including a basic world model), and motions. The main changes for this year are: basic world model, new behaviors/behavior architecture, walking and kicking motions. We placed third in the competition, losing only to the first place team.

2 Vision

The vision system is responsible for interpreting data from the robots' primary sensor, a camera mounted in the nose. The images are digitized are color segmented in hardware using color thresholds that are learned offline. The low level vision software then carries out the following steps:

- Connect neighboring pixels of the same color class to make regions.
- Connect nearby regions of the same color class into larger regions.
- Calculate statistics for the regions useful for high level vision.

Nearby regions are merged when the density occupied in their bounding area is above a threshold for that color class. The low level vision uses the CMVision library which is described in more detail in [3] and is freely available under the GPL [2]. After the low level processing is performed, the high level vision carries out the following steps for each type of object of interest (currently these include the ball, goals, markers, and other robots).

- Scan lists of regions for colors that include the object of interest.
- For each region or regions that may form the object, evaluate domain and geometric constraints to generate a confidence value.
- Perform additional filtering rules to remove false positives.
- Take the hypothesis with the highest remaining confidence value.
- Transform the object from image coordinates to ego-centric body coordinates (not yet implemented for the other robots).

We found our vision system to be generally robust to noise and highly accurate in object detection and determining object locations. However, like many vision systems it remains sensitive to lighting conditions, and requires a fair amount of time and effort to calibrate.

3 Localization

Our localization system, Sensor Resetting Localization [6], uses a probabilistic model of the world to estimate the robots location on the field. The robots location is represented as a probability density over the possible positions and orientations of the robot. Since the probability density is in general a very complex function, we approximate the probability density by a set of sample points. The samples are chosen such that if $x\%$ of the samples are expected to be found in a particular area then the probability that the robot is in that area is $x\%$. Each sample point represents a particular location on the field at which the robot might be located. Localization is the process of updating this probability density. To make the computation tractable, we make the Markov assumption that the robots future location depends only on its present location, the motions executed, and the sensor readings. Updates are done in such a way that the expected density of sample points in a region is proportional to the probability of the robot's location being within that region. The localization system automatically resets itself if it notices to high of an error. For more detail including a probabilistic derivation of the algorithm, see the Sensor Resetting Localization paper [6].

4 Behavior System

Our behavior system is a hierarchical behavior-based system. The system is primarily reactive, but some behaviors have internal state to enable behavior sequencing. The input to the system is information about the objects seen (from the vision) and an estimate of the robots location (from the localization). The output of the behaviors is a motion to be executed. The motion can be a type of kick or getup to execute (discrete) or a direction to walk (continuous). The behavior system consists of three interconnected hierarchies for sensors, behaviors, and control. The sensor hierarchy represents all that is known about the world, including a basic world model that tracks objects' locations when they are out of view. The behavior hierarchy makes all of the robot's choices. The control hierarchy encodes everything the robot can do. Our system is similar to the system used by the FU-Fighters small size RoboCup team [1]. We loosened the sensor hierarchy and made behaviors responsible for their own activation levels. This results in reduced coupling between the behaviors/sensors and different behavior levels.

The sensor hierarchy represents the knowledge that the robot has about the world. We divide the sensors into two categories, real sensors and virtual sensors. Virtual sensors represent a processed version of the real sensors that includes information that the behaviors are directly interested in. The virtual sensors serve to aggregate information and act as a model of the world complete with memory. Virtual sensors also avoid computing the same information twice in separate behaviors. The sensor hierarchy has a data structure for storing the sensor values and code to update the virtual sensors in the data structure from

the real sensors. The data structure output by the sensor hierarchy provides the input to the behavior hierarchy.

The behavior hierarchy and control hierarchies are tightly coupled and cooperate to decide on the action to perform. The behavior hierarchy consists of n behavior sets. Each behavior set represents a set of behaviors. The control hierarchy consists of n control sets. Each control set is a data structure representing all the actions the robot could perform. Behavior/control sets at different levels of the hierarchy operate at different level of detail. A behavior set at level k receives input from the control set at level $k + 1$ and the sensor hierarchy. The behavior set decides what action to perform and writes the decision into the more detailed control set at level k . For example, a behavior set contains a behavior for getting behind the ball. This behavior is activated when the control set above indicates it would help. The behavior uses the sensors to find out where the ball is and sets a goal for the next level down to run along an arc that gets behind the ball. Each behavior calculates its own activation and a combinator resolves conflicts to decide which behaviors are actually run. The combinator allows multiple behaviors to be run in parallel but assures that conflicting behaviors are never run together. Each behavior set takes an abstract description of the task to be performed makes it more concrete. The control hierarchy provides storage for the inputs and outputs of the levels of the behavior hierarchy. The lowest level of the control hierarchy is simply the commands to be sent to the motion module.

5 Motions

The motion component allows the robot to walk, kick the ball, and get up after falling. The behavior system requests a walking motion in the form of velocity requests for x , y , and θ , a getup routine, or a kick. The motion system determines the required angles and PID values for each of the joints to carry out the command and executes it as soon as stability allows. A successful walking motion must make the robot travel on the desired path direction, while ensuring that the robot does not fall over. Also, the other motions must transition smoothly to and from walking.

The high level motion system consists of walks with separate stepping patterns (walking forward, turning left, turning right), several kicks for manipulating the ball, and four getup routines, one for each possible fall (front, back, left side, right side). The motion system was constructed as a state machine. The non-walking motions were specified as time variant functions to determine raw joint angles and kinematic targets for the legs.

Our walk is a quasi-static crawl gait. This gait is characterized by having 3 feet in contact with the ground at all times. Quasi-static means that ignoring noise and momentum, the robot can stop at any point in the walk without falling over. Our walk uses a fixed attitude of the body to prevent unwanted oscillation of the head. We constrain feet in contact with the ground to not move relative to the ground. We choose a path for the body to follow and the

locations of the feet are calculated using the offset of the contact point to the center of the body and an inverse kinematic model. We represent the path taken by the body using a Hermite spline [5]. These splines take the initial and target points as parameters, as well as derivatives at each, and generate a smooth trajectory to follow conforming to the control points/derivatives. Every time a foot is picked up, the motion system takes the most recent target walk request from the behavior system. It then determines the current position and velocity of the body (simply by consulting the current spline) and plots a new spline path from the current position/velocity to the target position/velocity one cycle later. Thus we have fully continuous body paths and velocities regardless of the sequence of requests by the behaviors and latency of only a single step before beginning to execute a command. Walk speed ranged from 100mm/sec to 0mm/sec, allowing fine speed control. The path of the feet in the air was represented as a Catmull-Rom spline with points chosen to ensure continuity of horizontal velocity upon foot contact and sufficient elevation to clear the ground.

Overall we found the walk to perform significantly better than other implementations of the same gait at the competition, as well as allowing for a simplified software architecture thanks to high level primitives such as splines. We demonstrated the fastest and most stable walk using the crawling gait. Future goals include adapting our approach to other gaits such as trotting (two feet in the air at a time), which allow much higher speeds but which require dynamic stability.

6 Conclusion

We implemented a highly competent robot team. We plan to incorporate more machine learning into our robots to improve performance, robustness, and development time. We also plan to work on better knowledge of the locations of the other robots. We'd like to thank Sony for providing the excellent robots to work with.

References

1. S. Behnke, B. Frtschl, R. Rojas, et al. Using hierarchical dynamical systems to control reactive behavior. In *Proceedings of IJCAI-99*, pages 28–33, 1999.
2. J. Bruce, T. Balch, and M. Veloso. CMVision (<http://www.coral.cs.cmu.edu/cmvision/>).
3. J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of IROS-2000*, 2000.
4. J. Bruce, S. Lenser, and M. Veloso. CMPack '00 (<http://www.cs.cmu.edu/~robosoccer/legged/>).
5. J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics, Principles and Practice*. Addison-Wesley, Reading, Massachusetts, second edition, 1990.
6. S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of ICRA-2000*, 2000.

The McGill's RedDogs Legged League System

Guillaume Marceau

Center for Intelligent Machines^{**}

1 Introduction

McGill University was back on the Robocup fields for a second year of four legged robot competition. As always, we preferred scalable algorithms and long-term architectures.

This report will describe the aspects of our system which might inspire future work : section 2 discuss the support code, sections 3 and 4 review algorithms and give empirical data of performance. Finally, section 5 on behavior should give a good view at the style of decisions the system was able to make.

2 Architecture

The basic object setup is mostly unchanged since last year. The most profound modifications was the design and implementation of a concurrent control language, named Dogscript, for the highest-level decision making in our architecture. The control structure are similar in spirit to Esterel and has higher-level parameters. The language is typed inferred and type checked offline by a compiler written in OCAML¹ and then translated to a custom bytecode representation. Only a small interpreter runs on the dog. The language suggested a coding style similar to the description of behavior (in terms of actions, criteria and decisions) that is used in section 5 on behaviors. We find that the particular semantic of our language is useful to create generic/reusable libraries of behaviors.

We find critical to be able to stabilize the debugging environment as much as possible. The real-world noise as inputs to the system tend to be amplified into non-deterministic bugs. Moreover, in many cpu critical situations, adding debugging output would break some real-time constraint and render the run unrepresentative. Our solution is to implement fast mass-logging facility. We have created logging streams by overloading the usual stream operators. Every channel is selectively redirectable either to a memory buffered file, to the console, to both or to oblivion. To make redirecting to a memory buffered practical, we implemented a simple compression scheme for recurrent string labels. This allowed us to record up to a minute of comprehensive logging information for offline examination, without noticeable effect on timings.

Parameters of our systems are adjustable on a simple command shell that is shown at boot time and when the Dogscript interpretation pauses. Examples

^{**} <http://www.cim.mcgill.ca> and <http://www.cim.mcgill.ca/~robocup>

¹ <http://caml.inria.fr/ocaml/>

of particularly useful hot-assignable variables are : the name of the color space to use (different for different playing fields), the name of the behavior the use, debugging parameters to modules in development, and, of course, the logging redirections.

```

new segment
new map
new sonywalk
file brainini.cmd ## sources-in the configuration
## for the brain module
set desiredCsp blcknnC.csp
set localizationLog 3
set mapLog 0
set splitvision 0
run

```

Fig.1. An example of the shell-script, the memory-stick file /cmdshell.cmd is executed by the mini-shell to start the system.

All our code is cross-compilable between Cygwin's win32 port of the Gcc compiler and the Green-Hill compiler. On the Cygwin version, many data input paths are dummiied up to be loadable from file. For instance, the vision component can load ppm images and the localization component can load pole-detection logs. This allows us to bring individual runs off-line and reproduce bugs under the helpful eye of gdb. Also, it gives us pentium-class machines to confirm the validity of various parameters – like the color tables – and makes it possible to detect Green-hill's compilation bugs in a reasonable amount of time.

3 Vision

Even though we rewrote the code from scratch, our vision system is essentially the one described in last year's team technical report. Noticeable changes are the usage of slightly faster data structures and a nice interface-packaging which now makes our vision quickly usable in other projects. Recall that as a prime feature, our vision module produces a reliable contact graph between the elements detected in the picture, which give a natural methods of detecting localization poles, the corners of the goals and occlusions (We use occlusion of the sideband by the ball to trigger a special behavior, see sec. 5). Notwithstanding its apparent complexity, our vision sub-system is able to process vision frames at 20hz on a dedicated processor.

Our color table creation cycle is as follow : First we save pictures incoming from the camera to a link-list in memory. When the available memory is depleted, we save them to the memory-stick. During this saving process, we can always go on the field with another dog and take more pictures. In general we can take about 800 training pictures in 15 minutes. This time can probably be improved by using run-length compression on the images.

Once the pictures are downloaded to a PC, we use a java-gui tool to hand-pick about 30 representative pictures and mask them. Since 30 pictures is by far too few to cover the whole space, most Yuv values are assigned color classes through nearest-neighbor interpolation. This setup allows us to spend less time masking and more time doing cross-validation using the remaining hundreds of pictures.

It takes about three hours to achieve a reliable training. The asymptotic error rate is about 1 error per 300 pictures of one ball and two poles, all viewed across the field. As most teams, we are not able to separate the light grey surface of the robot from their dark blue jerseys, from the dark white background.

4 Localization

We implemented the Monte Carlo localization algorithm on the dog. Quickly put, we keep a Monte Carlo sampling of the probability density of our position over the field. Each point in this sampling can be thought of as a weighted hypothesis of the position. In contrast to Markov localization, m.c. is supposed to assign more points (and more computational resources) around areas of high probability.

The data comes for one part from last year's geometric odometry, and for the other part from the vision data. More precisely, we integrate into our distribution any sight of the six poles around the field but also the bottom left and bottom right corners of each goal.

Using 100'000 points, off-line, in non-real-time speeds, we can compute our position to a 2 cm precision. As we lowered the point count below 8'000 points, discretization errors would give topologically incorrect result with increased frequency. We used a point-shaking heuristic to overcome the problem : a Gaussian translation is added to a new point with variance inversely proportional to the average weight of previously computed points. The effect of this heuristic is similar to resetting the distribution on failed localizations, but it scales better to ambiguous situations. Still, an average computational weight of under 2000 points seemed to be required in order for the localization to be practical. By adapting the size of the hypothesis pool according to a heuristic of the quality of the points, we can lower the point rate on frame that have good chances of successfully recovering from discretization error.

On the dog, ± 20 cm localization is achieved in 5 seconds of computation (requiring about 20 images) 80% of the time. Correct localization is always achieved within 15 seconds. Tracking rotation is hardest, followed by tracking of forward motion. We do ± 20 cm tracking at 2/3 the normal forward speed and 1/2 the maximum rotation speed. We believe the main source of noise to be a timestamping problem within Sony's locomotion library which might be related to an Aperios channel communication bug which halves our frame rate.

We found the Monte Carlo algorithm easy to code and to get running. It was essential however, for both debugging and tuning, to take care of writing customized visualization tools to assert the correctness of probability distributions.

5 Behaviors

Our field player is constructed as a pool of high level actions, each of which have a termination criteria. In general, the attacker will alternate between the choice of an action and its execution to completion. In our final version, the top-level decision procedure relied entirely on the localization component.

The first set of actions moves the robot around the ball in order to get better attack angles. They are composed of a servo rotation around the ball followed by a sequence of blind, timed movements. By varying the numbers, we could achieve +90, -90 or 180 degrees rotation around the ball.

The *defaultNailTheBall* action is a simple servo designed to push the ball down-field. When we lose sight of the ball, we go through a search sequence in an increasingly more aggressive manner, which takes advantage that, most of the time, the ball is standing right beside the robot.

We programmed the termination criteria of this action after each pivot : we pause the robot and glance up and strait for the target goal. If it is visible at any point during the glance the robot will continue its attack assuming that it is making some kind of progress. Otherwise it will fall back to the main decision loop. Glancing only after pivots appeared to be sufficient. In practice it was very rare that we would complete a full turn towards our own goal without ever pivoting. Also, it was fairly rare to see two defaultNailTheBall following each other, which means that the criteria is not overly conservative.

Finally, we have an effective corner clearing action which would often place the ball right in front of the target goal. The action itself was very simple : a walkForward followed by a walkLeftForward or walkRightForward, depending on the current localization estimate. To make it work, we had to program a new visual primitive. It computes the percentage of the circumference of the ball which is directly in contact with some large white blob. This statistic turned out to be a very stable indicator of difficult ball cornered conditions, mostly because our noise filtering of the segmented picture is highly reliable.

6 Conclusion

For this year at Melbourne, the McGill Robocup team has shown a workable system that solved the first layers of problem. We presented a way of doing reliable vision, reliable localization and flexible behavior within the relatively tight computational constraints of the Sony-dog platform. In particular, the modules we created have generic nature and could inspire solution for other minimal-robotic problems. By far the weakest links of the system is the absence of motion customized for soccer playing, what upcoming efforts will have to focus upon first.

The quality of the control in the legged league this year was very appreciable. It is clearly mature enough to move into the problems of strategic play and multi-robot cooperations, which has traditionally been the domain of the older leagues.

BabyTigers: Osaka Legged Robot Team

Noriaki Mitsunaga, Yukie Nagai and Minoru Asada

Dept. of Adaptive Machine Systems, Graduate School of Engineering Osaka University, Suita, Osaka, 565-0871, Japan

1 Introduction

Our interests are learning issues such as action selection, observation strategy without 3D-reconstruction, and emergence of walking. This year we focused our development on embodied trot walking and behavior of goal keeper.

We consider that our embodied walking showed the fastest movement in the all twelve teams since we got 2nd place in the RoboCup Challenge 1 and 2, also achieved shortest time in the RoboCup Challenge 3 in spite of our stop-observe-act approach.

2 Team Development

Team Leader: Minoru Asada

Team Members:

Minoru Asada

- Osaka University
- Japan
- Professor
- Attended the competition

Noriaki Mistunaga

- Osaka University
- Japan
- Ph.D candidate
- Attended the competition

Yukie Nagai

- Osaka University
- Japan
- Ph.D candidate
- Attended the competition

Web page <http://www.er.ams.eng.osaka-u.ac.jp/>

3 Architecture

The robot control is based on an object oriented OS, and updating frequencies of the sensors and the camera are different. There are two approaches to handle the different frequencies. One is to prepare only one object (program) and absorb the difference by the internal buffers. The second one is to prepare several objects for different purposes (actuators, sensors, vision, and decisions) and connect each other by inter-object communications. Although, it is more complex than the former, it is more easy to implement timing critical motions. The former is adopted for the goal keeper and the latter for the attackers.

The goal keeper consists of one object. It receives the data from sensors and the camera, makes decisions, and move actuators.

For attackers, we composed four objects. One is for the vision, one for the walking and head movements, one for the sensors, and one for cognition and decision making. The inter-object communications are done through shared memories which are also used for absorbing the difference in frequencies. With this architecture timing sensitive parts, for example pusedo velocity control, are easily implemented.

4 Vision

In the RoboCup Legged Robot League field [1], seven colors (aqua-blue, yellow, orange, blue, red, pink, green) are used and robots need to detect and discriminate them. The SONY's legged robot has the color detection facility in hardware that can handle up to eight colors at frame rate. To detect colors with this facility, we need to specify each color in terms of subspace in *YUV* color space. *YUV* subspace is expressed in a table called Color Detection Table(CDT). In this table, *Y* are equally separated into 32 levels and at each *Y* level we specify one rectangle $(u_{\min i}, v_{\min i}), (u_{\max i}, v_{\max i})$ ($i = 1, \dots, 32$).

In order to make CDTs, we used the same method as in previous years [2] [3]. Briefly,

1. take an image which includes the object to detect with the camera of the robot,
2. specify pixels to detect with GUI program and make a list of YUV to detect,
3. order the program to classify each pixel according to the *Y* level as they are classified in CDT and make a bounding box of UV in each level,
4. check if detection satisfies the need and if not do 1) again.

Iterate these procedure for each color with several images.

After the color detection with the hardware, in the goal keeper program, we used the API which reports the size, centroid, bounding box of colors in a color detected image. The calculations are also done by the hardware. The discrimination between landmark poles which has yellow (aqua-blue) color and a yellow (aqua-blue) goal is done with if there is a blob of pink color (which

indicates a landmark) or not. The goal is so large enough that the color of landmarks rarely affect the centroid and size of the yellow (aqua-blue) goal.

The color detection of the attackers was followed by extraction of connected areas with 8-neighbor method in a color detected image by software. The extraction is done in one pass, in which each pixel is only checked once in a image. After connected areas are extracted, object recognition, including landmarks (all landmarks are consisted of two colors and recognitions are done by concatenating the two color areas) are done. To overcome the problems due to noises in the image, the order was determined empirically. Details of 1)-4) and objects recognition are described in [2].

We had some difficulties in discrimination of yellow and orange colors. Due to the lightning conditions and the shadows, the colors of yellow and orange objects sometimes partially were recognized as a same color. There were two options to make CDTs for the robots which use the software extraction of connected areas. One was to make CDTs which does not include common regions between the colors, and the other was to make CDTs which include all the common regions since our extraction program rejects the pixels detected in more than one color. We decided to use former approach, because we could use the same CDTs between hardware and software extraction method, and because it is difficult to specify all the common regions in colors while making CDTs.

5 Localization

Global positioning is useful to compose the behaviors of robots in game-like situations. However, it is not necessary to have position/posture expression in geometric form, and other form of expressions might be suitable for robots. So, we did not use explicit localization, instead we only used the relative direction of the ball, landmarks, and goals.

6 Behaviors

We had two attackers which also served as defenders. One of our attacker's basic behavior was, 1) to try to find the ball, 2) to go near the ball if it could find it, 3) to look around and determine which direction to push the ball according to the relative direction of goals and landmarks, 4) to turn around the ball while watching it until it comes to the desired position, 5) to push the ball and do 2). The other attacker did just try to find the ball and chase it. We expected the first one to push the ball accurately, the second one not to give it to opponents. They only used the camera and did not use sound sensors. We prepared the behavior to avoid other robots depending on the color of robots. Unfortunately we could not make CDTs properly, therefore our robots sometimes hit others.

The goal keeper did the followings; 1) to try to find the ball and watch it in front of own goal until the size of the ball in the image becomes large, 2) to go near the ball, 3) to do the diving motion, 4) to go back to own goal. It used both the camera and the infra-red distance sensor.

7 Action/Walking

We developed a walking program by ourselves for attackers, and we used the one provided by SONY for the goal keeper. Since it is important to move fast and efficiently, we developed an embodied trot walking, which enables simple controlling program and fast movement. Last year we only used walkings based on forward movement. This year we tuned the forward movement and developed walkings based on sideway movement. This enabled to turn around the ball watching it.

We did not prepare kicking motions for attackers but prepared the diving attack motion for the goal keeper. The motion is to dive from the body to the ball utilizing the gravity. For the posture control we used getting up program provided by SONY. We composed the program to sense of falling down, and called the SONY's recovering program when needed.

8 Conclusion

We implemented an embodied trot walking and showed fast movements. The stop-observe-act approach showed steady movements in RoboCup Challenge, but in the competition it seemed that the time for the observation was too long. We did not use the teaching method of last year because of the lack of time for teaching but we would like to incorporate some learning structure. There were times when the defenders saw the ball and the goal keeper could not see it. Team work is one of our future issues.

References

1. M. Veloso, W. Uther, M. Fujita, M. Asada, and H. Kitano. Playing soccer with legged robots. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 437–442, 1998.
2. N. Mitsunaga, M. Asada, and C. Mishima. Babytigers-98: Osaka legged robot team. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*, pages 498–506. Springer, Lecture Note in Artificail Intelligence (1604), 1999.
3. N. Mitsunaga and M. Asada. Babytigers-99: Osaka legged robot team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 762–765. Springer, Lecture Note in Artificail Intelligence (1856), 2000.

S.P.Q.R.

D. Nardi, C. Castelpietra, A. Guidotti, M. Salerno, and C. Sanitati

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113 - 00198 Roma, Italy

1 Introduction

The SPQR (Soccer Player Quadruped Robots, but also *Senatus Populus Que Romanus*) team was a newcomer of the Sony Legged League in RoboCup 2000. The work started in April 2000, based on the previous experience gained in the Middle Size League [3] and on the collaboration of Artificial Intelligence and Robotics researchers of our Department. Due to the very short time to prepare the team for the competition, we decided to focus on the previously developed software architecture, based on the explicit representation of the knowledge of the robotic agent [1, 2] and on the effective realization of some control primitives, the kick in particular. Given the above constraints, the performance of the team in Melbourne was very satisfactory. In fact, SPQR classified fourth, winning games with more experienced teams, playing very tightly with the 99 winners in the semi-final, and generally showing a good level of play.

2 Team Development

Team Leader: Daniele Nardi (nardi@dis.uniroma1.it)

Team Members: Luigia Carlucci Aiello, Alessandro De Luca, Daniele Nardi*
(Professors); Claudio Castelpietra *, Alice Guidotti *, Massimiliano Salerno*,
Claudia Sanitati * (students).

Affiliation: same as above

Country: Italy

Web page: <http://www.dis.uniroma1.it/~leggedro>

3 Architecture

In order to integrate deliberation and reactivity [4], we decided to build for our system a hybrid architecture with two layers [1]. In Figure 1 the functional modules forming our architecture are shown. A Perception module is involved in analyzing sensor data (especially from the camera and from the head and leg sensors): it stores and updates the internal world model. A Deliberation module receives the world model from Perception. It consists of a plan execution monitor, which controls the plan of actions to be executed, and of a set of primitive actions. These actions are then translated into commands either to our

* Attended to RoboCup 2000.

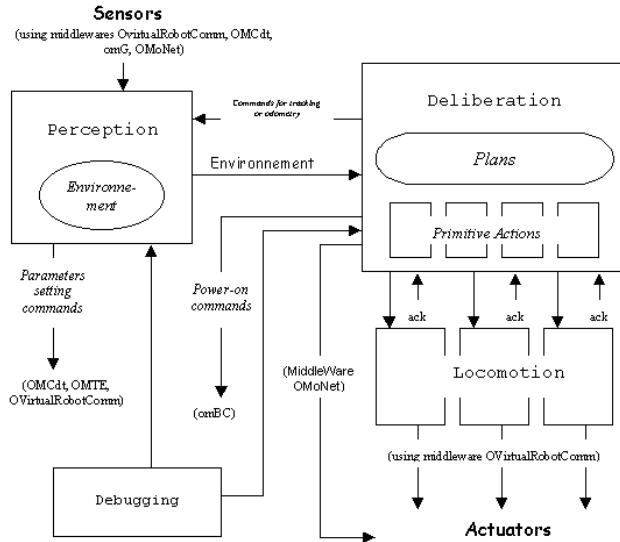


Fig. 1. Architecture

Locomotion module or to the OPEN-R middleware, OMOnet [5]. Locomotion is a module enclosing a set of four objects, each of which implements a different kick; this module provides an interface to the physical functionalities of the robot: it accepts abstract commands from the Deliberation module and translates them into effective commands to the actuators. A further module, named Debugging, is responsible for accepting debugging requests from the keyboard (through the serial connection to the robot): it allows printing sensor data on a terminal or storing images from the on-board camera in the memory stick.

The two-level layered architecture is based on a different representation of the information: there is a high-level symbolic representation, where knowing the exact position of the objects on the field is not necessary, and a low-level numerical representation, which instead is based upon the exact co-ordinates of the various objects. In the Deliberation module, there are: high-level plans, where the decisions on the actions to be executed are based on abstract conditions, such as *KnownBallPosition* or *NearBall*; primitive low-level atomic actions, that allow for an immediate reaction to unexpected situations and a certain amount of damping of the typical uncertainties of a dynamic environment.

4 Vision and Localization

Our vision system is based on AIBOs' hardware recognition and on a software module developed to build colour tables. The technique we used is to select, from a collection of pictures taken by the cameras of our robots, those areas containing the objects of the desired colour. We then check the Y, U, V value of

the pixels therein and set the extremes of the U and V ranges for each different value of Y, using an algorithm that aims at reducing, but cannot avoid, the intersections between colour tables.

Our localization is currently based on the recognition of two landmarks: the goals. This landmark recognition is done only when certain actions are being executed. Thus, there is no continuous localization and, at present, the robot does not determine its exact coordinates on the field. The objects' coordinates (ball and goals) are relative to the robots.

5 Behaviors

A Plan Execution Monitor is in charge of the correct execution of the actions composing the plans. In the monitor's implementation, a plan is stored as a graph data structure. The monitor's task is that of visiting the graph, calling and suspending the appropriate actions.

Each primitive implemented action is a C++ class derived from an abstract base class, named *pemAction*². All the implemented actions have then basically the same structure. Each plan is an object of the *pemPlan* type, which is also a class derived from *pemAction*. This strategy allows plans and actions to be treated in the same manner, combining them in plans of higher hierarchical order. This approach has the advantage of being modular, extensible, readable and reusable. Moreover, as the plans are composed in growing hierarchical order, it is possible to design them leaving aside the robot physical platform or the implementation of the lower plans. In other words, each level uses the functionalities offered by the underlying levels, without caring about their implementation's details.

Furthermore, a plan selector allows the monitor to choose the current plan to be executed. In fact, besides the normal playing actions, the robot must deal with a set of particular situations, such as the initial kick or rising from a fall. These situations are managed by a set of specific plans, each of which has an associated condition. The monitor's task is that of verifying these conditions and of activating the higher priority plan whose activation condition is valid.

A library of boolean functions, based on the world model received from Perception, represents the abstract conditions that allow the state transition or the plan activation. These functions are based on the local co-ordinates of the ball and of the goals, and on the reliability of this information. A hysteresis mechanism allows the decisions to be stabilized [2].

We implemented more than thirty primitive actions: head, tail and leg movements are treated in separate primitive actions. For instance, a primitive action is in charge of all the head searching movement: the class constructor of this action takes, as a parameter, the type of search that is needed and, on the basis of this parameter the appropriate commands are sent to the head. Another kind of atomic action manages all the commands that can be sent to the tail. More complex are the actions involved with the leg movements. They are based on

² pem stands for "plan execution monitor".

the local co-ordinates of the ball and goals. For instance, a *GoToBall* action calculates the angle between the ball and the robot direction and, depending on this angle, sends the appropriate movement commands to the robot: if the angle is small the robot must move straight forward, otherwise it must go to the right or to the left according to the angle's sign.

The primitive actions are combined together to form plans at different levels of abstraction. In a higher level plan, external conditions determine the plan to be activated next. We have three main high level plans, distinguished according to the roles of goalkeeper, defender (back) and forward.

Our robots have static co-ordination: this results from the differences between the plans for the various roles. In particular, the goalkeeper remains in front of its own goal and leaves it only if the ball is close enough, the defender tends to return to a backward position, while the forward attacks over the whole field. This strategy avoids interference among the robots and, most important, the execution of the same action at the same time, which would generally be counterproductive [2].

6 Action/Walking

We use the oMoNet service for walking, with a combination of default and fast walk, and kicking actions made by us. We developed two main kind of kicking actions: one using the front legs and the other using the head. In order to design the leg kick, we analyzed the cinematic of the leg and tried to maximize the speed in the motion direction. We then introduced this kicking action in a particular phase of the walking movement and we overlapped the kicking command to the walking command in order to make the action more effective through the help of the main-body push. The head kick can be frontal or lateral. Both these actions have been developed mainly by experimental tests.

7 Special Team Features and Conclusion

The main focus of our work has been the robots' architecture and the decision making system. In addition, we have designed several effective kicking actions.

In the future we intend to develop a more systematic approach to localization, to a robust vision system and to robot coordination.

References

1. Luca Iocchi. *Design and Development of Cognitive Robots*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1999.
2. D. Nardi. Artificial intelligence in robocup. In *Procs. of ECAI 2000*.
3. D. Nardi et al. ART-99: Azzurra Robot Team. In *Procs. of 3rd RoboCup Workshop*.
4. A. Saffiotti. Some notes on the integration of planning and reactivity in autonomous mobile robots. In *Procs. of the AAAI Spring Symposium on Foundations of Automatic Planning*, pages 122–126, Standford, CA, 1993.
5. Sony Corporation. *OPEN-R Software Programer's Guide*, 2000.

The University of Pennsylvania RoboCup Legged Soccer Team

Jim P. Ostrowski, Ken A. McIsaac, Aveek Das, Sachin Chitta, and Julie Neiling

University of Pennsylvania

1 Introduction

This was a “building” year for the ***UPennalizers***, where we focused on developing a solid foundation of interconnected modules that could be easily customized and upgraded as our team progressed. We suffered two hard fought losses in early competition, and so did not make it to the playoff rounds. We worked on developing our own walking routines— at first, relying on quasi-static stability, and more recently moving towards fully dynamic walking gaits. We also were the only team to develop an audio communication protocol that could be used to transfer small pieces of information (3-4 bits per second). Lastly, we developed new visual tracking routines that allowed for greater flexibility in how we track the ball and perform localizations.

2 Team Development

Team Leader: James P. Ostrowski

- Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, United States
- Assistant Professor
- Attended the competition

Team Members:

Aveek K. Das, Kenneth A. McIsaac, and Sachin Chitta

- Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, United States
- Graduate Students
- Attended the competition

Julie Neiling

- University of Evansville, United States
- Senior
- Attended the competition

Itamar Drexler, Caroline Close, Amir Give’on

- Computer Science and Engineering, University of Pennsylvania, United States
- Undergraduate Seniors (and Master’s student)
- Did not attend the competition

Max Mintz

- Department of Computer and Information Science, University of Pennsylvania, United States
- Professor
- Did not attend the competition

Web page ftp://ftp.cis.upenn.edu/pub/extra/RoboCup99/public_html/index.html

3 Architecture

In [2,3] we presented our hierarchical structure for control of autonomous robots. Here, we review the key ideas and describe their implementation in the RoboCup-2000 competition. Figure 1 shows the top level overview, along with a schematic of the main logical sensing and actuation blocks that are used. The major blocks are the *planner*, the *extended logical sensor module (XLSM)* and the *extended logical actuator module (XLAM)*.

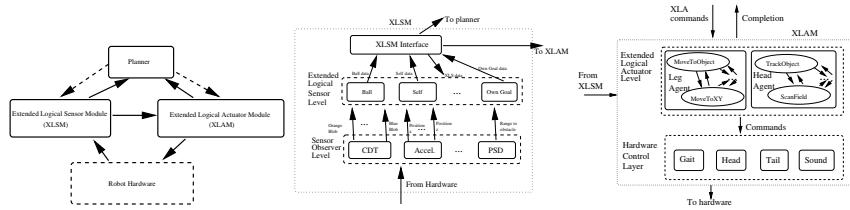


Fig. 1. (left) Top-level view of overall structure. Arrows indicate data flow; dashed arrows indicate commands. (center) XLSM (right) XLAM

The XLSM is implemented in the form of shared data objects available at run time to all the agents in the system. Task synchronization is accomplished through the use of a *Mutex* (mutual exclusion) object. There is one mutex associated with each shared data object that must be claimed before accesses to the global copy. Deadlock is prevented by the technique of allowing ownership of only one mutex object at a time. As a pleasant side benefit, the RS-232 output stream can be properly synchronized across threads.

The XLAM consists of all the application-domain primitives the dogs provide. Since head and leg movements are independently actuated, different agents are defined for locomotion (with modes *Move To Position*, *Move To Object*, etc.) and for head control (with modes *Scan Field*, *Track Object* etc.). The XLA for the soccer application are implemented as simple closed loop control laws, using XLS data as sampled quantities, and can be hierarchically composed of sub-modes.

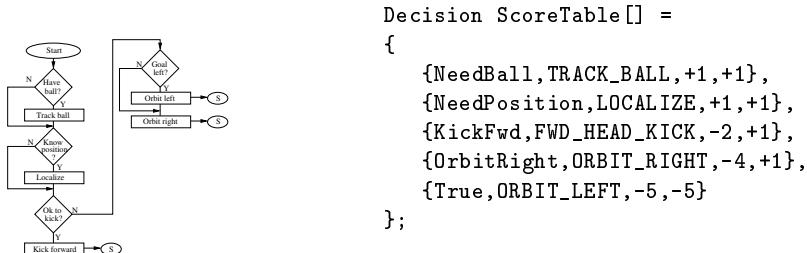


Fig. 2. A simple scoring task implemented using the *PlannerTask* formalism. The *Decision* table (left) exactly reproduces the flow diagram (right).

The *planner* module performs the system's high-level decision making. Based on state information and input from the XLSM, the planner module sends high level commands to the XLAM. Planners are designed exclusively in the application domain and should be re-usable across a broad range of hardware platforms. Frequently performed planner tasks (such as finding the ball, localization and so on) are captured as *PlannerModes*. In our Robocup Challenge

competitors, these switches took the form of the ***Planner Task*** (see Figure 2), a state-table driven decision tree, is used to allow for rapid development of behaviours by recombining the ***PlannerMode*** primitives in different ways.

4 Vision

Object recognition method for a ball, golas, poles

For vision, we focused on interface improvements and added a visual servoing routine for color tracking [1]. We model the robot's head and neck as a two degree-of-freedom manipulator with the camera as the end effector. We do not use the roll axis on the head. We then use the image *Jacobian* to capture the relationship between small changes in image feature parameters (centroid coordinates (u, v)) to small changes in the manipulator joint positions (angular velocities, $\dot{r} = (\omega_x, \omega_y, \omega_z)$).

This leads us to a systems of the form:

$$\dot{\mathbf{f}} = \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -\frac{uv}{\lambda} & \frac{\lambda^2 + u^2}{\lambda} \\ \frac{-\lambda^2 - u^2}{\lambda} & \frac{uv}{\lambda} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} = \mathbf{J}\dot{r}, \quad (1)$$

where \mathbf{J} is the image Jacobian. We can calculate the J^{-1} and control the joints via \dot{r} to track (u, v) to a desired position (u^d, v^d) in the image using the following:

$$\dot{r} = \mathbf{KJ}^{-1} \begin{bmatrix} u^d - u \\ v^d - v \end{bmatrix}. \quad (2)$$

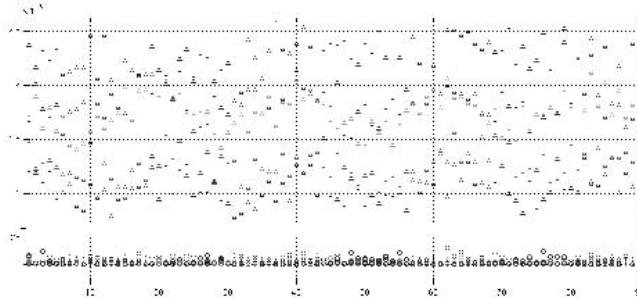
We have successfully implemented the above visual servo control technique to track the ball (or any other colored object) on the AIBO platform. Since we look at velocities of image points, the tracker is able to catch up even when the object being tracked goes out of view momentarily.

5 Action/Walking

The team decided to use a ***quarter phase, $\frac{3}{4}$ duty cycle*** gait. The gaits designed always had 3 legs in contact with the ground, with the three legs defining a support tripod. We showed that the phasing of the walking can be changed to allow the robot to be quasi-statically stable for any of its gaits. This need not be done only for straightforward linear or pure rotational motions. One of the situations that occurred frequently throughout the game was when the dog is a slight distance from the ball but is facing a direction perpendicular to the length of the field. This means that the dog has to walk around the ball and turn so that it ends up facing the opponent's goal. The solution, ***orbiting gaits***, were achieved using a combination of the translational and rotational gaits. The motion is achieved by solving the necessary motions to have each leg generate a combined translation and rotational effect on the body.

6 Special Team Features: Audio Communication

One of the unique features of our team was the ability of the robots to communicate with each other using sound. To implement sound detection, we chose ***quadrature detection*** over a more broadband approach such as DFT or FFT's, because of its speed of processing. Quadrature detection can be used to determine if a signal of a particular frequency is present.

**Fig. 3.** Sound data

For communication, we implemented a binary number method, which sends 3-4 bits of data simultaneously, using multiple carrier frequencies. We used a fast method to play multiple frequencies concurrently, by pre-computing the 1000 samples for each frequency and then adding them together (and scaling as necessary to comply with the maximum amplitude) before sending them to the speaker.

Figure 3 helps depict how the process of choosing thresholds was executed. In this example we used the frequencies of 4000, 4500, 5000, and 6000Hz. The calculations were done on 90 sets of 64 samples for each file to give a wide range of results to evaluate. The triangles represent the results from the samples that contained the 5000Hz signal, and the circles those that did not contain the signal. In this figure there is a clear break between the results from the files that contained the signal and those that did not. For detection, we receive 1000 samples of data every 32 msec from the stereo microphone. Since we don't want to check for sound that often, we only look at the first 64 samples every fourth set of data from the microphone, or once every 0.1s.

7 Conclusion

Contributions in RoboCup-2000 We felt that our main contributions to RoboCup2000 were the development of audio communication, a sound modular architecture (testing of many modules working together), and a very stable, but easy to develop set of walking algorithms.

Future work for RoboCup-2001 We will be entering a team in next year's competition, and will focus on three areas:

- Localization
- Parameterized, dynamic, high-speed walking, and
- Team coordination through communication.

References

1. S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, May 1996.
2. K. A. McIsaac, A. K. Das, J. M. Esposito, and J. P. Ostrowski. A hierarchical structure for hybrid control applied to AIBO robotic soccer. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1020–1025, Japan, November 2000.
3. J. P. Ostrowski. Steering for a class of dynamic nonholonomic systems. *IEEE Transactions on Automatic Control*, 45(8):1492–1498, August 2000. Special issue on Mechanics and Nonlinear Control Systems.

Team Sweden

A. Saffiotti¹, M. Boman³, P. Buschka¹
P. Davidsson², S. Johansson², and Z. Wasik¹

¹ Center for Applied Autonomous Sensor Systems
Dept. of Technology, Örebro University, S-701 82 Örebro, Sweden

² Dept. of Software Eng. and Computer Science
Blekinge Institute of Technology, S-372 25 Ronneby, Sweden

³ DSV, Department of Computer and Systems Sciences
Stockholm University and Royal Institute of Technology, S-164 40 Kista, Sweden

1 Introduction

“Team Sweden” is the Swedish national team that entered the Sony legged robot league at the RoboCup ’99 and RoboCup 2000 competitions. We had two main requirements in mind when preparing our entries:

1. The entry should effectively address the specific challenges present in this domain; in particular, it should be able to tolerate errors and imprecision in perception and execution; and
2. it should illustrate our research in autonomous robotics, by incorporating general techniques that can be reused in different robots and environments.

While the first requirement could have been met by writing some *ad hoc* competition software, the second one led us to develop principled solutions that drew upon our current research in robotics, and that pushed it further ahead.

2 Team Development

The work was distributed over three universities in Sweden, in the cities of Örebro, Ronneby, and Stockholm. These cities are separated by a geographical distance of up to 600 Km, which made the project organization especially demanding. In return our team work, started with RoboCup ’99, created a successful cooperation framework, which went beyond the RoboCup experience.

Team Leader: Alessandro Saffiotti (asaffio@aass.oru.se)

Team Members: include the authors of this paper, plus eight undergraduate students: M. Karlström and K. LeBlanc from Örebro University; M. Broberg, I. Bergmann, J. Johansson, P. Johnsson, R. Krejstrup, B.M. Lindberg, and B. Smeds from Blekinge Institute of Technology.

Sponsors: the Swedish KK foundation, Qualisys AB, Örebro University, and the Blekinge Institute of Technology provided material and financial support.

The Team in Melbourne was represented by six members: J. Johansson, P. Johnsson, K. LeBlanc, B.M. Lindberg, A. Saffiotti, and Z. Wasik.

Web page: <http://www.aass.oru.se/Living/RoboCup/>.

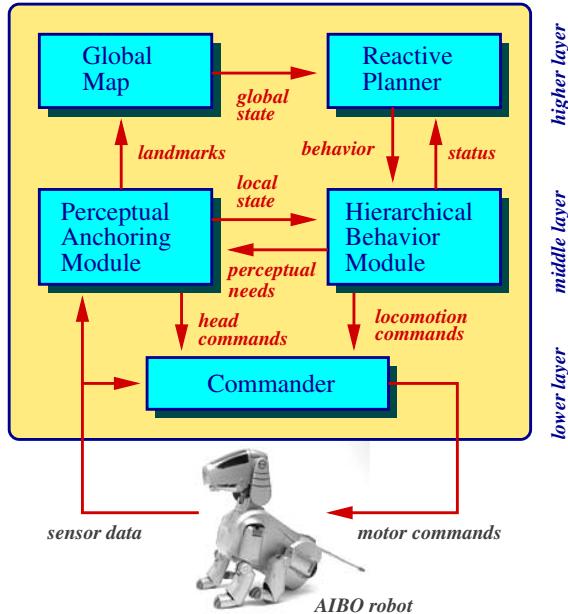


Fig. 1. The variant of the Thinking Cap architecture used by Team Sweden.

3 Architecture

Each robot is endowed with a layered architecture, inspired by the Thinking Cap architecture,¹ sketched in Fig. 1. The lower layer provides an abstract interface to the sensori-motoric functionalities of the robot. The middle layer maintains a consistent representation of the space around the robot (PAM), and implements a set of robust tactical behaviors (HBM). The higher layer maintains a global map of the field (GM) and makes real-time strategic decisions (RP).

4 Vision

The locus of perception is the PAM, which acts as a short term memory of the location of the objects around the robot. The position of each object is updated by a combination of three mechanisms: by *perceptual anchoring*, whenever the object is detected by vision; by *global information*, for the static objects only, whenever the robot re-localizes; and by *odometry*, whenever the robot moves.

The PAM also takes care of selective gaze control, by moving the camera according to the most urgent perceptual needs. Current perceptual needs are communicated to the PAM by the HBM in the form of a degree of importance attached to each object in the environment. The PAM uses these degrees to guarantee that all currently needed objects are perceptually anchored as often as possible. (See [4] for details.)

¹ The autonomous robot architecture based on fuzzy logic in use at Örebro University: see <http://www.aass.oru.se/~asaffio/Software/TC/>.

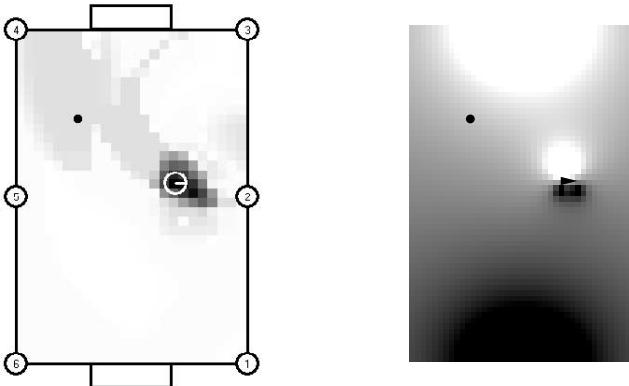


Fig. 2. Left: fuzzy position grid, including a black circle representing the ball location. Right: the corresponding electric field (white is positive).

Object recognition relies on the color detection hardware in the Sony robot, to which we provide the intended color signatures (produced off-line from samples). We use a model-based approach to combine color blobs into features, and use domain knowledge to filter them. For instance, a green blob over a pink one are fused into a landmark; this is rejected, however, if it is too low over the field.

5 Localization

Self-localization in the Sony legged robot league is a challenging task since: egomotion information is extremely inaccurate due to leg slippage and collisions; landmarks can only be observed sporadically, since the camera is needed for tracking the other objects; and visual recognition is subject to unpredictable errors (e.g., mislabeling). To meet these challenges, we have developed a new self-localization technique based on fuzzy logic, reported in [1]. The main advantages of this technique are: (i) it only needs qualitative motion and sensor models; (ii) it can accommodate sporadic observations; (iii) it can recover from arbitrarily large errors; and (iv) it involves low computational costs.

This technique, implemented in the GM module, relies on the integration of approximate position information, derived from observations of landmarks and nets, into a fuzzy position grid — see Fig. 2 left. To include egomotion information, we dilate the grid by a fuzzy mathematical morphology operator. Using this technique, our robots could maintain a position estimate within $\pm 10\text{ cm}$ and $\pm 5^\circ$ from the true position in average game situations. Localization was done continuously during normal action; stopping the robot to re-localize was only needed occasionally, e.g., in case of major errors due to an undetected collision.

6 Behaviors

The HBM implements a set of navigation and ball control behaviors realized using fuzzy logic techniques and organized in a hierarchical way [3]. As an illustration, the following set of fuzzy rules implement the ‘‘GoToPosition’’ behavior.

```

IF (AND(NOT(PositionHere), PositionLeft))   TURN (LEFT);
IF (AND(NOT(PositionHere), PositionRight))  TURN (RIGHT);
IF (OR(PositionHere, PositionAhead))        TURN (AHEAD);
IF (AND(NOT(PositionHere), PositionAhead))  GO (FAST);
IF (OR(PositionHere, NOT(PositionAhead)))    GO (STAY);

```

More complex behaviors are written using fuzzy meta-rules that activate concurrent sub-behaviors. We used this hierarchical composition strategy to write some significantly complex behaviors, like the “GoalKeeper” one.

Game strategies for the players are dynamically generated by the RP. This implements an action selection scheme based on the artificial electric field approach (EFA) [2]. We attach sets of positive and negative electric charges to the nets and to each robot, and we estimate the heuristic value of a given field situation by measuring the electric potential at the ball position — see Fig. 2 right. This heuristic value is used to select the behavior that would result in the best situation: in our example, performing a “GoBehindBall” would maximize the potential at the ball position.

The EFA can account for motion, manipulation, and information gathering actions in the same framework. Moreover, different strategies can be encoded and tested very easily. For instance, in order to prepare the robots for the three technical challenges, we only had to modify a few charges.

7 Action/Walking

The Commander module implements head movements and kicking actions. It also accepts locomotion commands from the HBM in terms of linear and rotational velocities, and translates them to an appropriate walking style. We relied on the walking styles provided by OpenR. These turned out to be less effective than most of the specialized walking routines implemented by other teams.

8 Conclusion

The general principles and techniques developed in our research could be successfully applied to the RoboCup domain. Fuzzy logic proved beneficial for writing robust behaviors, developing an effective gaze control strategy, and providing reliable self-localization. The electric field approach was a convenient way to encode high level strategies. Our main weakness was the ineffective walking style.

References

1. P. Buschka, A. Saffiotti, and Z. Wasik. Fuzzy Landmark-Based Localization for a Legged Robot. *IEEE/RSJ Intl. Conf. on Intell. Robots and Systems* (IROS), 2000.
2. S.J. Johansson, J. Johansson, and A. Saffiotti. An Electric Field Approach to Autonomous Robot Control. Technical Report 17, Blekinge Institute of Technology, 2000. ISSN 1103-1581. Submitted for publication.
3. A. Saffiotti. Using fuzzy logic in autonomous robot navigation. *Soft Computing*, 1(4):180–197, 1997. Online at <http://www.aass.oru.se/Living/FLAR/>.
4. A. Saffiotti and K. LeBlanc. Active Perceptual Anchoring of Robot Behavior in a Dynamic Environment. *IEEE Intl. Conf. on Robotics and Automation* (ICRA), pp. 3796–3802, 2000.

RoboMutts

Robert Sim, Paul Russo, Andrew Graham, Andrew Blair
Nick Barnes & Alan Blair

Dept. of Computer Science & Software Engineering
The University of Melbourne
3010, Australia
`{nmb,blair}@cs.mu.oz.au`

1 Introduction

A software control system for an autonomous mobile robot must be capable of performing many complex processing tasks in real time. Such tasks may include localising within the environment, recognising objects or safely avoiding static and dynamic obstacles. The focus for our team this year was to build a solid structural base, to be used in future RoboCup competitions. This led us to concentrate on development of effective reusable components such as a robust vision system. The RoboMutts team was a joint venture between the University of Melbourne and RMIT University. This paper focuses on the parts of the system developed at the University of Melbourne.

2 Team Development

Team Leaders: (University of Melbourne) Dr. Nick Barnes & Dr. Alan Blair

Team Members: (University of Melbourne)

Robert Sim, Paul Russo, Andrew Graham & Andrew Blair

– Mechatronics Students (Principal Development Team)

Gavin Baker, David Shaw

– Computer Science Students (Peripheral Development)

Team Leaders: (RMIT University) Assoc. Prof. Lin Padgham

Team Members: (RMIT University)

Simon Duff, Indra Indra, Christian Guttman

– Computer Science Students (Goalie Development)

Web page: <http://www.cs.mu.oz.au/robocup/dogs/>

3 Architecture

Our system model, although based around the Aibo robot, could in principle be applied to other hardware; the simple interfaces between modules means that it is straightforward to abstract away from the hardware level. The system is broken into four modules.

The *Visual Perception* module is responsible for obtaining data describing the world. In this implementation, data is extracted from the robot's digital camera and range-finder. It passes information to the *Body Perception* module which contains the relative positions of unidentified objects in a sensor centred coordinate system. The *Body Perception* module observes the internal state of the robot, including battery level, temperature, gyroscope and joint angles. This module combines information about internal state with visual information to identify visible objects, and express their relative positions in a robot-centred coordinate system. The *Knowledge Representation* module uses the data provided by the *Body Perception* module to maintain a database on all objects in the world, as well as localising with respect to the world. Finally, the *Planning & Execution* module consults the *Body Perception* database for information to construct plans of action. The following sections outline the implementation of our system.

4 Vision

Colour calibration: In the Sony Legged League, surfaces can be uniquely identified by applying labels to colour regions within the spectrum. Hence, colour calibration is required to precisely define colour regions in YUV.

To gain the required information about the YUV colour space, a large set of images which contained colours of interest were collected prior to runtime. To ensure that each colour was well represented, several images of each colour were collected, in different lighting conditions, with different camera offsets. Desired regions of colour were extracted from each image to yield the colour map in UV shown in Figure 1(a).

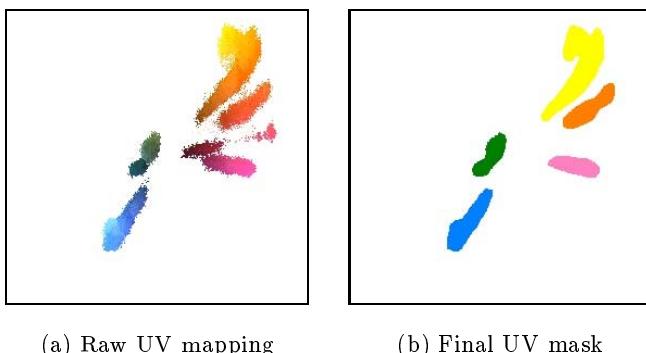


Fig. 1. Creation of the UV colour mask

Image segmentation: Our team has developed a representation that is fast and space efficient. The 3-dimensional YUV space is collapsed down into the UV plane by discarding the Y values. As there is separation between all colours in this representation, it is sufficient to neglect the Y values. By defining the *colour mask* in the UV plane (see Figure 1), segmentation was performed in a manner that is more accurate than the standard hardware segmentation in which a series of cubes is used to define colour regions. Only using U and V is forgiving of the lighting conditions, since different lighting levels affect the Y component more than U and V. Once segmented, a connected components routine is carried out on the images to identify *regions* of colour. Figure 2 demonstrates the result of performing segmentation using the colour map technique.

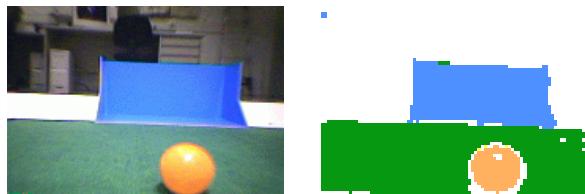


Fig. 2. Actual captured image before and after segmentation.

Object recognition: Objects in the image need to be identified based on colour, shape and position relative to other colours. We developed a set of heuristics for the competition, designed to optimise the trade-off between *false positives* (objects falsely identified) and *false negatives* (observed objects not being identified). False negatives waste computation, as the images are processed, but no useful information is extracted. False positives cause havoc with future calculations relying on the data. Our heuristics are of the form: *if an orange pixel joins a green pixel, the object is the soccer ball*. The use of such heuristics facilitate identification based on connected regions of colour leading to robust, reliable object detection.

5 Localisation

We developed a passive localisation system. *Passive* localisation refers to the process of localising without explicitly looking for landmarks. It is based on the notion that landmarks will come into view when the robot is moving around, or looking for objects, such as the soccer ball in RoboCup.

We adopted a triangulation algorithm to determine the robot's position. Triangulation differs to trilateration in that it uses distances *and* angles to landmarks to calculate position, instead of just distances. This means that position can be uniquely determined by two landmarks, instead of three with trilateration [See [NSPK99] for a trilateration implementation]. In our implementation, we

chose the two landmarks for which the relative position is known to the greatest confidence, according to a data history.

At times, the robot may be in a situation where every entry in the data history is unreliable, in which case it can't perform the localisation calculation. In these cases, odometric data is used to estimate the robot's position.

6 Behaviours

Perfect vision and localisation is of little use without the ability to make decisions based on the knowledge obtained. We implemented a state based decision making strategy that provided reactive, deterministic decision making. We identified a clear set of states for which different robot behaviours would be desirable. Transition between these states were reactive, based on the robot's perception. Reactive behaviour was desirable to ensure the robot was not carrying out a pre-computed plan no longer relevant in the dynamic soccer environment. Due to time restrictions Sony's MoNet was used for motor control.

7 Conclusion

We implemented a flexible framework allowing for further development in future RoboCup competitions. In particular, a robust vision system allowed our robots to see the ball when almost the full length of the field away.

The most obvious area in which further research could focus is development of custom motor control. In the area of vision, one significant improvement would be the incorporation of an edge detection routine, to assist in localisation. The localisation system also has avenues for further development in areas such as walking drift corrections. This could be done through improved odometry, or by some other means, such as using a single landmark to correct pose.

Acknowledgements

We would like to thank the following:

- Sony, for technical support and for generously providing us with a soccer field for development.
- Dr. Andrew Howard, for general knowledge, guidance relating to Autonomous Robots and RoboCup.
- David Hornsby, Andrew Peel and Thomas Weichert, for technical support.

References

- [NSPK99] Luis E. Navarro-Serment, Christiaan J. J. Paredis, and Pradeep K. Khosla. A beacon system for the localisation of distributed robotic teams. Technical report, The Robotics Institute, Carnegie Mellon University, 1999.

Humboldt Heroes

Matthias Werner, Helmut Myritz, Uwe Düffert,
Martin Lötzsch, Hans-Dieter Burkhard

Institute of Computer Science, Humboldt University of Berlin

1 Introduction

Our last year's approach to the Sony legged league suffered from a number of drawbacks:

- The motion system was ways to slow and imprecise.
- The different tasks were not well coordinated. The robot got quite a number of deadlock situations and was not real-time at all.
- The controlling based on a pure reactive approach, that did not allow any real planning.

With this year's code, we tried to overcome some of these disadvantages.

We partly succeed. We get second in group and went to the final round. However, we never had any chance in the quarter final against the cup-dominating UNSW team.

2 Team Development

Team Leader: Hans-Dieter Burkhard

Team Members:

Matthias Werner

- Assistant professor
- Technical leader, also responsible for the run-time system
- Did attend the competition

Helmut Myritz

- Graduate student
- Responsible for AI and self-localization
- Did attend the competition

Andrej Georgi

- Graduate student
- Responsible for visioning
- Did not attend the competition

Uwe Düffert

- Undergraduate student
- Responsible for locomotion
- Did attend the competition

Martin Lötzsch

- Undergraduate student
- Responsible for tool support and locomotion
- Did attend the competition

Web page <http://www.informatik.hu-berlin.de/~mwerner/res/robocup/index.en.html>

3 Architecture

Our soccer software for the ERS 110 consist of 4 active modules, which are supported by runtime system:

- Central control
- Perception unit
- Localization system
- Motion unit

The runtime system encapsulates each module as an Aperios object and activate them in a time-triggered manner. All modules communicate among each other via a shared memory.

All players share the identical program. The information about player's role in the game, direction of attack, etc. are coded in a configuration file which is parsed during the robot's initialization.

4 Vision

Our vision system bases on the color detection provided by Sony. To generate the needed color tables, we use a semi-automatic tool, which allows to construct appropriate filter data for each color on base of a number of sample pictures.

To determine the distance and direction of other robots, we do an iterative pattern matching of the gained data against a projection of a simplified robot model. Since this procedure is quit time-consuming, the search is invoked for close robots only (large red or blue spots).

Unfortunately, our visioning system turns out to be to sensible against small changes of light. Also it had quite a few difficulties to distinguish between yellow and orange.

5 Localization

To solve the problem of localization we decided to use a case-based reasoning approach. In this approach the robot imitates the remembering process of humans.

The robot uses a given case-base consisting of 126 (14 x 9) cases, corresponding to a 14 by 9 grid that covers the pitch. Thus, robot knows for each case the correct position on the field, the angles to all landmarks, and the seen sizes of these landmarks.

With this knowledge the robot can now try to localize itself on the pitch with the granularity of the grid. During the localization our robot has to stand safely, without any movement. His head now scans the whole environment by turning to left an right.

During this scan all information about all seen landmarks (flags and goals) are stored and processed through a CRN (Case Retrieval Net). Finally, the

information is used to activate different cases in different priority. If succeed, the position of the case with the highest priority describes a position which is close to the real position of the robot on the pitch.

During the interception of the ball the robot uses a relative world model, where its position is described relative to the ball. The described localization procedure is invoked when the robot successfully intercepted the ball.

6 Behavior Control (BDI)

For the behavior control we use the BDI approach that was already successful in the simulation league.

The BDI-Model (Belief-Desire-Intention) can easily be described by following steps of a decision process:

- At first the robot tries to collect as much information about his environment as possible (position by CRN, informations about ball and other players) and stores it in its world model (belief phase).
- After that, it searches for all options (e.g.: InterceptBallOption, GoalKick-Option ...) and determines, which of them is the option with the highest priority (desire phase).
- Now, the robot will realize the desired option, by executing a plan assigned to this option (for example: walk 2 steps in front and then kick) (intention phase).

Finally, the robot is ready to handle the new environment informations and the process will repeat. The decision component was invoked two times a second. If a new plan comes up during the execution of an old one, the robot tries to calculate the cost of an abort of the old plan. To stabilize the decision process, every change of a non-finished plan includes a minimum cost, even if the new plan could be easily be integrated.

7 Action/Walking

Thinking of the troubles with not completely known time behaviors, dependencies and parameterizing possibilities of the motions given in OMOnet we decided to implement our own motions.

Therefore, we developed a high level language that can describe a motion net quite similar to OMOnet and a flex/bison based compiler called MotionConfigTool that produces C++ source code from motion net descriptions in that language, which can be compiled and linked into Aibo binaries. The compiler checks a lot of dependencies, such as the existence of transitions between all motions, so that we had a very robust solution for our motions on the Aibo.

Motions can be defined in different ways. Some simple motions (like a Getup or our Headkick¹) can fully sufficient be defined as a small number of vectors

¹ The idea of headkick is got form the ARAIBO team.

with values for destination positions for each joint of the dog and with a time, these positions have to be reached in.

For all walking-like motions (walking and turning in all directions) we had to use a more complex approach. We consider a number of three-dimensional constants or variables like shoulder height or side distance between body and feet. To make the dog move we simulate the moving of the center of gravity of the dog while keeping all feet on the ground and calculate the resulting arcs for all 12 joints.

Of course we have to lift some legs from time to time to make steps. At the moment we make a step by letting a foot move on a half circle from the old position to the simulated position the foot should have a whole step later.

We recognized that leaving all feet on the ground half the time and rising 2 diagonal legs to make a step the other half gives a quite stable walk although the center of gravity cannot be between three legs touching the ground at all times.

Acceleration is not considered in our model yet and because it is hard to simulate it and easier to minimize the occurring divergence by correcting existing parameters, we will probably never implement it.

8 Conclusion

Our team's biggest drawback was the color recognition and blob separation. Thus, next year we will not use Sony's CDT anymore but develop our own color detection system.

Also, we have the intention to improve the following features:

- **Motion system.** Acceleration of the movement, more flexible model to allow non-horizontal positions
- **AI.** Multi-level BDI to allow long-range intentions
- **Self-Localization.** Additional to the CRN, we want to use an odometric system.

Overall, there was a significant improvement in handling the robot, compared with the RoboCup99. We hope that we can continue our work and successful take part in RoboCup2001.

Author Index

- Aaghai, T.H., 583
Abbaspour, A., 397
Abraham, D., 531
Adorni, G., 279, 559
Akhapkin, S., 505
Akita, J., 139
Akiyama, H., 401
Arai, K., 303
Asada, M., 1, 189, 385, 607, 631
Asahara, S., 509
Atsumi, M., 321
Au, G., 285
Auerbach, W., 52
- Babish, M., 41
Bach, J., 405
Balch, T., 1, 575
Baltes, J., 76, 515, 519
Barnes, N., 647
Becht, M., 599
Behnke, S., 239, 547
Benosman, R., 531
Blair, A., 647
Blazevic, P., 615
Bokhove, W., 149
Boman, M., 643
Bonarini, A., 559
Bonnin, P., 615
Bräunl, T., 523
Brüggert, S., 259
Bras, F., 531
Bredenfeld, A., 579
Browning, B., 527, 555
Bruce, J., 623
Brusey, J., 563
Buchheim, T., 599
Buck, S., 169, 567
Burger, P., 599
Burkhard, H.-D., 259, 405, 651
Buschka, P., 643
Butler, M., 119, 357
- Caarls, J., 149
Cagnoni, S., 279
Candea, C., 409
- Casper, J., 339
Castelpietra, C., 86, 635
Chang, M.M., 527
Chen, W., 425
Chiba, R., 321
Chiniforooshan, E., 433
Chitsaz, H., 583
Chitta, S.C., 639
Clemente, G., 559
Cordurié, G., 531
Couder, N., 531
Cristaller, T., 579
- D'Andrea, R., 41
Düffert, U., 651
Dümller, B., 52
Dahlström, A., 413
Dalglish, J., 64
Das, A.K., 639
Davidsson, P., 643
de la Rosa, J.L., 373, 551
de Pascalis, P., 603
Demura, K., 571
Dietl, M., 52
Dorer, K., 417
Dorval, T., 531
Douret, J., 531
Drücker, C., 391, 421
- Ebina, A., 219
Ehrmann, R., 367
Ejima, T., 129, 595
Emery, R., 575
Enderle, S., 291
Enokida, S., 129, 595
Esaki, T., 425
- Fantone, K., 563
Fardad, H., 437
Ferraresso, M., 297, 603
Ferrari, C., 297, 429
Figueras, A., 373, 551
Foroughi, E., 437
Foroughnassiraei, A., 583
Fox, D., 291
Frötschl, B., 547

- Frank, I., 139, 303
Fujita, M., 1
Futamase, S., 425
- Ganguly, P., 41
Garelli, F., 429
Ghaffarzadegan, N., 583
Ghorbani, R., 583
Gollin, M., 405
Goss, S., 285
Grahm, A., 647
Groen, F., 587
Gu, D., 611
Gudarzi, M., 583
Guenther, H., 579
Guidotti, A., 635
Gutmann, J.-S., 52
- Hübner, S., 391, 421
Habibi, J., 433, 437
Hanek, R., 169, 567
Hatami, H., 437
Hatayama, M., 321
Heintz, F., 309, 413
Heinze, C., 285
Hengst, B., 64
Hermes, J., 579
Hetzl, G., 599
Hildreth, N., 76
Hiramoto, M., 449
Hock, T.B., 543
Hoffman, A., 367
Hoffmann, A., 485
Howard, A., 535
Howard, T., 119, 357
Hu, H., 441, 611
Hugel, V., 615
Humble, S., 523
Hunter, M., 441
- Ibbotson, D., 64
Igarashi, H., 315, 445
Iidoi, S., 445
Iizuka, H., 449
Ikenoue, S., 607
Imai, M., 219
Indiveri, G., 579
Innocenti, B., 373, 551
Inui, S., 607
Iocchi, L., 86
- Ishiguro, H., 219
Israel, M., 373
Ito, N., 425
- Jacobsson, M., 413
Jaeger, H., 579
Jamzad, M., 583
Johansson, S., 643
Jonker, P., 149, 587
- Kalmár-Nagy, T., 41
Kalyviotis, N., 441
Kaneda, T., 321
Karimian, P., 437
Kawamoto, K., 595
Kawamura, H., 449
Kawarabayashi, T., 465
Kazemi, M., 583
Khabbazian, M., 433
Khessal, N.O., 539
Kiat, N.B., 543
Kindermann, G., 599
Kitano, H., 139, 209, 269, 351
Klupsch, M., 169, 567
Knipping, L., 547
Kobayashi, Y., 619
Kobiałka, H.-U., 579
Koh, S., 543
Kostiadis, K., 441
Koto, T., 379, 453
Kozhushkin, A.N., 457
Kraetzschnmar, G., 1, 291
Kubo, N., 571
Kubo, T., 461, 473
Kummeneje, J., 309
Kunifugi, S., 501
Kurita, H., 509
Kuwata, Y., 159
- Lönneberg, M., 493
Lötzsch, M., 651
Lafrenz, R., 599
Lau, N., 29
Lawther, M., 64
Lenser, S., 623
Levi, P., 599
Li, B., 611
Lima, P.U., 96
Lorenzetti, M., 603
Lund, H., 1

- Müller, K., 52
Maeda, K., 139
Maire, F., 327
Makies, M., 563
Marceau, G., 627
Marchese, F.M., 179
Marcon, M., 531
Marko, K., 52
Marques, C.F., 96
Martinetz, T., 481
Matsubara, H., 303
Matsui, T., 269, 321
Matsuno, F., 321
McAllester, D., 333, 489
McIsaac, K.A., 639
Meier, D., 367, 485
Meinert, T., 405
Merke, A., 367, 485
Micire, M., 339
Minatodani, J., 595
Ming, Q.Y., 543
Mirrokni, V.S., 583
Mirzazade, M., 433
Mitsunaga, N., 189, 631
Miwa, K., 571
Miyashita, T., 269
Mobasser, F., 583
Modolo, A., 603
Moghaddam, M.E., 583
Montaner, M., 373, 551
Mordonini, M., 279
Mori, Y., 345
Morishita, T., 465, 473
Motamed, M., 437
Muñoz, I., 551
Murakami, K., 345
Murphy, R.R., 339
Murray, J., 199, 469
Myritz, H., 259, 651

Nagai, Y., 631
Nagasaki, Y., 345
Nakadai, K., 139
Nakagawa, Y., 139, 209
Nakamura, T., 219, 591
Nardi, D., 86, 559, 635
Naruse, T., 345
Nazemi, E., 397
Nebel, B., 52
Neiling, J., 639

Nilsson, P., 493
Nishino, J., 465, 473
Noda, I., 229, 379
Noori, A.H., 583

Öberg, M., 413
Obst, O., 199, 469
Ogasawara, T., 219
Ogura, H., 465
Ohashi, T., 595
Ohasi, T., 129
Ohta, M., 351, 477
Ohuchi, A., 449
Okuno, H.G., 139, 209
Oohara, M., 219, 591
Ostrowski, J.P., 639
Oswald, N., 599

Padgham, L., 563
Pagello, E., 297, 429, 559
Palm, G., 291
Pearce, A.R., 285
Peluso, M., 603
Perl, J., 108
Petit, T., 531
Pham, S.B., 64
Philip, D., 531
Piaggio, M., 86, 279, 559
Plöger, P.-G., 579
Polani, D., 108, 481
Polesel, R., 297, 603
Preston, P., 64
Prokopenko, M., 119, 357

Quedec, N., 531

Radjabalipour, B., 397
Rahmani, M., 397
Ramon, J.A., 373, 551
Rebello, M., 531
Reed, N., 493
Reinholdtsen, P., 523
Reis, L.P., 29
Riedmiller, M., 367, 485
Riley, P., 489
Ritter, M., 291
Rojas, R., 239, 547
Rosati, R., 297, 603
Russó, P., 647

Sablatnög, S., 291

- Safari, M., 433
Saffiotti, A., 643
Sakushima, T., 425
Salerno, M., 635
Sammut, C., 64
Sander, G., 405
Sanitati, C., 635
Scalzo, A., 86
Scattolin, N., 603
Scerri, P., 1, 309, 493
Schanz, M., 599
Schappel, B., 497
Schmidt, E., 391, 421
Schmitt, T., 169, 567
Schoell, P., 579
Schröter, K., 405
Schulé, M., 599
Schulz, F., 497
Sgorbissa, A., 86
Shern, R., 575
Shigeoka, Y., 595
Shimora, H., 465
Shinjoh, A., 159
Shinoda, K., 501
Siegberg, A., 579
Sikorski, K., 575
Sim, R., 647
Simon, M., 239
Singh, S., 249
Sinner, A., 367, 485
Sorrenti, D.G., 179
Speranzon, A., 297, 603
Staicu, M., 409
Stankevitch, L., 505
Stolzenburg, F., 199, 469
Stone, P., 1, 249, 333, 489
Stroupe, A., 575
Surtet, G., 531
Sutton, R.S., 249
Suzuki, T., 509
Szerbakowski, B., 52

Tachi, N., 571
Tadokoro, S., 1, 379
Takahashi, H., 321

Takahashi, T., 345, 351, 379, 425
Takahashi, Y., 385, 607
Takeda, H., 591
Takeda, M., 385, 591
Takeuchi, I., 379, 509
Takeuchi, K., 321
Tamura, T., 607
Tanaka, N., 513
Tanaka-Ishii, K., 139, 303
Tayama, K., 321
Taylor, D., 327
Terada, T., 591
Tews, A., 555
Thapper, J., 413
Thate, O., 367, 485
Thiel, M., 52

Uchibe, E., 607
Uthmann, T., 108

van Geest, W., 587
Veloso, M., 489, 623
Visser, U., 391, 421

Wünstel, M., 108
Wada, K., 425
Wasik, Z., 643
Weigel, T., 52
Weland, H.-G., 391, 421
Wendler, J., 259, 405
Werner, M., 651
Wiren, T., 493
Woodvine, B., 563
Wyeth, G.F., 1, 527, 555

Yamamoto, M., 449, 513
Yamamoto, T., 591
Yamasaki, F., 269
Yamauchi, Y., 445
Yanase, M., 607
Yee, Y.S., 543
Yoshida, T., 129, 595
Younesi, H., 433

Zanette, W., 297, 603