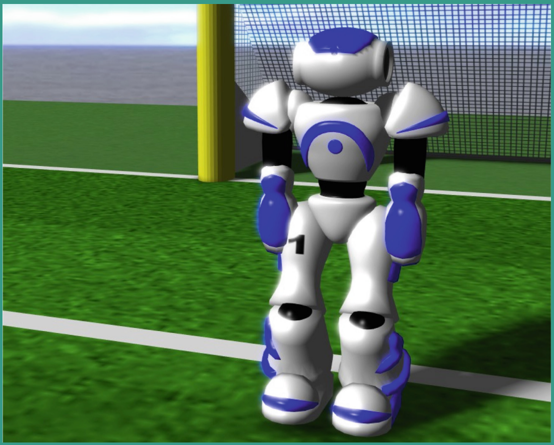
Xiaoping Chen
Peter Stone
Luis Enrique Sucar
Tijn van der Zant (Eds.)

LNAI 7500

RoboCup 2012: Robot Soccer World Cup XVI



Springer

RoboCup

Xiaoping Chen Peter Stone
Luis Enrique Sucar Tijn van der Zant (Eds.)

RoboCup 2012: Robot Soccer World Cup XVI

Springer

Volume Editors

Xiaoping Chen
University of Science and Technology of China, Computer Science School
Hefei 230027, China
E-mail: xpchen@ustc.edu.cn

Peter Stone
The University of Texas at Austin, Department of Computer Science
Austin, TX 78712-1757, USA
E-mail: pstone@cs.utexas.edu

Luis Enrique Sucar
Instituto Nacional de Astrofísica, Óptica y Electrónica
72840 Puebla, Mexico
E-mail: esucar@inaoep.mx

Tijn van der Zant
University of Groningen, Faculty of Mathematics and Natural Sciences
Institute for Artificial Intelligence and Cognitive Engineering
9747 AG Groningen, The Netherlands
E-mail: tijn@ieee.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-39249-8 e-ISBN 978-3-642-39250-4
DOI 10.1007/978-3-642-39250-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013941253

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

LNCS Sublibrary: SL 7 – Artificial Intelligence

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

Printed on acid-free paper

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

Preface

RoboCup 2012, the 16th in the annual series of RoboCup International Competitions, was held during June 18–24, 2012, in the International Exhibition and Convention Center of the World Trade Center in Mexico City, Mexico. This book documents the event, and serves as the proceedings of the 16th edition of the RoboCup Symposium, that is associated with the competition every year.

As one of its features, RoboCup aims to transfer research results and technological advances in robotics and related areas from the laboratory to the real world. The annual RoboCup competitions play a particular role toward this purpose. An autonomous robot is typically a large-scale system that integrates a variety of techniques from different domains, and thus needs the extensive effort of a whole group. Meanwhile, different groups throughout the world have developed different robots based on different ideas, concepts, and approaches. Therefore, it is crucial to have these endeavors tested and compared under the same conditions. RoboCup competitions have provided such tests and comparisons on a world-class scale annually. Each year the conditions are enhanced deliberatively by the community, to keep the tests challenging, fruitful, and approaching closer to real-world conditions.

The RoboCup International Symposium has been an indispensable part of RoboCup, where researchers exchange and discuss their cutting-edge ideas and results related to the initiative, especially those that are stimulated and tested by the competitions. Therefore, the symposium provides a unique forum for exploring and disseminating insights and other results of the worldwide efforts as a part of the RoboCup initiative.

For the 16th RoboCup International Symposium, we received 64 submissions. The submissions were reviewed carefully by the International Program Committee, which consisted of 54 members. Each paper was reviewed by three reviewers. Overall, we accepted 25 of those (39%), of which 12 (19%) were chosen for oral presentation. The symposium also included two invited talks, by Edwin Olson and Martial Hebert. All accepted papers were presented as posters during the symposium. The decision on the best paper award, given to the authors of "Lateral Disturbance Rejection for the Nao Robot," was made by an Award Selection Committee, based on the results of the review process, and the presentation of the three nominated papers at the symposium. The two other nominees were "Robot Localization Using Natural Landmarks," and "Throwing Skill Optimization Through Synchronization and Desynchronization of Degree of Freedom."

This book begins with papers from some of the Champion teams in the 2012 RoboCup competitions. These papers both serve as an insight into the technical challenges emphasized by each different event, and also document some of the key technical innovations that were introduced in 2012. The papers that were nominated for the best paper award come next, followed by the rest of the papers that were accepted for oral presentation at the symposium. Finally, the papers that were accepted for poster presentation round off the remainder of the book.

We would like to take this opportunity to thank the Program Committee members and reviewers for their valuable work, especially their constructive comments and suggestions about the submissions. We also thank all of the authors for their contributions. We are grateful to the Award Selection Committee members for their careful evaluation and decision of the best paper award. We give our special thanks to the local Organizing Committee for their effective and efficient arrangements and support offered the symposium.

Xiaoping Chen
Peter Stone
Luis Enrique Sucar
Tijn van der Zant

Organization

Program Committee

H. Levent Akin	Bogazici University, Turkey
Luis Almeida	Universidade do Porto, Portugal
Sven Behnke	University of Bonn, Germany
Andrea Bonarini	Politecnico di Milano, Italy
Ansgar Bredenfeld	Dr. Bredenfeld UG, Germany
Stefano Carpin	University of California, USA
Stephan Chalup	The University of Newcastle, UK
Xiaoping Chen	University of Science and Technology of China, China
Eric Chown	Bowdoin College, USA
Amy Eguchi	Bloomfield College, USA
Alessandro Farinelli	Verona University, Italy
Bernhard Hengst	UNSW, CAS, NICTA, Australia
Todd Hester	University of Texas at Austin, USA
Dirk Holz	University of Bonn, Germany
Luca Iocchi	Sapienza University of Rome, Italy
Mansour Jamzad	Sharif University of Technology, Iran
Jianmin Ji	University of Science and Technology of China, China
Alexander Kleiner	Linköping University, Sweden
Gerhard Kraetzschmar	Bonn-Rhein-Sieg University, Germany
Michail Lagoudakis	Technical University of Crete, Greece
Darwin Lau	University of Melbourne, Australia
Tim Laue	DFKI Bremen, Germany
Pedro Lima	Institute for Systems and Robotics, Instituto Superior Técnico, Portugal
Jim Little	University of British Columbia, Canada
Norbert Michael Mayer	National Chung Cheng University, Taiwan
Manuele Menegatti	University of Padua, Italy
Cetin Mericli	Carnegie Mellon University, USA
Tekin Mericli	Bogazici University, Turkey
Eduardo Morales	INAOE, Mexico
Tadashi Naruse	Aichi Prefectural University, Japan
Itsuki Noda	National Institute of Advanced Industrial Science and Technology, Japan
Paul G. Plöger	Bonn-Rhein-Sieg University of Applied Science, Germany

A. Fernando Ribeiro	University of Minho, Portugal
Raul Rojas	Freie Universität Berlin, Germany
Javier Ruiz-Del-Solar	Universidad de Chile, Chile
Thomas Röfer	Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Germany
Claude Sammut	University of New South Wales, Australia
Jesus Savage	Universidad Nacional Autonoma de Mexico, Mexico
Matthijs Spaan	Delft University of Technology, The Netherlands
Mohan Sridharan	Texas Tech University, USA
Gerald Steinbauer	Graz University of Technology, Austria
Peter Stone	University of Texas at Austin, USA
Luis Enrique Sucar	Nat. Inst. for Astrophysics, Optics and Electronics, Mexico
Komei Sugiura	National Institute of Information and Communications Technology, Japan
Tomoichi Takahashi	Meijo University, Japan
Yasutake Takahashi	University of Fukui, Japan
Ubbo Visser	University of Miami, USA
Oskar Von Stryk	Technische Universität Darmstadt, Germany
Alfredo Weitzenfeld	University of South Florida Polytechnic, USA
Feng Wu	University of Southampton, UK
Xihong Wu	Peking University, China
Tijn Van Der Zant	University of Groningen, The Netherlands
Mingguo Zhao	Tsinghua University, China
Changjiu Zhou	Singapore Polytechnic, Singapore

Table of Contents

Lateral Disturbance Rejection for the Nao Robot

Juan José Alcaraz-Jiménez[1], Marcell Missura[2], Humberto Martínez-Barberá[1], and Sven Behnke[2]

[1] Information and Communications Engineering, Computer Science, Univ. of Murcia, Spain
{juanjoalcaraz,humberto}@um.es
http://robolab.dif.um.es/

[2] Autonomous Intelligent Systems, Computer Science, Univ. of Bonn, Germany
{missura,behnke}@cs.uni-bonn.de
http://ais.uni-bonn.de

Abstract. Maintaining balance in the presence of disturbances is crucial for bipedal robots. In this paper, we focus on the lateral motion component. In order to attain disturbance rejection and to quickly recover balance, we combine three different control approaches. As a principal building block, we generate center of mass trajectories with a linear model predictive controller that takes scheduled footsteps into account. Strong disturbances generate unexpected angular momenta that can compromise stability. A second control layer extends the underlying preview controller with two recovery strategies that modify the planned CoM trajectories to dampen the rotational velocity of the robot and adapt the timing of the steps according to the expected orbital energy of CoM trajectories at support exchange. Experiments with a real Nao robot show that the system is able to recover from lateral disturbances as long as the robot does not tip over the current support leg.

Keywords: Nao, Disturbances, Angular Momentum, Orbital Energy.

1 Introduction

The Nao bipedal robot enjoys an increasing amount of scientific attention, especially since it has been selected to play humanoid soccer in the RoboCup standard platform league competitions. Several gaits that show reasonable performance on the soccer field have been presented for the Nao. The response to unexpected disturbances, however, remains a weakness of all gaits up to date.

Here, we are presenting a locomotion system based on the model predictive control framework (MPC), whose performance is improved by two additional controllers. The linear inverted pendulum model (LIPM) used by the MPC neglects the angular momentum of the robot, which is not a good assumption if the robot has been disturbed. To overcome this limitation, our first additional controller decreases the tipping moment around the outer border of the sole, ensuring a smooth recovery after disturbances. The second controller adjusts the timing of the steps to account for increased single-support durations while the

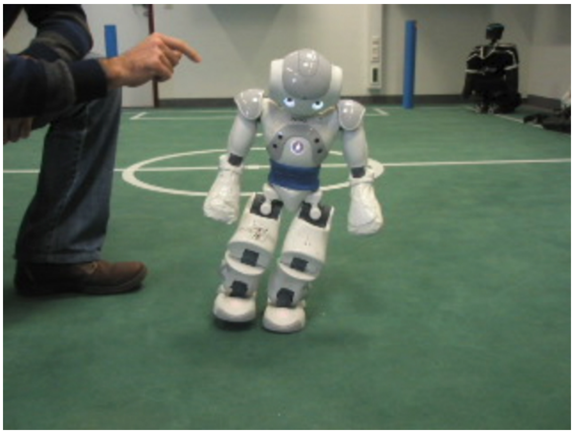


Fig. 1. Nao robot reacting to lateral disturbances

robot is recovering from a lateral push. Both additional controllers are designed for the lateral motion component exclusively. With this configuration, the robot is able to reject relatively strong perturbations from the side—as long as it does not tip over the current support leg.

The importance of lateral stability is often overlooked. While in sagittal direction the swing foot can be flexibly placed virtually anywhere in front of or behind the robot, in lateral direction the location and the timing of footsteps are much more constrained. Most humanoid robots cannot cross their legs, therefore the locations on the outer side of the support leg are not available to maintain stability. Moreover, the rhythmic oscillation induced by the alternating role of support between the left and the right leg dictates a steady timing which is sensitive to disturbances and can quickly lead to a fall, if not adjusted on the fly.

This paper is structured as follows. After reviewing related work in Section 2, we describe the core of our walking engine in Section 3. Section 4 explains the feedback controllers that modify the MPC approach and Section 5 discusses the experimental results obtained.

2 Related Work

Numerous approaches have been proposed to implement dynamic walking for bipedal robots. For example, central-pattern generated omni-directional gaits proved to be an effective approach as they are used by leading teams [1,2] in different leagues of the RoboCup competition.

On the other hand, locomotion systems based on the Linear Inverted Pendulum Model (LIPM) [3] and the Zero Moment Point (ZMP) [4] concepts have become more popular in recent years [5,6,7,8], because they provide a simpler

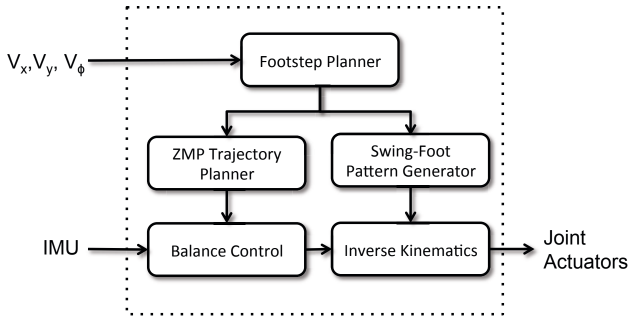


Fig. 2. Architecture of our locomotion module

set of equations to generate Center of Mass (CoM) trajectories and reduce the number of parameters to tune.

In order to exploit the ZMP stability criterion, Kajita *et al.* proposed the use of Preview Control [9] to generate stable trajectories for the CoM. Following this approach, Wieber presented a slightly modified version with an analytical solution under certain constraints [10]. In this work, we utilize this solution for the generation of CoM trajectories that will be subsequently modified by another controller to reduce the angular momentum of the robot.

Kajita *et al.* described in [11] how the CoM trajectories that have been generated with a LIPM-based approach follow potential energy conserving orbits. In this work, we define a target energy level that is used to adapt the timing of the steps. The timing control is a key feature to recover the regular step frequency after strong disturbances. In a similar way, the use of potential energy conserving orbits to regulate the duration of single support stages has also been used in [12], where the focus is also set on the lateral component of the movement. However, in that work only the duration and size of the steps are adapted, but not the CoM trajectories.

3 Walking Pattern Generation

The balance controllers presented in this paper are embedded in the locomotion architecture sketched in Fig. 2. The input received from a higher behavior layer is used for the Footstep Planner to define the timing and position of future footsteps and the trajectory of the swing foot. Further details can be found in [13].

When the robot is walking, its feet swing alternately to reach the new positions of the footstep route. The trajectory that a foot follows in the air is calculated by the Swing-Foot Pattern Generator by means of Bezier curves. The output of this module is a sequence of Cartesian positions and a rotation matrix of the nonsupporting foot in the support-foot frame. These positions are delivered to the inverse kinematics module.

Additionally, to prevent the robot from falling, it is necessary to assure certain stability conditions. In this work, we will employ the LIPM model and the ZMP stability criterion. The LIPM involves two assumptions. First, the robot behaves like a single point mass concentrated at the center of the mass distribution of the body. And second, the motion of the point mass is restricted to a horizontal plane. The dynamic balance condition requires to keep the ZMP within the convex hull of the support polygons. The ZMP trajectories are generated in the ZMP Trajectory Planner module.

Given a certain state of the CoM, it is possible to utilize an optimal control strategy called *Model Predictive Control* to generate the future positions of the CoM that minimizes both, the tracking error of the ZMP trajectories, and the first derivative of the CoM acceleration. In this way, the CoM and ZMP trajectories are first discretized in constant time fragments of duration T, where a constant jerk ($\dddot{x}_k$) is applied to the CoM:

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ \ddot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ \ddot{x}_k \end{bmatrix} + \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix} \dddot{x}_k. \tag{1}$$

Following the approach described in [10], we can find an analytical solution to obtain the value of $\dddot{x}_k$

$$\dddot{x}_k = -e\left(\left(M_u^T M_u + \frac{R}{Q} I_{NxN}\right)^{-1} * M_u^T \left(M_x \hat{x} - P_k^{ref}\right)\right). \tag{2}$$

The matrixes used in Equation (2), are defined in the expressions (3)-(6), where p_k is the reference position of the ZMP at the sample k, N is the number of reference samples and R/Q is a parameter to tune the trade-off between minimum reference tracking error and minimum jerk.

$$e = [1, 0...0], \tag{3}$$

$$M_u = \begin{bmatrix} \frac{T^3}{6} & 0 & 0 \\ \vdots & \ddots & 0 \\ (1+3N+3N^2)\frac{T^3}{6} & \dots & \frac{T^3}{6} - T\frac{z}{g} \end{bmatrix}, \tag{4}$$

$$M_x = \begin{bmatrix} 1 & T & \frac{T^2}{2} - \frac{z}{g} \\ \vdots & \vdots & \vdots \\ 1 & NT & \frac{N^2T^2}{2} - \frac{z}{g} \end{bmatrix}, \tag{5}$$

$$P_k = \begin{bmatrix} p_k \dots p_{k+N-1} \end{bmatrix}. \tag{6}$$

The position of the CoM obtained in (1) is delivered to the inverse kinematics module that, together with the swing-foot pose, will generate the next position for the joint actuators. Although the open-loop execution of the locomotion

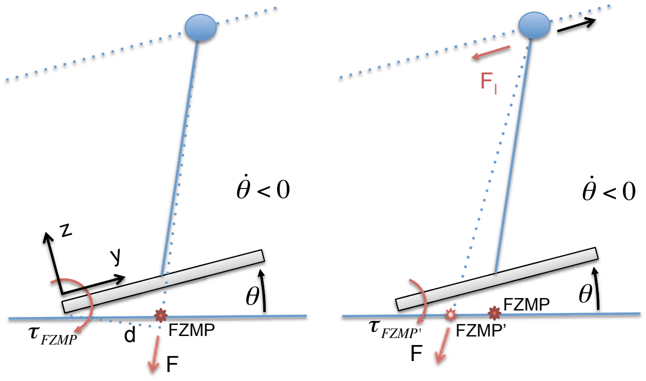


Fig. 3. On the left, the FZMP generates a moment that decreases θ. On the right, the effect of the angular velocity controller is shown. Additional acceleration of the CoM increases the inertial force and pushes the FZMP towards the axis of rotation, which decreases the restoring moment and avoids a too large rotational velocity when θ reaches the horizontal position.

approach described above is acceptable for low speeds, there are important deficiencies in its performance that prevent the robot from attaining a robust gait that rejects disturbances.

Since the system is based on the LIPM, the inertial effects due to rotations of the different parts of the robot, which are important in the case of strong disturbances, are neglected. In the next section, we propose a control approach that copes with this simplification.

4 Balance Control

To improve the performance of the gait pattern generation described previously, we modify the Balance Control module to include controllers that regulate the angular velocity of the CoM and the timing of the next footstep. Since this work focuses on the lateral component of the walking motion, we restrict the equations to the frontal plane.

4.1 Angular Velocity Control

The ZMP specifies the point on the ground where the tipping moment acting on the robot, due to gravitational and inertial forces, equals zero. This point can only exist within the limits of the convex hull of the support polygons. When the ground projection of gravitational and inertial forces lies outside the convex hull, this point is called Fictitious Zero Moment Point (FZMP) [14]. The FZMP involves the presence of a moment that causes a rotational acceleration of the CoM around the closest point of the convex support region.

Given an angle θ between the sole of the support foot and the ground, the total force F resulting from gravity and inertia generates a torque around the contact

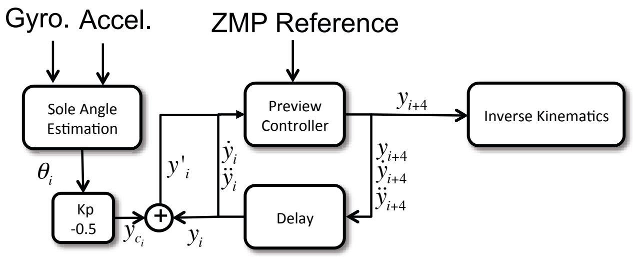


Fig. 4. The linear model predictive controller is complemented by the angular velocity controller

point between sole and ground, as illustrated in Fig. 3. For a robot walking on a flat surface, ZMP-based gaits usually assume $\theta(t) = 0$, and place the target ZMP position approximately at the center of the sole. If a small disturbance occurs, θ will be different from zero and the support polygon reduces to a point at the edge of the sole. In this situation, the target ZMP should be placed theoretically at the edge of the foot to avoid applying any additional torque on the robot, which is the goal of the ZMP stability criterion. Nevertheless, it is common practice to neglect sole angles different from zero and to keep the projection of inertial and gravitational forces approximately at the same point, which is no longer a ZMP but a FZMP. This fact is generally beneficial for balance, because the torque generated by the FZMP will increase the rotational velocity of the sole such that θ decreases, as displayed in Fig. 3 (left).

When the sole reaches the horizontal position again, the robot is rotating with a nonzero angular velocity $\dot{\theta}$, and has therefore an angular momentum that forces the sole to keep rotating beyond the position $\theta = 0$. The rotational velocity of the sole $\dot{\theta}$ is then reduced by the torque that appears at the opposite side of the sole, but high magnitudes of $\dot{\theta}$ in this instant can directly lead to a fall or cause the landing of the swing-foot at an unexpected time and induce further instabilities.

Our strategy to mitigate this problem is to add an offset y_c to the position of the CoM proportional to the estimated angle of the sole

$$y_{c_i} = -K_c\theta_i, \tag{7}$$

where K_c is the positive proportional gain of the controller and i is the discrete time index.

In this way, when the CoM of the robot rotates around the edge of the sole of the supporting foot, the controller will accelerate the CoM to change the position of the FZMP. While the angle between the sole and the ground is growing, the FZMP is shifted away from the rotation axis to increase the torque that decelerates the rotation. On the other hand, when the angle of the sole is decreasing to recover the horizontal position, the FZMP is shifted towards the axis of rotation (Fig. 3 right) to reduce the torque and to reach the horizontal

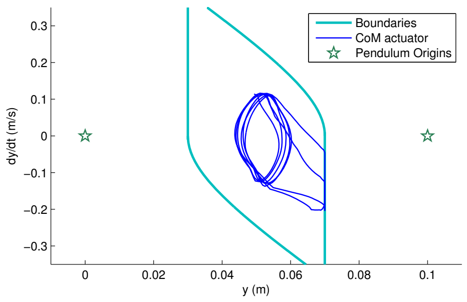


Fig. 5. The states of the CoM delivered by the preview controller are bounded in the phase space. A robot walking on the spot is pushed twice and the trajectory of the CoM reaches the limits.

position with a moderate angular velocity. The integration of the angular velocity controller into the Linear Model Predictive Controller is illustrated in Fig. 4.

Since the lag estimated for the system is four cycles of $10ms$, the actuator commands must be delayed by this amount of time before fusing the control signal of the angular velocity controller and the state of the CoM used by the linear model predictive controller.

In order to obtain an estimate of the sole angle, we estimate the orientation and angular velocity of the torso using the inertial sensors and subtract the delayed torso angle as it was commanded by the actuators four cycles before. The difference between the two angles is equal to the angle of the sole with respect to the floor.

The gyrometers provide accurate angular velocity measurements that can be integrated to obtain an estimate of the torso orientation. This orientation is fused with the angle estimated by the accelerometers to reduce the cumulative error generated by the integration of angular velocity. Since the use of the accelerometers is not sufficient to contain this drift, the FSR sensors in the feet of the robot are used to reset the zero position of the sole angle when a flat contact is detected.

When the robot is pushed from a side, the angular velocity controller will generate a yielding motion of the torso away from the pushing force to avoid the inclination of the sole. This absorption effect must be limited, however, because it can take the CoM to a position beyond the support foot, from where it is not possible to recover. To avoid this situation, the permitted CoM states are bounded in the phase space, as depicted in Fig. 5.

4.2 Step Timing Control

The combination of the linear model predictive controller and the angular velocity controller improves the balance of the robot significantly. Nevertheless, for external disturbances exceeding a certain magnitude, it is necessary to adapt the timing of the step.

Our strategy is to define a target orbital energy E_t in the frame of the next support foot and to calculate the remaining time to reach that orbital energy using the current pendulum origin. To determine the CoM trajectory with respect to the current pendulum origin, we use the 3D-LIPM [3] equations

$$y(t) = y_0 \cosh(kt) + \frac{\dot{y}_0}{k} \sinh(kt), \tag{8}$$

$$\dot{y}(t) = y_0 k \sinh(kt) + \dot{y}_0 \cosh(kt). \tag{9}$$

The position and velocity of the CoM at $t = 0$ are y_0 and $\dot{y}_0$, $k = \sqrt{g/h}$, where g is the gravitational acceleration and h the CoM height. Since we know that the CoM will "rebound" from the current support foot and accelerate towards the next support foot, we set y_0 to the apex of the trajectory and $t = 0$ at the time when the CoM is at y_0. Equations (8) and (9) can then be simplified to

$$y(t) = y_0 \cosh(kt), \tag{10}$$

$$\dot{y}(t) = y_0 k \sinh(kt). \tag{11}$$

The current time and the apex position are calculated as

$$t_n = \frac{\operatorname{atanh}\left(\frac{\dot{y}_n}{y_n k}\right)}{k}, \tag{12}$$

$$y_0 = \frac{y_n}{\cosh(kt_n)}, \tag{13}$$

where y_n and $\dot{y}_n$ are the current estimated position and velocity of the CoM with respect to the center of the current support foot.

Given the length of the step S_y, we can calculate the orbital energy relative to the frame of the next support foot:

$$E_{sw}(t) = \frac{1}{2}\left(\dot{y}^2(t) - \frac{g}{z_c}(y(t) - S_y)^2\right). \tag{14}$$

Our goal is to change the support foot when E_{sw} has the value of the target orbital energy E_t. Substituting (10), (11), and (13) in (14), we can calculate the optimal instant t_s for the the support exchange

$$t_s = \frac{\operatorname{acosh}\left(\frac{2E_t + k^2(S_y^2 + y_0^2)}{2y_0 k^2 S_y}\right)}{k}. \tag{15}$$

The remaining time to the optimal exchange instant will be $\Delta t_s = t_s - t_n$.

We calculate a limit case where the pendulum origin is placed at the limit of the sole to estimate the minimum value for Δt_s. If the current scheduled time to exchange the support $\Delta t_{s_{sch}}$ is less than $\Delta t_{s_{min}}$, we add a delay of just one control cycle to $\Delta t_{s_{sch}}$. This way, the stepping motion is delayed as soon as the instability is detected and we achieve robustness against noisy estimations of the CoM position and velocity, since only large disturbances that are repeatedly detected in multiple control cycles will cause a significant delay of the step timing.

Finally, the support exchange is delayed until the CoM has reached at least 45% of the distance between the current and the next pendulum origin to enforce a symmetrical pose of the robot at support exchange with the CoM half way between the feet.

5 Experimental Results

In this section, we describe the experiments performed to validate our disturbance rejection approach. The platform used is the commercial humanoid robot Nao, developed by the French company Aldebaran Robotics. During the experiments, the robot is placed on a carpet similar to the ones used at RoboCup competitions.

The goal of the first experiment is to validate the performance of our angular velocity controller in combination with the preview control approach. During this experiment, feet motion is disabled so that the robot stands still, but the linear predictive controller is active and the ZMP is held fixed in the middle between the two feet. With this setup, the robot is tilted laterally by 45 degrees, so that it is standing on the outer edge of the sole. Then, the robot is released and allowed to freely swing back to the middle position. Fig. 6 illustrates the performance gained from the angular velocity controller. When the controller is disabled, the angular momentum accumulated by the robot during the time that the torso needs to recover the vertical position compels the robot to keep rotating, and thus the robot oscillates from one side to the other during the next four seconds. On the other hand, the angular velocity controller notably compensates the overshoot of the angular velocity and the equilibrium position is recovered in 1.25 seconds. When the angular velocity controller is disabled,

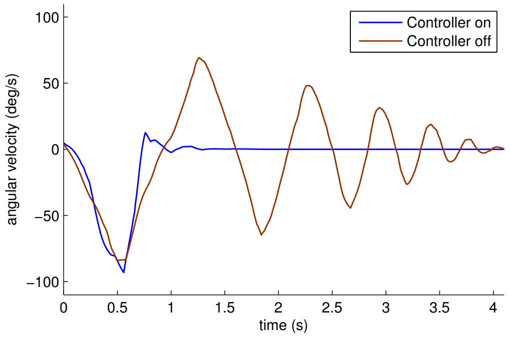


Fig. 6. Overshooting is reduced when the angular velocity controller is enabled

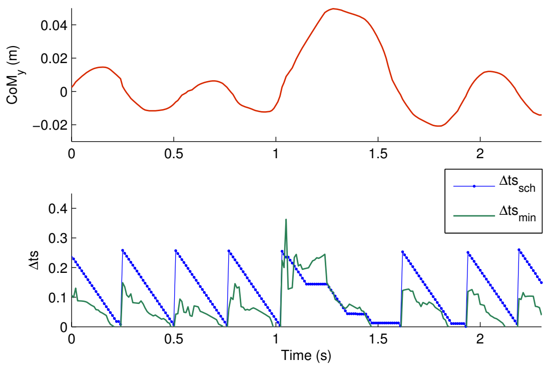


Fig. 7. The robot is pushed at t=1s while walking on the spot and adapts the duration of the step online. The scheduled time to change the support foot $\Delta t_{s_{sch}}$ is delayed during three phases. The first two times the delay happens because the estimated lower bound $\Delta t_{s_{min}}$ exceeds the scheduled time. The third time, $\Delta t_{s_{sch}}$ is delayed until the CoM covers 45% of the distance between the current and the next support foot.

the magnitude of the peak angular velocity in the first rebound is reduced only by 18%. Enabling the controller increases the reduction rate to 88%.

The goal of the second experiment is to demonstrate the performance of the disturbance rejection system while walking on the spot. The value used for the R/Q parameter of the linear model predictive controller is $1e-7$, the CoM height 0.255 m, the default distance between the feet is 0.1 m, and the K_c gain for the momentum controller is set to 0.5. The duration of every step is 0.25 ms and 5% of the time both feet are on the ground. The reference trajectories for the ZMP jump from one foot to the other at the support exchange time and have an offset of 0.01 m towards center of the robot in the lateral dimension and an offset of 0.005 m in forward direction with respect to the center of the foot.

An example for the effectiveness of the step timing control is shown in Fig. 7. Here one can observe that after a strong disturbance, the robot delays the next step in three phases and continues its normal walking rhythm afterwards.

Fig. 8 shows CoM trajectories during the pushing experiment. In the first row, the preview controller is working in open-loop mode. Before the robot is disturbed at t=2.5 s, the estimated position of the CoM follows a rhythmic pattern which is not synchronized with the CoM position sent to the inverse kinematics module. For example, at t=1.75 s, the measured and commanded CoM positions have opposite signs. After the disturbance, the CoM rebounds from the support leg, but the robot tips over on the opposite side.

In the second row, the angular velocity controller is enabled and the measured and commanded CoM trajectories stay synchronized. However, when the robot is pushed, the system is not able to recover the regular pace in time and tries to lift the foot that supports the robot. As a result, the robot needs two seconds to fully recover its balance.

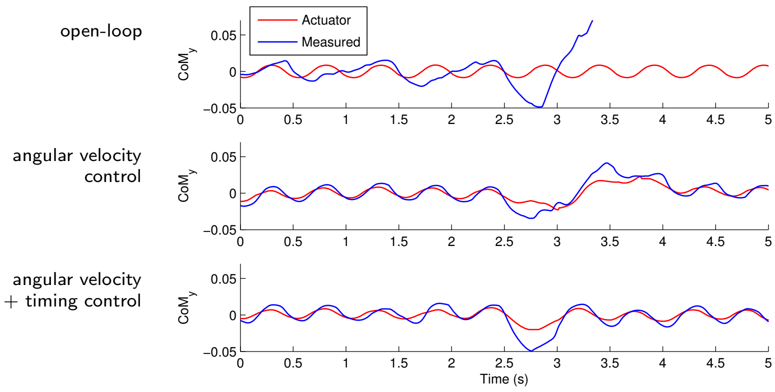


Fig. 8. Performance comparison of three different configurations for disturbance rejection.The robot is pushed at t=2.5 s while walking on the spot. On the top, the preview controller is working in open-loop mode. In the second row, the angular velocity controller is enabled. In the last row, the angular velocity controller and the timing control are both switched on.

This problem is solved when the timing controller is added to the configuration, as shown in the third row. In that case, the regular pace is recovered only 0.5 seconds after the disturbance.

The step timing controller alone (without angular velocity control) has worse performance than the open loop mode in the walking on the spot experiment. The open loop controller ignores the real position of the CoM and frequently supports the robot with the swing foot. In such situations, the LIPM is no longer a suitable model to describe the dynamics of the system because it does not take into account vertical oscillations. On the other hand, when the timing controller is enabled, the robot succeeds in using the scheduled foot to support the robot, but angular momentum cumulates through steps and causes the robot to tip over.

The accompanying video material [15] shows the Nao robot dealing with several disturbances while walking on the spot and recovering from states with high angular velocity.

6 Conclusions

We presented a bipedal locomotion system that combines three different approaches to reject disturbances and to rapidly recover the default posture and gait frequency. The base of the system is a model predictive controller that generates CoM trajectories based on footsteps scheduled for the future. The internal state of the CoM used by this controller is modified to reduce the angular momentum of the robot. Finally, the duration of every step is dynamically adapted to make sure that the orbital energy of the next step is above a minimal threshold.

In future work we will extend the concepts employed to control the lateral component of the walking motion to the sagittal dimension. The main difference in this case is that the velocity of the CoM does not change its sign in every step. On the other hand, the landing position of the feet can be freely modified, since it is not limited by self-collisions.

Acknowledgement. This work is supported by the Spanish Ministry of Education through its FPU program under grant AP2008-01816. Additionally, funding for the project is provided by Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under grant BE 2556/6.

References

1. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 1597–1603 (2006)
2. Graf, C., Härtl, A., Röfer, T., Laue, T.: A Robust Closed-Loop Gait for the Standard Platform League Humanoid. In: Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots (2009)
3. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., Hirukawa, H.: The 3D Linear Inverted Pendulum Mode: A simple modeling for a biped walking pattern generation. In: Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (2001)
4. Vukobratovic, M., Frank, A.A., Juricic, D.: On the Stability of Biped Locomotion. IEEE Transactions on Biomedical Engineering 17(1), 25–36 (1970)
5. Czarnetzki, S., Kerner, S., Urbann, O.: Observer-based dynamic walking control for biped robots. Robotics and Autonomous Systems 57(8), 839–845 (2009)
6. Gouaillier, D., Collette, C., Kilner, C.: Omni-directional closed-loop walk for NAO. In: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids), pp. 448–454 (2010)
7. Xue, F., Chen, X., Liu, J., Nardi, D.: Real Time Biped Walking Gait Pattern Generator for a Real Robot. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 210–221. Springer, Heidelberg (2012)
8. Graf, C., Röfer, T.: A center of mass observing 3D-LIPM gait for the RoboCup Standard Platform League humanoid. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS (LNAI), vol. 7416, pp. 102–113. Springer, Heidelberg (2012)
9. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In: IEEE Int. Conf. on Robotics and Automation (2003)
10. Wieber, P.-B.: Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In: IEEE-RAS Int. Conf. on Humanoid Robots, Humanoids (2006)
11. Kajita, S., Yamaura, T., Kobayashi, A.: Dynamic walking control of a biped robot along a potential energy conserving orbit. In: Robotics and Automation (1992)
12. Missura, M., Behnke, S.: Lateral Capture Steps for Bipedal Walking. In: IEEE-RAS Int. Conf. on Humanoid Robots, Humanoids (2011)
13. Alcaraz-Jiménez, J.J., Herrero-Pérez, D., Martínez-Barberá, H.: A Closed-Loop Dribbling Gait for the Standard Platform League. In: Workshop on Humanoid Soccer Robots of the IEEE-RAS Int. Conf. on Humanoid Robots, Humanoids (2011)
14. Vukobratovic, M., Borovac, B.: Zero-moment point - Thirty five years of its life. International Journal of Humanoid Robotics 1(1), 157–173 (2004)
15. Alcaraz-Jiménez, J.J., Missura, M., Martínez-Barberá, H., Behnke, S.: Nao robot performs disturbance rejection and angular speed reduction tests, `http://www.ais.uni-bonn.de/movies/LateralControlNao.mp4`

HELIOS2012: RoboCup 2012 Soccer Simulation 2D League Champion

Hidehisa Akiyama[1] and Tomoharu Nakashima[2]

[1] Faculty of Engineering, Fukuoka University, Japan
akym@fukuoka-u.ac.jp
[2] Graduate School of Engineering, Osaka Prefecture University, Japan
tomoharu.nakashima@kis.osakafu-u.ac.jp

Abstract. The Soccer Simulation 2D League is one of the oldest competitions among the RoboCup leagues. In the simulation 2D league, the simulator enables two teams of 11 simulated autonomous agents to play a game of soccer with highly realistic rules and game play. This paper introduces the RoboCup 2012 Soccer Simulation 2D League champion team, HELIOS2012, a joint team of Fukuoka University and Osaka Prefecture University.

1 Introduction

The RoboCup Soccer Simulation 2D League is one of the oldest competitions among the RoboCup leagues. It is based on the RoboCup Soccer 2D Simulator [1] that enables two teams of 11 autonomous player agents and an autonomous coach agent to play a game of soccer with highly realistic rules and game play. Due to its stability, the 2D soccer simulator is a very good research and educational tool for multiagent systems, artificial intelligence, and machine learning.

The 2D soccer simulator models only the (x, y) positions of objects. The players and the ball are modeled as circles. In addition to its (x, y) location, each player has a direction that its body is facing, which specifies the direction it can move, and a separate direction in which it is looking, which determines the vision area that the agent covers. Actions are abstract commands such as turning the body or neck by a specified angle, dashing to one of eight directions with a specified power, kicking at a specified angle with a specified power (when the ball is near), or slide tackling in a given direction. A team consists of 11 players including a goalie that has special capabilities of catching the ball when it is near the goalie. The 2D soccer simulator does not model the physical motion of any particular robot, but does capture realistic team level strategic interactions.

In 2012, up to 24 teams were allowed to participate in the 2D competitions. Since the teams competing in the 2D League are already highly competitive, the qualification was done based on a measurement of the quality of the team's scientific work expressed in the submitted team description paper and also the log files with appropriate annotations. Finally, the 2D competitions included 19 teams from 9 countries.

This paper introduces the RoboCup 2012 Soccer Simulation 2D League champion team, HELIOS2012, a joint team of Fukuoka University and Osaka Prefecture University. Especially, we explain the planning framework implemented in team HELIOS2012. There are two characteristic features in HELIOS2012. One is online multiagent planning using tree search, and the other is the decrease in oscillations in decision making. The online multiagent planning using tree search was first implemented in 2010, when HELIOS won the first championship in RoboCup 2010. This framework plays an important role again for more flexible and appropriate action selection in RoboCup 2012. The technique for decreasing oscillations in decision making gives the stability of agent's decision making so that a particular action sequence is likely to be fully completed before the agent changes its mind.

The remainder of this paper is organized as follows. Section 2 introduces the tree search framework implemented in HELIOS2012. Section 3 introduces the modified evaluation function model to decrease oscillations during planning iterations with the tree search framework. Section 4 shows the result of RoboCup 2012 competitions. Section 5 concludes.

2 Online Multiagent Planning Using Tree Search

This section presents the tree search framework for online multiagent planning. This framework is implemented in HELIOS2012. It enables an agent to plan cooperative behavior which involves other agents. For more details of this framework, please refer [2].

2.1 Framework for Searching Action Sequence

In order to simplify the problem, we consider only ball kicking actions in offensive situations. This means that a cooperative behavior can be represented as a sequence of kick actions that are taken by multiple agents. Under this assumption, a cooperative behavior can be generated by tree search algorithms.

The framework generates and evaluates a number of action sequences performed by multiple agents in a continuous state-action space. Generated actions are stored as a node of a search tree. A path from the root node to a leaf node represents an action sequence that defines an offense plan taken by multiple agents. Figure 1 shows an example of an action sequence.

This framework generates action sequences and evaluates their values using the following modules:

– ActionGenerator: This module generates candidate action instances for a node in the search tree. An action instance is generated if it is likely to be performed successfully. The action instance and the predicted state are combined to form an action-state pair instance. The action-state pair instance is added as a new node in the search tree.

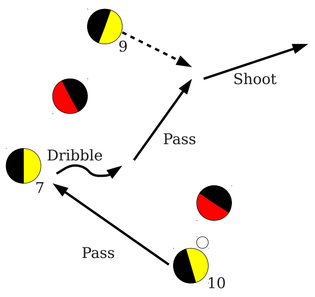


Fig. 1. An example of an action sequence. The chain of four actions is shown: 1) pass from Player 10 to Player 7, 2) dribbling by Player 7, 3) pass from Player 7 to Player 9, and 4) Player 9 shoots to the goal.

– FieldEvaluator: This module evaluates the value of the action-state pair instances that are generated by ActionGenerator. We introduced various state variables and hand-coded rules into the implementd Evaluator instance. The rules evaluate each state variable and the sum of them is returned as the value of action sequence.

In the current implementation, we employed the best first search algorithm [3] as a tree search algorithm. Each node has a value calculated by FieldEvaluator based on the corresponding action-state pair instance.

2.2 Experiments

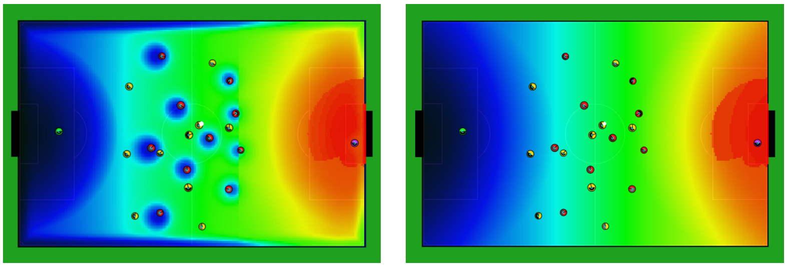
In order to analyze the performance of our framework, we performed computational experiments with several parameter specifications. We used the following parameters:

– Maximum tree depth : { 1(no tree search), 2, 3, 4, 5 }
– Maximum number of traversed node : { 10, 100, 1000, 10000, 100000 }
– ActionGenerator : { Normal, Reduced }
– FieldEvaluator : { Complex, Simple }

Type Normal for ActionGenerator is the same as the one used by HELIOS2011. The number of actions that Type Reduced for ActionGenerator is allowed to generate is about a half of that for Type Normal.

Type Complex for FieldEvaluator is the same as the one used by HELIOS2011, which uses the hand-coded rules with various state variables. Type Simple for FieldEvaluator also uses the hand-coded rules, however they are much simpler than Type Complex. Figure 2 shows an example value mapping on the 2D simulation soccer field.

We used an average goal difference as a team performance indicator. Figure 3 shows the results for each parameter specification. All values are the average of 100 games. In all cases, we can find that the team performance becomes worse



(a) Example value mapping evaluated by the Complex type FieldEvaluator.

(b) Example value mapping evaluated by the Simple type FieldEvaluator.

Fig. 2. Example value mapping evaluated by FieldEvaluator used in the experiments. The red color means the highest value and the black means the lowest value.

when the maximum number of traversed node is ten. This is because agents easily fell into the local minimum. On the other hand, it seems that the team performance is stable if the maximum number of traversed node is more than or equals to 100. This result means that the valuable action sequences can be found within nearly 100 node traversals.

Figure 4 shows the results for each pair of ActionGenerator and FieldEvaluator. The results show that various state variables and rules should be considered in FieldEvaluator. Furthermore, we can find that the number of action patterns generated by ActionGenerator has some impact on the team performance. We could not find the clear reason why the maximum tree depth has no correlation to the team performance. We guess that the oscillation of decision making caused by the poor accuracy of predicted state produced these results.

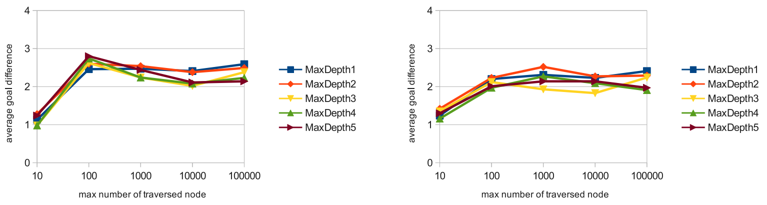
As future works, we have to establish the method to predict future state more accurately and have to establish more effective search algorithm. We are now trying to introduce various game tree search algorithms such as Monte Carlo Tree Search [4].

3 Decreasing Oscillations in Multiagent Planning

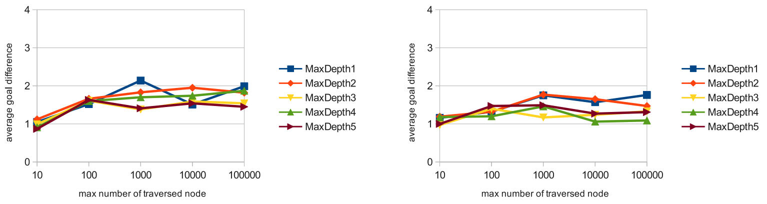
The oscillation of decision making in this paper is defined as follows: When the ball owner agent holds the ball more than one cycle,

- the action type is changed,
- the target player is changed, or
- the error of target position is over the pre-specified threshold.

It is important to decrease the oscillations of decision making in order to stabilize the agent's behavior. In this section, we introduce a modified evaluation function model to decrease the oscillations of decision making.



(a) ActionGenerator: Normal type. FieldEvaluator: Complex type. (b) ActionGenerator: Reduced type. FieldEvaluator: Complex type.



(c) ActionGenerator: Normal type. FieldEvaluator: Simple type. (d) ActionGenerator: Reduced type. FieldEvaluator: Simple type.

Fig. 3. The results of average goal difference for each setting

3.1 Modified Evaluation Function Model

We propose a modified evaluation function model that adjusts the values evaluated by FieldEvaluator. The evaluation value e is modified by the following equation:

$$e' = e \times \exp(-k\frac{||\boldsymbol{p}_{t_n} - \boldsymbol{p}_{t_m}||}{(1 + (t_n - t_m))}), \tag{1}$$

where e' is the modified evaluation value, t_n and t_m are the current time and the time at the previous decision making respectively, p_{t_n} and p_{t_m} are the current target position and the target position at the the same tree depth of the previous decision making, and k is a non-negative real value parameter to change the effect of the time and the distance.

3.2 Experiments

Table 1 shows that the proposed model decreases the oscillations of decision making. It seems that the suitable value of parameter k is between 1.0 and 5.0.

Table 2 and 3 shows the performance evaluation against the opponent teams that participated in the RoboCup2011. We performed 20 games for each team and analyzed the average ball possession ratio and the average goal difference. The result shows the ball possession becomes better for some teams. On the other hand, the goal difference becomes worse for most teams. It is necessary to analyze the games in more detail to evaluate the team performance.

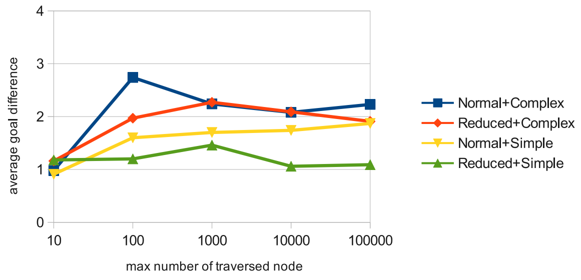


Fig. 4. The average goal difference for each pair of ActionGenerator and FieldEvaluator. The maximum tree depth is fixed to four. Each line corresponds to the pair of ActionGenerator and FieldEvaluator.

Table 1. The number of oscillations and their ratio. The results are the average values of 20 games against agent2d.

k	# of decision making	# of oscillations	ratio
0(No effect)	1926	1389	0.7212
0.1	2677	791	0.2955
0.5	3130	709	0.2265
1.0	3634	594	0.1635
3.0	4047	619	0.1530
5.0	4085	643	0.1574
10.0	4414	966	0.2188
50.0	5264	1012	0.1922
100.0	4676	963	0.2059

Table 2. Ball possession rate for each opponent team

	Without model	With model
agent2d	0.6578	**0.6965**
Edin	0.6429	**0.6840**
Hfut	0.5909	0.6014
Photon	0.5454	**0.6037**
RMAS	0.7720	0.7761
Wright	0.4381	0.4272

Table 3. Average goal difference for each opponent team

	Without model	With model
agent2d	2.5	2.8
Edin	**19.85**	17.15
Hfut	**19.2**	15.8
Photon	18.65	18.05
RMAS	**19.7**	16.7
Wright	4.0	3.0

4 RoboCup 2012 Soccer Simulation 2D League Results

In RoboCup 2012, team HELIOS2012 won the championship by winning all 22 games during the competition, scoring 118 goals and conceding 3 goals.[1] Team WrightEagle from University of Science and Technology of China won the second place, and team MarliK from University of Guilan of Iran won the third place.

5 Conclusion

This paper introduced the champion of RoboCup 2012 Soccer Simulation 2D league. First, we described the tree search approach for multiagent planning implemented in HELIOS2012. Second, we described the modified evaluation funciton model to decrease oscillations in planning iterations. The HELIOS2012 team won 2 championships and 2 runner-ups in the past 4 years of RoboCup competitions. Moreover, team HELIOS have released a part of their source codes in order to help new teams to participate in the competitions and to start the research of multiagent systems[2].

Acknowledgment. The authors would like to thank the additional contributing members of HELIOS2012 (Yosuke Narimoto and Katsuhiro Yamashita)

References

1. Noda, I., Matsubara, H.: Soccer server and researches on multi-agent systems. In: Kitano, H. (ed.) Proceedings of IROS 1996 Workshop on RoboCup, pp. 1–7 (November 1996)
2. Akiyama, H., Nakashima, T., Aramaki, S.: Online cooperative behavior planning using a tree search method in the robocup soccer simulation. In: Proceedings of the 4th International Conference on Intelligent Networking and Collaborative Systems (2012)
3. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall (2009)
4. Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with patterns in monte-carlo go. Technical report, INRIA RR-6062 (2006)

[1] The detailed competition results and all game log files can be found at: `http://www.socsim.robocup.org/files/2D/log/RoboCup2012/`

[2] More information about the HELIOS2012 team can be found at the team's website: `http://sourceforge.jp/projects/rctools/`

RoboCup 2012 Rescue Simulation League Winners

Francesco Amigoni[1], Arnoud Visser[2], and Masatoshi Tsushima[3]

[1] Politecnico di Milano, Milano, Italy
`francesco.amigoni@polimi.it`
[2] University of Amsterdam, Amsterdam, The Netherlands
`a.visser@uva.nl`
[3] Ritsumeikan University, Shiga, Japan
`is0077er@ed.ritsumei.ac.jp`

Abstract. Inside the RoboCup Rescue Simulation League, the mission is to use robots to rescue as many victims as possible after a disaster. The research challenge is to let the robots cooperate as a team. This year in total 15 teams from 8 different countries have been active in the competitions. This paper highlights the approaches of the winners of the virtual robot competition, the infrastructure competition, and the agent competition.

1 Introduction

The RoboCup Rescue Simulation League consists of three competitions:

The **Virtual Robot competition** has the goal to study how a team of robots can work together to get as fast as possible a situation assessment of a devastated area which allows first responders to enter the danger zone well informed. The simulation of the robots is realistic enough to apply the same algorithms to real rescue robots.

The **Infrastructure competition** is a prize to stimulate the innovation factor and the impact of the competition. Progress inside the RoboCup Rescue Simulation League can only be made when each year the challenge gets harder. This can be accomplished by scaling the simulation environment up (larger disaster areas, more agents) or by including more realism into the simulation models. The Infrastructure competition is meant to foster innovation of models and components inside the simulation environment.

The **Agent competition** consists of a simulation platform which resembles a city after an earthquake. In this environment intelligent agents can be spawned, which influence the cause of events in the simulation. The agents have the role of police forces, fire brigades, and ambulance teams.

This paper presents the winner teams of the three competitions within the RoboCup 2012 Rescue Simulation League.

2 Virtual Robot Competition Winner Team PoAReT

PoAReT (*Politecnico di Milano Autonomous Robotic Rescue Team*) won the Virtual Robot competition of the Rescue Simulation League at RoboCup 2012. The PoAReT system is developed by six MSc students in Computer Engineering at the Politecnico di Milano. Full information about the team, including a link to the source code and the list of members with their roles is available at http://home.dei.polimi.it/amigoni/research/PoAReT.html. In the following sections, we overview the PoAReT system architecture and summarize the most interesting scientific results obtained during the competition.

2.1 System Architecture

This section outlines the main features of the PoAReT system, as reported in [1], to which the reader is referred for further information. In developing PoAReT, we push along the *autonomy* axis, attempting to equip the robotic system with methods that enable its autonomous operation for extended periods of time. At the same time, the role of human operator is not neglected, but is empowered by the autonomous features of the system.

Besides the base station, our PoAReT system is composed of mobile platforms (usually the Pioneer All Terrain robot P3AT), each equipped with laser range finders, sonars, and a camera. Laser range finders are used to build a geometrical map of the environment that is represented with two sets of line segments. The first set contains the line segments that represent (the edges of) perceived obstacles. The second set contains the line segments that represent the *frontiers*, namely the boundaries between the known and the unknown portions of the environment.

The main cycle of activities of the PoAReT system is: (a) building a geometrical map of the environment composed of line segments, (b) selecting the most convenient frontiers to reach, and (c) coordinating the allocation of robots to the frontiers. A distinguishing feature of our system is that it can maintain a *semantic map* of the environment that labels areas of the geometrical map with human-like names, like 'room' or 'corridor'. At the same time, the system performs the detection of victims on the basis of the images returned by the onboard cameras and the interaction with the human operator via the user interface.

The architecture of our system is organized in two different types of processes, one related to the base station and one related to the mobile robots, to have a clear separation between their functionalities. Fig. 1 shows the PoAReT system architecture.

The base station embeds the user interface module. The base station process can spawn new robots in the USARSim environment [2]: for each robot, a new independent process is created and started. The processes of the base station and of the robots communicate only through WSS [3] and do not share any memory space, as required by the rules for the competition. A distance vector routing protocol [4] is implemented to deliver messages. Although in principle there is no need to maintain a direct connection between robots and base station (robots explore autonomously and, when connection is active, they can report to the base station and share collected information with other robots), the routing protocol maintains indirect connectivity between robots and base station in order to extend the operative range of the human operator.

The PoAReT User Interface (UI) allows a single human operator to control a relatively large group of robots in an easy way. It displays data to the user and accepts commands from the user to control the spawned robots. It reduces the workload of the operator and increases her/his situation awareness. These two objectives are reached by our UI through a mixed-initiative approach [5]. The PoAReT UI allows a single

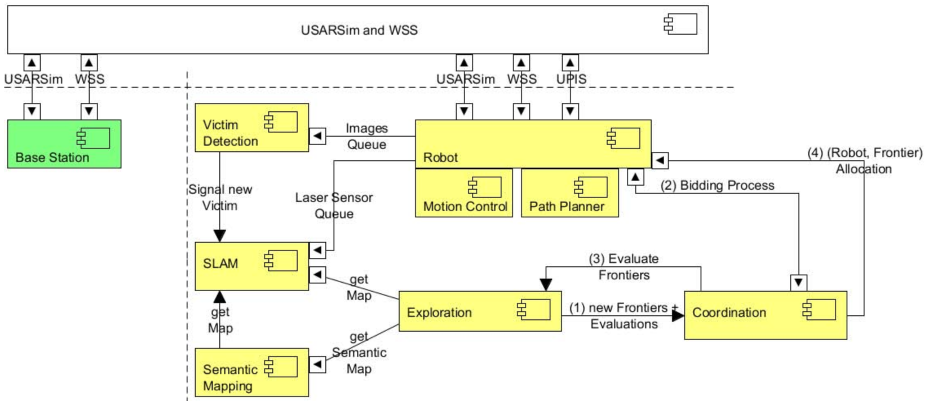


Fig. 1. PoAReT system architecture. The base station module is in green, while the mobile robot modules are in yellow.

operator to control the system by issuing high level commands to robots, like "explore along a direction", by controlling a single robot using waypoints, and by directly teleoperating the robot manually. The UI is also able to filter notifications arriving from the other modules, based on the operator's preferences, past behaviour, and situation parameters.

The robot process is structured in seven different modules, each one related to a high-level functionality: motion control, path planning, SLAM, semantic mapping, exploration, coordination, and victim detection. Almost all of these modules are threads that communicate through a queue system. The main functionalities of the above modules are described in the following.

First, we briefly discuss the motion control module, which is straightforward, given the locomotion model of P3AT, and the path planning module. Path planning is invoked to reach a position with a path that lies entirely in the known space (e.g., the position can be a point on a frontier between known and unknown space). The algorithm we use is a variant of RRT [6].

In our team, the simultaneous localization and mapping (SLAM) problem is tackled by adopting a feature-based method similar to that described in [7]. The SLAM module associates the line segments of a laser scan (points of a scan are approximated with line segments by using the split and merge algorithm [8]) to the the linear features in the map, with respect to distance measures, such as those described in [9, 10]. Then, the module executes an Iterative Closest Line (ICL) algorithm (like [10]) with constraints on the maximum rotation and on the maximum translation to align the scan and the map. All the line segments of a scan are added to the map; periodically a test is carried out to determine whether there is enough evidence to support the hypothesis of two previously associated line segments being in fact the same; if so, they are merged.

The semantic mapping module performs a semantic classification of places and works in parallel with the SLAM module. This module takes as input the line segment map of an indoor environment (updated by the SLAM module) and tries to extract more information than the basic geometrical features, exploiting prior knowledge on the typical structure of buildings. Our approach aims at extending that presented in [11] and [12] to line segment maps. The mapped area is divided into single rooms, identifying the area that belongs to each room and the doorways that divide the rooms. With this information, the space portion marked as *room* is divided into different parts representing every single room separately. Later, each room is classified according to its own characteristic, as a *small room*, a *large room*, or a *corridor*.

The exploration module selects new frontiers to explore, in order to discover the largest possible amount of the environment within the time allowed in the competition. This module evaluates the frontiers by assigning them utilities and, finally, calls the coordination module to find an allocation of robots to the frontiers. We employ an exploration strategy that exploits the geometrical and semantic information gathered by the robots. We take inspiration from [13], where the authors achieve a good exploration performance by distinguishing if the robot is in a hallway or in a room. In our system, we integrate this semantic information into a framework, called Multi-Criteria Decision-Making (MCDM), that is described in [14].

The coordination module is responsible of allocating tasks to the robots. The mechanism we use is market-based and sets up auctions in which tasks (i.e., frontiers to reach) are auctioned to robots [15]. These market-based mechanisms provide a well-known mean to bypass problems like unreliable wireless connections or robot malfunctions.

Finally, the victim detection module is responsible for searching victims inside the competition environment. It works by analysing images coming from the robots' cameras and classifying them according to the presence or absence of victims. In the first case, the victim detection module signals the human operator. We have chosen to implement a skin detector using HSV (Hue, Saturation, Value) color space, followed by a version of the Viola-Jones algorithm [16], a well-known image analysis method already used by many teams in previous editions of the competition.

2.2 Discussion on Competition Results

Besides the good performance that allowed the PoAReT team to win the Virtual Robot competition, some potentially interesting scientific outcomes have been obtained, as discussed in this section.

Firstly, the geometrical maps built by the system and representing the environments of the competition are of good quality, demonstrating the viability of using line segments to represent indoor environments. For example, Fig. 2 shows the geometrical map built by the PoAReT system for the environment of the Day 2 of the competition. Note that, in this run, the maps built by different robots are not merged together, in order to reduce computational burden (this is why some obstacles are represented by multiple aligned line segments). The structure of the environment is represented quite well for understanding by human operator and, importantly, using a limited amount of data (each line segment can be naïvely represented by four numbers). It is also interesting noting that on the Day 3 of the competition, a non-regular indoor environment has been used with several obstacles and with different (vertical) levels. In this case, the segment-based approach has not been much effective to represent the environment.

Second, the availability of a semantic map has been exploited for improving path planning. In particular, the identified *doorways* (i.e., openings between two rooms; an

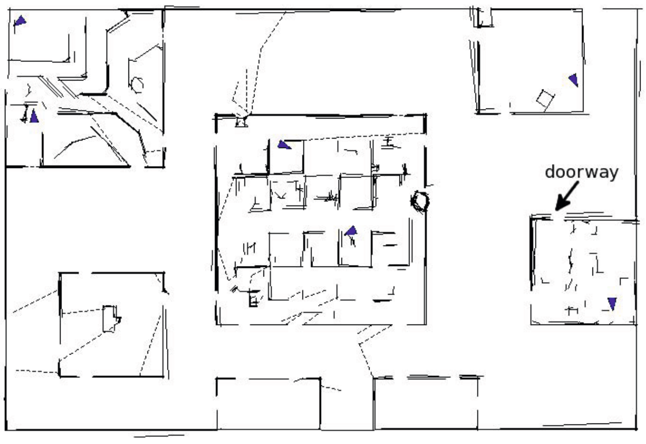


Fig. 2. The geometrical map built by the PoAReT system on Day 2 of the competition. The solid line segments represent obstacles, the dashed line segments represent the frontiers, and the blue triangles represent the positions of the robots.

example is shown on the right of Fig. 2) have been used by the path planner to set waypoints such that a robot crosses a doorway by heading perpendicularly to the line segment representing it. In this way, the robots of our system have been able to cross narrow doors, significantly enhancing their path planning capability.

Third, the high level of autonomy exhibited by the PoAReT system has allowed to explore structured indoor environments very quickly. For example, the results of the first three days of the competition show that PoAReT found 11 victims in 41 minutes, while the next two teams found the same number of victims in 57 and 59 minutes, respectively.

Finally, the autonomy of the PoAReT system does not reduce the role of human operator. In some runs at the competition, Kenaf and AirRobot mobile platforms have been used. Kenafs have been controlled using a controller that is very similar to that of P3AT (without fully exploiting the Kenaf abilities), while AirRobots have been teleoperated manually. The increased workload for the human operator has been compensated by the ability of the system to reach areas (e.g., requiring to climb a stair) that P3ATs cannot reach. In general, teams mainly composed of P3ATs and of some Kenafs and AirRobots showed a good level of adaptability to environments, autonomous behavior, and performance.

3 Infrastructure Competition Winner Team UvA Rescue

The University of Amsterdam is active in the Rescue Simulation League with the UvA Rescue team since 2003 [17, 18]. For several years it had a close cooperation with Oxford University [19]. On several occasions the team contributed to the infrastructure of the competition [20–24]. The system presented at the 2012 Infrastructure competition was developed by a master student in Artificial Intelligence [25].

3.1 The Context

It is well known [26–28] that an aerial robot is a valuable member of a robot team. Several groups indicated the usage of aerial robots in their teams [29–31], as illustrated in Fig. 3, but without a map it is difficult to coordinate the action between the teammembers [28]. Because of the limited payload the aerial robots can carry, it is difficult to equip those robots with range scanners. Without range scanners it is difficult for aerial robots to navigate [32] and to create a map of the environment [33].

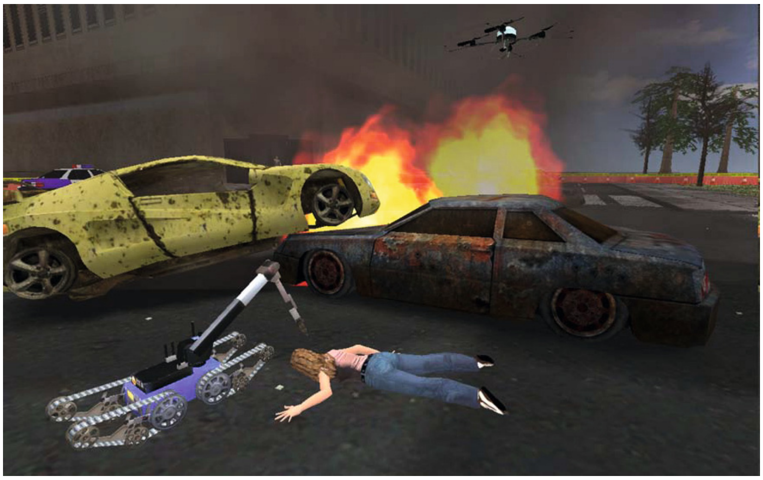


Fig. 3. An early example of the usage of an aerial robot in robot rescue team (courtesy [31]).

Nowadays, small quadrotors with on-board stabilization like the Parrot AR.Drone can be bought off-the-shelf. These quadrotors make it possible to shift the research from basic control of the platform towards applications that make use of their versatile scouting capabilities. Possible applications are surveillance, inspection, and search and rescue. The Parrot AR.Drone is attractive as platform, because it is stabilized both horizontally and vertically. Horizontal movement is reduced based on the images of the bottom camera, while the altitude is maintained based on the signal of a downlooking sonar sensor. Still, the limited sensor suite and the fast movements make it quite a challenge to fully automate the navigation for such platforms. One of the prerequisites for autonomous navigation is the capability to make a map of the environment.

Once such a map exists, a team of micro aerial vehicles could be used to explore an area like a city block. The map is needed to coordinate the actions between the team members. After a disaster one could not rely on prior satellite maps, part of the job of the rescue team is to do a situation assessment and an estimation of damage (roads blocked, buildings on fire, locations of victims visible from the sky).

In the paper presented at the Infrastructure competition [34] a method is described that shows how such a visual map can be built. More details can be found in [25]. To summarize; the visual map consists of a feature map which is built based on storing the most distinguishable SURF features on a grid. This map can be used to estimate the movement from the AR.Drone on visual clues only, as described in previous work [35]. In this paper the focus is on an extension of the previous method, an experimental method [25] to create an elevation map by combining the feature map with ultrasound measurements. This elevation map is combined with textures stored on a canvas and visualized in real time. An elevation map is a valuable asset when the AR.Drone has to explore unstructured terrain, which is typically the case after a disaster (an urban search and rescue scenario).

More details about how the feature map can be used to localize the AR.Drone can be found in [35]. What is really innovative is how the 2D feature map is extended with a method to build an elevation map based on sonar measurements, which was published in the paper for the Infrastructure competition [34]. This paper demonstrates how the elevation mapping method was validated with experiments.

3.2 Elevation Mapping Method

An elevation map can be used to improve navigation capabilities of both aerial and ground robots. For example, ground robots can use elevation information to plan routes that avoid obstacles.

The elevation information is stored in a grid that is similar to the feature map described in [35]. For each ultrasound distance measurement, elevation δ_t is computed and stored in the grid cell that corresponds to the world coordinates where a line perpendicular to the AR.Drone body intersects the world plane. These world coordinates are the position where the center of the ultrasound sensor's cone hits the floor. Because the exact size of an object is unknown, the elevation is written to all grid cells within a radius $\gamma_{elevationRadius}$ around the intersection point. This process is visualized in Figs. 4a and 4b.

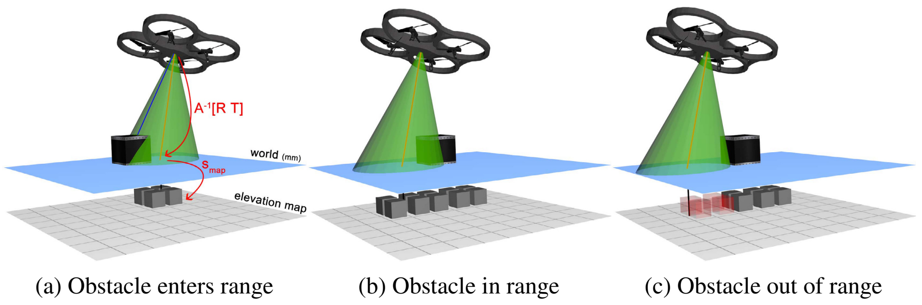


Fig. 4. Overview of the elevation map updates. The green cone indicates the range of the ultrasound sensor. The red line inside the cone represents the center of the cone, perpendicular to the AR.Drone body. In 4a and 4b an elevation is measured. All grid cells within a radius $\gamma_{elevationRadius}$ around the center of the cone (red line) are updated to store the measured elevation. 4c describes the refinement step. When no elevation is measured, all grid cells within the cone (red cubes) are reset to zero elevation.

This approach may lead to cases where the size of an obstacle is overestimated in the elevation map, as can be seen in Fig. 4b. Therefore, an additional refinement step was added to the elevation mapping method. If no elevation is measured ($\delta_t \approx 0$), it can be assumed there is no obstacle inside the cone of the ultrasound sensor. Using this assumption, all grid cells within the cone can be reset to zero elevation and locked to prevent future changes. This refinement step is visualized in Fig. 4c. The radius of the cone is computed using the following equation:

$$r = \tan(\alpha_{ultrasound} \times z_{sensor}) \tag{1}$$

where r is the radius of a cone of height z_{sensor} and $\alpha_{ultrasound}$ is the opening angle of the ultrasound sensor.

3.3 Elevation Mapping Results

The elevation mapping approach has been validated with several experiments. As shown in Fig. 5, the AR.Drone is able to estimate the height and length of two obstacles (a grey and blue box). The brown box is not detected due to its limited height. Therefore, the measured (ultrasound) acceleration is insufficient to trigger an elevation event. As expected, the ramp was not detected. The gradual elevation change does not produce a significant acceleration.

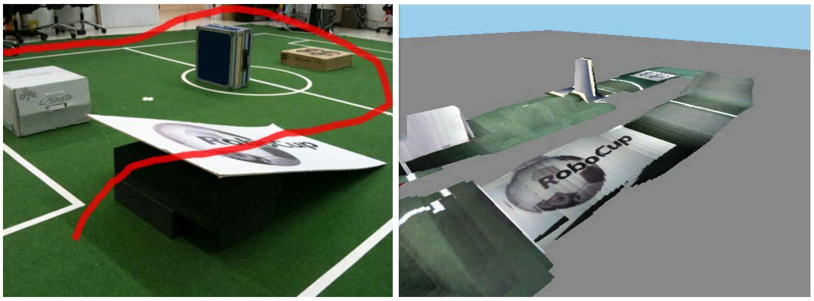


Fig. 5. Elevation map of a flight over several obstacles. On the left the experimental setting (including 4 obstacles) is visible, augmented in red with the path of AR.Drone. On the right the resulting map is displayed, with at the back to elevated areas, representing the white and blue box. For convenience the grid cells of the elevation map are colored with the texture at that point as perceived by the downlooking camera of the AR.Drone.

Elevation changes are detected using a filtered second order derivative (acceleration) of the sonar measurement z_{sensor}, as illustrated in Fig. 6.

Obstacles that enter or leave the range of the ultrasound sensor result in sudden changes in ultrasound distance measurements. These changes are detected when the second order derivative exceeds a certain threshold $\gamma_{elevationEvent}$ and an elevation event is triggered. The threshold $\gamma_{elevationEvent}$ was carefully chosen such that altitude corrections performed by the AR.Drone altitude stabilization are not detected as being elevation events. An elevation event ends when the sign of the second order derivative

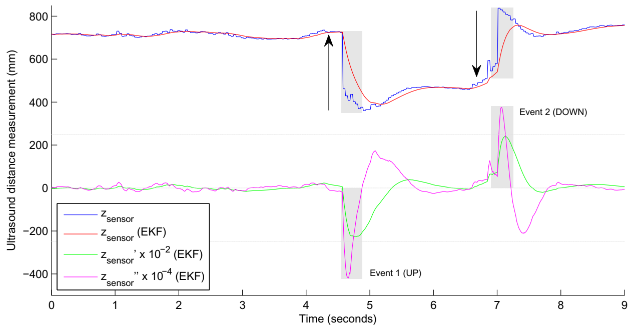


Fig. 6. Response of the ultrasound sensor when flying over an object of approximately $(60, 60, 40)$ mm. The light gray lines indicate the threshold $\gamma_{elevationEvent}$ and null-line. When the second order derivative (magenta line) exceeds the threshold, an event is started (light gray rectangle). An event ends when the derivative swaps sign. Each arrow indicates the change in elevation caused by the event. The first event increases the elevation when entering an object and the second event decreases the elevation when leaving the object. Between both events, the AR.Drone performs an altitude correction, as can be seen by the relatively slow increase of the distance. This increase is not informative about the elevation and is ignored by the elevation mapping method.

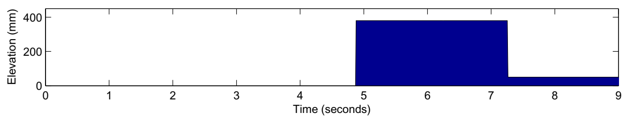


Fig. 7. Elevation δ below the AR.Drone over time. The elevation increases to approximately 40 cm when flying above an obstacle. The elevation is decreased when the obstacle is out of the ultrasound sensor's range. There is a small error between both elevation events, resulting in a small false elevation (± 50 mm) after the AR.Drone flew over the obstacle and is flying above the floor again.

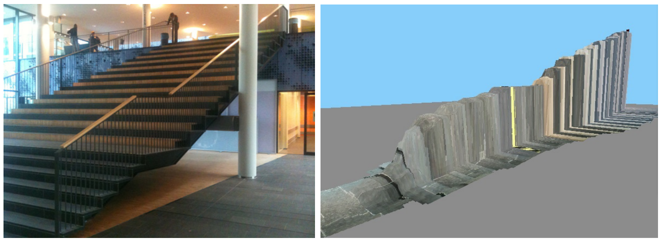


Fig. 8. Photo and map of a large stair at our university which is traversed by the AR.Drone. The depth of each step is 30 cm and the height of each step is 18 cm. The total height of the stair is 480 cm.

switches. This happens when the AR.Drone altitude stabilization starts to change the absolute altitude to compensate for the change in measured altitude. Now, the elevation change can be recovered by subtracting the measured distance at the end of the elevation event from the measured distance before the elevation event was triggered, as illustrated in Fig. 7.

In a final experiment, the AR.Drone flew with a constant speed over a large stair (Fig. 8). The depth of each step is $30\ cm$ and the height of each step is $18\ cm$. The total height of the stair is $480\ cm$. After the stair is fully traversed by the AR.Drone, the estimated elevation is compared against the actual height of the stair.

After fully traversing the stair, the measured elevation is $313\ cm$ and the error is $\frac{480-313}{480} \times 100 = 35\%$. The shape of the measured elevation corresponds with the shape of the stair. However, the approach underestimates the elevation. When traversing the stair, the AR.Drone's altitude stabilization increases the altitude smoothly, which causes a continuous altitude increase. Therefore, the observed altitude difference within an elevation event is smaller than the actual altitude difference caused by an object. Another explanation of the underestimation is the error in the triggering of elevation events (due to thresholds). When an elevation event is triggered at a suboptimal timestamp, the full altitude difference is not observed.

3.4 Summary

The experiments demonstrate what is possible for rapid development when a realistic simulation environment for the AR.Drone is available. The simulation model of the AR.Drone is made publicly available[1] inside the USARSim environment. The validation of this model was described in [35]. We hope that the availability of such a model inside the infrastructure of the RoboCup Rescue Simulation League will contribute in attracting researchers to develop advanced algorithms for such a system.

The mapping method described in [34] is able to map areas visually with sufficient quality for both human and artificial navigation purposes. Both the real and simulated AR.Drone can be used as platform for the mapping algorithm. The visual map created by the simulated AR.Drone contains fewer errors than the map of the real AR.Drone. The difference can be explained by the variance in the lighting conditions encountered by the real AR.Drone. Notice that the USARSim environment is based on a commercial game engine (Unreal Tournement) which already simulates many lighting conditions (e.g., shadows, reflections) quite realistically.

Earlier work [35] shows that the visual map can be used to localize the AR.Drone and significantly reduce the error of the estimated position when places are revisited. Important for a good visual map is that sufficient information is available on the ground; for instance when long straight lines in a gym are followed the travelled distance is underestimated. To conclude; visual mapping is an important capability to scale up the robot team to the level where they can explore a city block, which is close to the current challenge in the Agent competition of the RoboCup Rescue Simulation league.

[1] `http://sourceforge.net/apps/mediawiki/usarsim/index.php?title=Aerial_Robots#AR.Drone`

4 Agent Competition Winner Team Ri-one

This section describes the RoboCup 2012 Rescue Simulation League Agent competition champion team, Ri-one. Our team consists of four students from the Faculty of Information Science and Engineering at Ritsumeikan University. The basic structure of our agent is divided into models and skills.

4.1 Models

The agents have to make informed decisions. The decisions are based on information which is embedded in a World Model. The World Model not only stores a priori and perceived information, but also makes inferences to allow more efficient actions of the agents.

Applying Point of Visibility Navigation Graph. When the agents have the intention to move somewhere in the simulated city, they must send their path as a list of areas (cells) to the server. The agents get the map without any obstacle at the start of simulation, and receive information about nearby open space and obstacles via their sensors at each step. Information given to agents about open space has the form of connected two-dimensional closed shapes, as illustrated in Fig. 9. Obstacles blocking routes have also the closed shapes. With this information, the agents must plan their path in limited time. The path planning is typically performed with a graph search algorithm [36]. The map cannot be converted directly into a graph of connected nodes because the shapes of the cells are not always a convex polygon. The map has to be converted to a new graph which is called a Point of Visibility Navigation Graph when the simulation starts. Therefore, we developed the following method to generate this graph automatically.

In order to generate the graph, we have to consider the relation between nodes and areas defined by the closed shapes. First of all, nodes can be defined as the points (do not need to be centers) within the areas. This is necessary since the path to move along is determined by the list of areas to be visited. However, when connecting the points whose areas are adjacent does not guarantee a collision free path, as shown in red in Fig. 9. Hence, we added additional nodes to the graph to solve this problem. These intermediate nodes are placed on the boundaries of adjacent areas and not explicitly shown in Fig. 9. Therefore, the end result is the undirected bipartite graph which has two kinds of nodes, terminal nodes and non-terminal nodes. Terminal nodes can be used as a start- or end-point of a path, and they must inside an area. On the other hand, non-terminal nodes are intermediate points, on the edges between areas and cannot be a start or an end node to any path.

The algorithm to generate Point of Visibility Navigation Graph has the following pseudo code:

1. Set a terminal node to every area.
2. List all pairs of adjacent areas.

3. For all pairs of areas, define the middle point of the shortest line segment between them (refer to two buildings as A and B, edge of A to B and B to A) as non-terminal node.
4. Relate terminal nodes and non-terminal nodes mutually, according to their visibility.

This method creates new traversable edges according to the visibility of nodes. Fig. 10 shows these lines. With the Point of Visibility Navigation Graph the agents can efficiently perform collision free path planning.

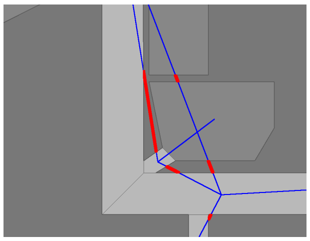


Fig. 9. Connecting areas by connecting the center of the cells. Blue and red line segments represent traversable and non-traversable paths, respectively.

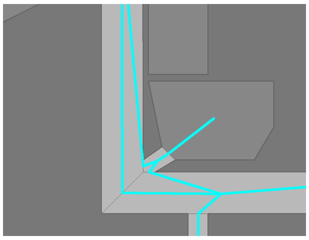


Fig. 10. Generated Point of Visibility Navigation Graph. Cyan line segments represent edges.

Estimating Fires. This estimation makes two assumptions. The first assumption is that an influence of the building's temperature depends on the temperature of another building on fire. The other assumption is that heat spreads in the form of concentric spheres centered on the burning building. When a building whose temperature is t affects another building r meters away from its center, a surface area of sphere with radius r is defined to be S, and a coefficient defined to be k, the influence I satisfies the following relation:

$$\oint_S IdS = kt \tag{2}$$

This I is the influence within the sphere. Then, the angle formed by the lines from the burning building to the intercept of the affected building and affected edge is defined to be θ. An influence I of an infinitesimal surface of the sphere is defined as follows:

$$I = \sin\theta \frac{kt}{4\pi r^2} \tag{3}$$

Fig. 11 shows this idea. This will make it possible to estimate the probability that an invisible building is on fire or not, by calculating the value of I in relation to the temperature of that building.

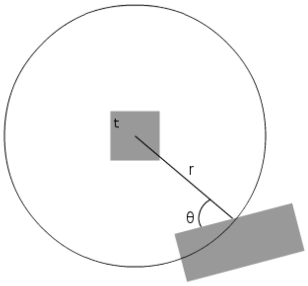


Fig. 11. Influence of temperature t of a building on another building at distance r and angle θ

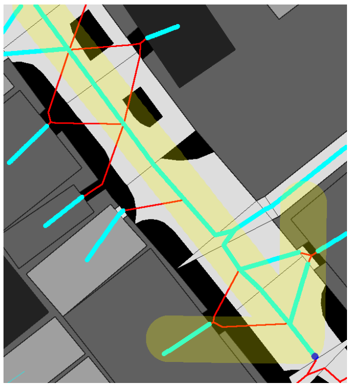


Fig. 12. The result of using Point of Visibility Navigation Graph. The traversable edges are represented by cyan line segments and the non-traversable edges are represented by red line segments.

4.2 Agent Skills

Agents select the best actions from the World Model in order to carry out the operations. The main idea of success of Team Ri-one is the Police Force's skill.

Police Force. Police Forces clear the obstacles caused by the disaster. They must clear the obstacles efficiently to help actions of other agents including Ambulance Teams and Fire Brigades. Therefore, police forces have to choose an obstacle and decide the amount to clear. In order to solve this problem, we use Point of Visibility Navigation Graph from Section 4.1 to decide the obstacle which the police forces will clear. First, a police force computes the shortest path to a target entity without considering obstacles. Secondly, they consider the line segments which compose the shortest path in cleared range. Since each line segment belongs to a single area because of the way the graph has been defined, it is evaluated whether intersections of the line segments and the shapes of all obstacles expanded by a fixed amount exist or not. When an intersection exists on the path, they clear the obstacle. When an intersection does not exist on the path, they move along the path. After that, if an intersection appears on the path, they clear the obstacle likewise when they discover the new obstacle. By repeating this method, they can reach the target entity without clearing obstacles which do not need to be cleared. Fig.12 shows the result of the algorithm application.

4.3 RoboCup 2012 Rescue Simulation League Agent Competition Results

In RoboCup 2012, the Ri-one team won the competition. The success was based on the reduction of unnecessary steps in the planning (for instance the improved Police Force's skill which ignores obstacles which do not have to be cleared) and on the prediction of the dynamics of the simulation (for instance the estimation of the spread of a fire). The improved efficiency was enough to beat our competitors with a narrow margin[2]. The ZJUBase team from Institute of Cyber-Systems and Control, Zhejiang University, China, won second place, and the S.O.S team from Amirkabir University of Technology, Iran, won third place.

5 Conclusion

This paper gives a brief insight in the methods applied by the three winners of the RoboCup Rescue Simulation League. It demonstrates the variety of methods which have to be integrated to create a robot team which is able to cooperate to accomplish the mission to rescue as many victims as possible. The Rescue Simulation League is a competition which keeps on innovating. The DARPA organization has chosen robot rescue as the next challenge and it will be task of the RoboCup community to demonstrate the value of its benchmarks with relation to the scenario of the DARPA Challenge.

Acknowledgement. The PoAReT team gratefully thanks the Fondazione Banca del Monte di Lombardia for the financial support. The UvA research is partly funded by the EuroStars project 'SmartINSIDE' and the Dutch ICT Innovation Platform Cooperation Challenge 'SI4MS'.

References

1. Amigoni, F., Caltieri, A., Cipolleschi, R., Conconi, G., Giusto, M., Luperto, M., Mazuran, M.: PoAReT Team Description Paper. In: RoboCup 2012 CD (2012)
2. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: a robot simulator for research and education. In: Proceedings of the International Conference on Robotics and Automation (ICRA 2007), pp. 1400–1405 (2007)
3. Pfingsthorn, M.: RoboCup rescue virtual robots: Wireless simulation server documentation. Technical Report, Jacobs University (2008)
4. Comer, D.: Internetworking with TCP/IP, vol. 1. Addison-Wesley (2006)
5. Wang, J., Lewis, M.: Human control for cooperating robot teams. In: Proc. HRI, pp. 9–16 (2007)
6. LaValle, A., Kuffner, J.: Rapidly-exploring random trees: Progress and prospects. In: Donald, B., Lynch, K., Rus, D. (eds.) Algorithmic and Computational Robotics: New Directions, pp. 293–308 (2001)
7. Garulli, A., Giannitrapani, A., Rossi, A., Vicino, A.: Simultaneous localization and map building using linear features. In: Proc. ECMR, pp. 44–49 (2005)
8. Nguyen, V., Gächter, S., Martinelli, A., Tomatis, N., Siegwart, R.: A comparison of line extraction algorithms using 2D range data for indoor mobile robotics. Autonomous Robots 23(2), 97–111 (2007)
9. Elseberg, J., Creed, R., Lakaemper, R.: A line segment based system for 2D global mapping. In: Proc. ICRA, pp. 3924–3931 (2010)
10. Li, Q., Griffiths, J.: Iterative closest geometric objects registration. Computers & Mathematics with Applications 40(10-11), 1171–1188 (2000)

[2] The detailed competition results can be found at:
`http://roborescue.sourceforge.net/`

11. Pronobis, A., Martinez Mozos, O., Caputo, B., Jensfelt, P.: Multi-modal semantic place classification. International Journal of Robotics Research 29(2-3), 298–320 (2009)
12. Martinez Mozos, O.: Semantic Labeling of Places with Mobile Robots. Springer (2010)
13. Stachniss, C., Mozos, O.M., Burgard, W.: Speeding up multi-robot exploration by considering semantic place information. In: Proc. ICRA, pp. 1692–1697 (2006)
14. Basilico, N., Amigoni, F.: Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. Autonomous Robots 31(4), 401–417 (2011)
15. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proc. ICRA, pp. 3016–3023 (2002)
16. Viola, P., Jones, J.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
17. Fassaert, M.L., Post, S.B.M., Visser, A.: The common knowledge model of a team of rescue agents. In: Proc. 1st International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster (2003)
18. Post, S.B.M., Fassaert, M.L., Visser, A.: The high-level communication model for multiagent coordination in the robocuprescue simulator. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 503–509. Springer, Heidelberg (2004)
19. de Hoog, J.: Role-Based Multi-Robot Exploration. PhD thesis, University of Oxford (2011)
20. Schmits, T., Visser, A.: An Omnidirectional Camera Simulation for the USARSim World. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 296–307. Springer, Heidelberg (2009)
21. Balaguer, B., Balakirsky, S., Carpin, S., Visser, A.: Evaluating maps produced by urban search and rescue robots: lessons learned from RoboCup. Autonomous Robots 27(4), 449–464 (2009)
22. Terwijn, B., Formsma, O., Dijkshoorn, N., van Noort, S., de Hoog, J., Out, N., Bastiaan, C., Visser, A.: Amsterdam Oxford Joint Rescue Forces: Community Contribution (2010) (published online)
23. Formsma, O., Dijkshoorn, N., van Noort, S., Visser, A.: Realistic Simulation of Laser Range Finder Behavior in a Smoky Environment. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS, vol. 6556, pp. 336–349. Springer, Heidelberg (2010)
24. van Noort, S., Visser, A.: Validation of the dynamics of an humanoid robot in USARSim. In: Proc. PerMIS (2012)
25. Dijkshoorn, N.: Simultaneous localization and mapping with the AR.Drone. Masters thesis, Universiteit van Amsterdam (2012)
26. Davids, A.: Urban search and rescue robots: from tragedy to technology. IEEE Intelligent Systems 17(2), 81–83 (2002)
27. Balakirsky, S., Carpin, S., Kleiner, A., Lewis, M., Visser, A., Wang, J., Ziparo, V.A.: Towards heterogeneous robot teams for disaster mitigation: Results and Performance Metrics from RoboCup Rescue. Journal of Field Robotics 24(11-12), 943–967 (2007)
28. Alnajar, F., Nijhuis, H., Visser, A.: Coordinated action in a Heterogeneous Rescue Team. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 1–10. Springer, Heidelberg (2010)
29. Pfingsthorn, M., Rathnam, R., Stoyanov, T., Nevatia, Y., Ambrus, R., Birk, A.: Jacobs Virtual Robot 2008 Team - Jacobs University Bremen, Germany. In: RoboCup 2008 CD (2008)
30. Calisi, D., Randelli, G., Valero, A., Iocchi, L., Nardi, D.: SPQR Rescue Virtual Robots Team Description Paper. In: RoboCup 2008 CD (2008)
31. Balaguer, B., Carpin, S.: UC Mercenary Team Description Paper: RoboCup 2008 Virtual Robot Rescue Simulation League. In: RoboCup 2008 CD (2008)

32. Visser, A., Nguyen, Q., Terwijn, B., Hueting, M., Jurriaans, R., van der Veen, M., Formsma, O., Dijkshoorn, N., van Noort, S., Sobolewski, R., Flynn, H., Jankowska, M., Rath, S., de Hoog, J.: Amsterdam Oxford Joint Rescue Forces - Team Description Paper - Virtual Robot competition - Rescue Simulation League - RoboCup 2010. In: RoboCup 2010 CD (2010)
33. Nguyen, Q., Visser, A.: A color based rangefinder for an omnidirectional camera. In: Balakirsky, S., Carpin, S., Lewis, M. (eds.) Proc. IROS Workshop on Robots, Games, and Research: Success stories in USARSim, pp. 41–48 (2009)
34. Dijkshoorn, N., Visser, A.: An elevation map from a micro aerial vehicle for urban search and rescue. In: RoboCup 2012 CD (2012)
35. Dijkshoorn, N., Visser, A.: Integrating sensor and motion models to localize an autonomous AR. Drone. International Journal of Micro Air Vehicles 3(4), 183–200 (2011)
36. Buckland, M.: Programming Game AI by Example. Wordware Publishing (2005)

UT Austin Villa 2012: Standard Platform League World Champions

Samuel Barrett, Katie Genter, Yuchen He, Todd Hester, Piyush Khandelwal, Jacob Menashe, and Peter Stone

Department of Computer Science
The University of Texas at Austin
{sbarrett,katie,hychyc07,todd,piyushk,jmenashe,pstone}@cs.utexas.edu
http://www.cs.utexas.edu/~AustinVilla

Abstract. In 2012, UT Austin Villa claimed Standard Platform League championships at both the US Open and RoboCup 2012 in Mexico City. This paper describes the key contributions that led to the team's victories. First, UT Austin Villa's code base was developed on a solid foundation with a flexible architecture that enables easy testing and debugging of code. Next, the vision code was updated this year to take advantage of the dual cameras and better processor of the new V4 Nao robots. To improve localization, a custom localization simulator allowed us to implement and test a full team solution to the challenge of both goals being the same color. The 2012 team made use of Northern Bites' port of B-Human's walk engine, combined with novel kicks from the walk. Finally, new behaviors and strategies take advantage of opportunities for the robot to take time to setup for a long kick, but kick very quickly when opponent robots are nearby. The combination of these contributions led to the team's victories in 2012.

Keywords: RoboCup, Nao, SPL, UT Austin Villa.

1 Introduction

RoboCup, or the Robot Soccer World Cup, is an international research initiative designed to advance the fields of robotics and artificial intelligence, using the game of soccer as a substrate challenge domain. The long-term goal of RoboCup is to build a team of 11 humanoid robot soccer players that can beat the best human soccer team on a real soccer field by the year 2050 [7].

RoboCup is organized into several leagues, including both simulation leagues and leagues that compete with physical robots. This report describes the championship team in the Standard Platform League (SPL)[1]. All teams in the SPL compete with identical robots, making it essentially a software competition. All teams use Aldebaran Nao humanoid robots[2], shown in Figure 1.

UT Austin Villa has competed in the Standard Platform League with the Nao robots every year since the Nao was introduced in 2008. Through these years, we have built a substantial code infrastructure for robot soccer that served

[1] http://www.tzi.de/spl/
[2] http://www.aldebaran.com/

Fig. 1. UT Austin Villa's Aldebaran Nao robots (in pink) competing at RoboCup 2012

as the base for our championship in 2012 [1,2,4,5]. This paper describes the team's key components, including its software architecture, stream-lined vision processing, localization for a field with same-colored goals, quick kicks from the walk engine, and a strategy that adapts to the space each robot has on the field. In addition, a partial release of the UT Austin Villa code can be found at: `http://www.cs.utexas.edu/~AustinVilla/?p=downloads/source_code_and_binaries`

Section 2 describes the software architecture that allows for easy extendability and debugability. Updates to the vision system for the new Nao V4 robots are explained in section 3. We discuss the development of our custom localization simulator and our solution to the challenge of same-color goals in Section 4. Section 5 describes the motion modules used on the robot, including new walk engine kicks. In Section 6, we explain the team's behavior and strategy, which focused on making the right decision based on how close opponent robots were to the ball. Finally, in Section 7, we recount the team's successful performance at RoboCup 2012.

2 Software Architecture

One of the most important aspects of developing RoboCup software is that it needs to be easily debugable and testable. To this end, we have developed an architecture that allows for easy debugging, modification, and testing of individual modules of the code. The design's key element is to enforce that the environment *interface*, the agent's *memory*, and its *logic* are kept distinct (Figure 2). In this case *logic* encompasses the expected vision, localization, behavior, and motion modules. Figure 3 provides a more in-depth view of how data from those modules interact with the system.

The design advantages of this architecture are:

Consistency. The *core system* remains identical irrespective of whether the code is run on the robot, in the simulator or in our debug tool. As a result,

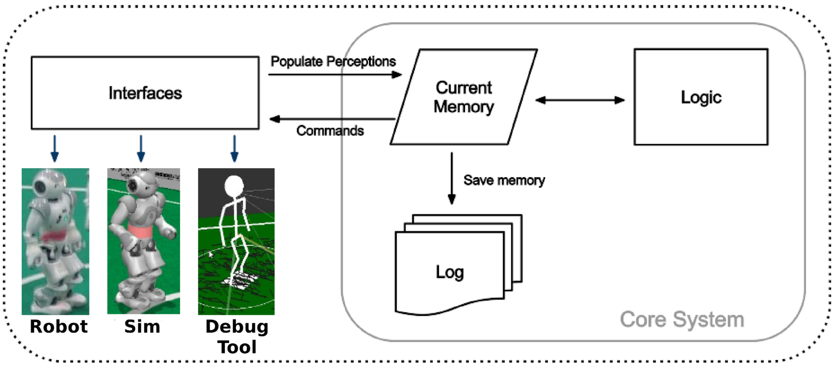


Fig. 2. Overview of the UT Austin Villa software architecture, which separates interface, memory, and logic

we can test and debug code in any of the 3 environments without code discrepancies. The robot, simulator and tools each have their own *interface* class which is responsible for populating *memory*. The interface talks with the system being run to populate perceptions in memory, and then reads from memory to send commands back to the system.

Flexibility. The internal *memory* design is shown in Figure 3. We can easily "plug & play" modules into the system by allowing each module to maintain its own local memory and communicate to other modules using the common memory area. By forcing communication through these defined channels we prevent "spaghetti code" that often couples modules together. For example, a Kalman Filter localization module reads the output of vision from common memory, work in its own local memory and then write object locations back to common memory. The memory module takes care of the saving and loading of the new local memory, so the developer of a new module does not have to be concerned with the low-level saving/loading details associated with debugging the code.

Debugability. At every time step only the contents of current *memory* is required to make the logic decisions. We can therefore save a "snapshot" of the current memory to a log file (or send it over the network) and then examine the log in our debug tool and discover any problems. The debug tool not only has the ability to read and display the logs, it also has the ability to take logs and process them through the logic modules. As a result we can modify code and watch the full impact of that change in our debug tool before testing it on the robot or in the simulator. The log file can contain any subset of the saved modules. For example saving only percepts (i.e. the image and sensor readings) is enough for us to regenerate the rest of the log file by passing through all the logic modules (assuming no changes have been made to the logic code).

It would be remiss not to mention the main disadvantage of this design. We implicitly have to "trust" other modules not to corrupt data stored in memory. There is no hard constraint blocking one module writing data into a location it should not. For example localization could overwrite a part of common memory that should only be written to by vision. We could overcome this drawback by introducing read/write permissions on memory, but this would come with performance overheads that we deem unnecessary.

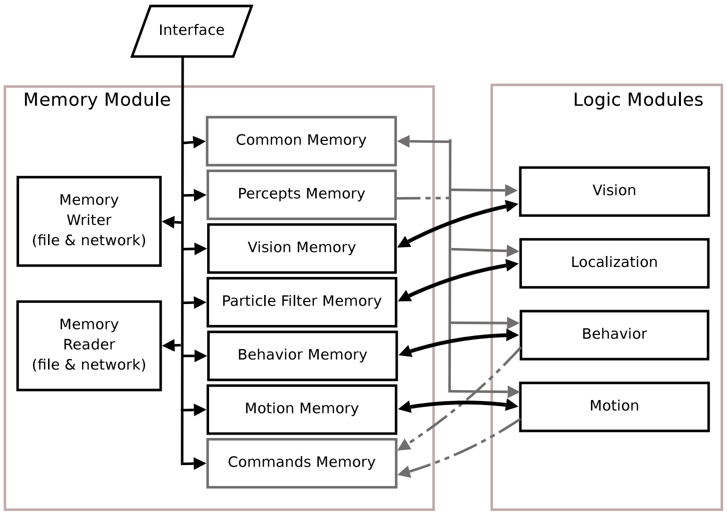


Fig. 3. Design of the Memory module. The gray boxes indicated memory blocks that are accessed by multiple logic modules. Dotted lines show connections that are either read or write only (percepts are read only, commands are write only).

3 Vision

The team's vision system divides the object detection task into 4 stages, each of which is carried out on both cameras. These stages are listed below:

1. **Segmentation** - The raw image is read and segmented using a color table.
2. **Blob Formation** - The segmented image is scanned using horizontal and vertical scan lines, and "blobs" are formed for further processing.
3. **Object Detection** - The blobs are merged into different objects. In this paper, we shall primarily limit our discussion to line and curve detection.
4. **Transformation** - the information given by the pose of the robot is used to generate ground plane transformations of the line segments detected.

These processing steps are outlined in detail in our previous technical report [2]. We therefore focus here on the modifications to allow the use of both cameras, as well as a new set of color table generation and analysis tools.

3.1 Dual Cameras

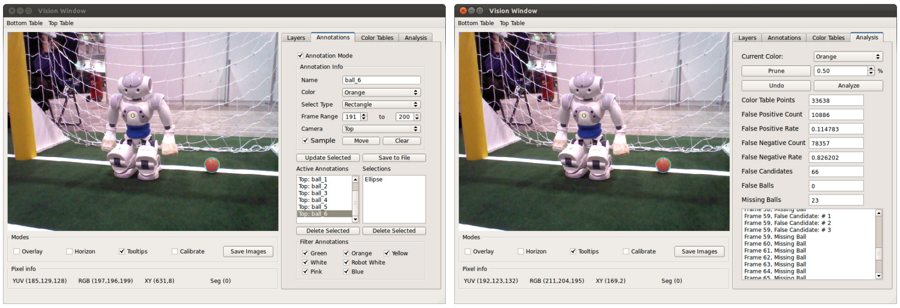
A major hardware upgrade on the Nao V4 was support for streaming image frames from both the top and bottom cameras simultaneously. Combined with improved processing power and faster bus speeds, this made it possible to run the object detection module on images from both cameras while maintaining the hardware-enforced framerate limit of 30 Hz.

It was quickly determined that merging images would come at high computational cost due to small variations in the camera coloring and orientation. Ultimately we chose to modularize the existing detection sequence to run on each camera image. While there is a small amount of overlap between cameras, this is ignored except in the case of the ball. When the ball is detected in both cameras, we use the bottom camera's detection as our selected candidate.

Two key advantages come from this use of both cameras. The first and most obvious is an increased field of view and thus a greater number of detected objects. The second advantage is that at specific head tilt levels the robot's field of view is guaranteed to range from its feet to the edge of the field. We therefore keep the head at a constant downward angle and restrict head movements to horizontal pans. Minimizing head motion in this manner significantly improved our walk stability, and enabled the creation of camera-specific color tables. The bottom camera, for instance, can identify a much wider range of YUV tuples as orange, leveraging the knowledge that this camera rarely has the potential of seeing false positives in its field of view.

3.2 Color Tables and Analysis

One of the significant modifications to our vision codebase came from a set of analysis tools intended to improve the process of creating color tables. These tools include a log annotation system for identifying key regions and objects in image logs; a color table generation tool based on annotations; and an annotation analysis tool. The annotation and analysis tools are shown in Figure 4.



(a)
Overlay [ ] Horizon [ ] Tooltips [✓] Calibrate [ ]
Pixel info: YUV (185,129,128) RGB (197,196,199) XY (631,8) Seg (0)
Annotation Mode [✓]
Annotation Info
Name: ball_6
Color: Orange
Select Type: Rectangle
Frame Range: 191 to 200
Camera: Top
Sample [✓]
Active Annotations: Top: ball_1, Top: ball_2, Top: ball_3, Top: ball_4, Top: ball_5, Top: ball_6
Selections: Ellipse
Filter Annotations: Green [✓] Orange [✓] Yellow [✓] White [✓] Robot White [✓] Pink [✓] Blue [✓]

(b)
Overlay [ ] Horizon [ ] Tooltips [✓] Calibrate [ ]
Pixel info: YUV (192,123,132) RGB (211,204,195) XY (169,2) Seg (0)
Current Color: Orange
Prune: 0.50 %

Color Table Points	33638
False Positive Count	10886
False Positive Rate	0.114783
False Negative Count	78357
False Negative Rate	0.826202
False Candidates	66
False Balls	0
Missing Balls	23

```
Frame 59, False Candidate: # 1
Frame 59, False Candidate: # 2
Frame 59, False Candidate: # 3
Frame 59, Missing Ball
Frame 60, Missing Ball
Frame 61, Missing Ball
Frame 62, Missing Ball
Frame 63, Missing Ball
Frame 64, Missing Ball
```

(a) Annotations (b) Analysis

Fig. 4. Examples of the annotation and analysis tools in use. The blue circle around the ball indicates an elliptical region selection for orange.

The annotation system allows us to graphically select regions of particular images from log data, identify selections by name, and group selection sets over multiple frames to designate the lifetime of a particular object. The tool provides the option of selecting regions as rectangles, ellipses, or polygons. Multiple selections may be combined and labeled by name and color, allowing a user to indicate all key objects in the frame.

These annotations provide the groundwork for color table generation and detection analysis. The generator uses each annotation's assigned label to associate all encompassed YUV values with the indicated color. Color assignment instances are then aggregated to determine false positives (FP), true positives (TP), and false negatives (FN). Using this data, YUV-color mappings are then sorted by their mapping score, $FP/(TP + 1)$. This sorting enables the user to prune off associations with many false positives. When generating color tables for the ball, this pruning provides an effective way of removing mappings from very dark regions under the ball, or from white areas due to reflections on the ball's surface. The analysis portion of the tool additionally displays all FP and FN rates to provide some feedback on the quality of the current color table.

In most cases, no other analysis is provided. When analyzing orange mappings, the tool provides feedback on detected ball candidates and selections as well. The end result is that hundreds of frames of log data can be analyzed in seconds and instantly provide black-box-style feedback on the efficacy of the generated (and pruned) color table.

This technique was initially used with all of the field objects, however there was little to no benefit for colors such as green and white that occur all over the field. The technique was most effective with the ball, where annotations were simple to create and evaluate. Initial setup time and overall accuracy of the resulting color table was comparable with our previous method of creating tables by hand, however with the added bonuses of measurable accuracy and simplified ad-hoc adjustments.

4 Localization

For localization and world modeling, our goals were simple: always know where you are, always know where the ball is, and have a good idea where the opponents and teammates are as well. Like many other teams at the competition [9,6], since 2011 we have used a multi-modal 7-state Unscented Kalman Filter (UKF) for localization [2]. The UKF provides a number of benefits compared to other approaches such as Monte Carlo localization as it is computationally efficient and enables easy sharing and integration of ball information between teammates.

One of the major challenges facing teams in the 2012 RoboCup competition is that the goals were no longer uniquely colored; instead, they were both yellow. This change required major changes in the team's localization system, as now the robots must track which goal is which from their initial positions. First, we removed all "resetting" code that would enable a robot to recover from a kidnapping, as such code could reset the robot to a symmetrical position on the wrong half of the field. Second, we had to improve and update odometry so the robot could successfully track and maintain its position from the beginning of the game using only ambiguous landmarks.

Even with these changes, there were still two problems that arose: bumps and falls. First, the robot's odometry can be incorrect if the robot is bumped or pushed by another robot. Second, if the robot falls down, it is unsure of its orientation upon getting up. In the corners and edges of the field, the robot knows its location and can figure out its orientation from looking around. However, at the middle of the field, two of the orientations are symmetrical. Our solution to this problem at the US Open was to have the robot stay down if it fell near the middle of the field. Rather than risk choosing the wrong direction, it was better to take the 30 second penalty for failing to get up and be reset in a known location. While this solution worked at the US Open when all teams (including us) were still mid-development, we knew that it would not be sufficient for RoboCup.

Clearly, the robot would need to take advantage of shared information from teammates to resolve which direction it was facing. However, testing and debugging localization with full teams of robots is quite challenging. Therefore, we wrote a localization simulator to implement, test, and debug our solutions, shown in Figure 5. The simulator took advantage of the modularity of our code. Instead of interfacing with the code at the perception and motor command layers, it populated the code with simulated output from the vision module, and then used the code's output to the kicking and walking modules to move the robots in the simulation. The simulator proved useful not just for localization, but also for testing various strategies and behaviors, as described in Section 6.

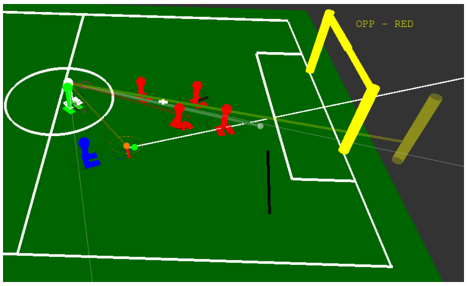


Fig. 5. The localization simulator, showing simulated observations for the green robot in the center circle, along with that robot's estimate of its own location in white

Our final solution to resolving the same-color goal issue was to have robots check whether their teammates thought the ball was in the same location or in the symmetrically opposite location. It is important to listen to more than one teammate, as we do not want one teammate that is going the wrong way to convince the entire team to start going the wrong direction. Thus, each robot keeps a counter which is incremented each time it receives an estimate of the ball within DIST-THRESH of the symmetrical opposite location of its estimate, and decremented when it receives a message with the ball within DIST-THRESH of its own estimate. If this counter reaches a threshold (OPP-BALL-THRESH), the robot spawns a new model with high probability placing it in the symmetrical opposite location of where it was. Through thorough testing in the simulation, we debugged and tuned the values of DIST-THRESH and OPP-BALL-THRESH until this approach worked reliably.

5 Motion

This section describes the walk engine as well as the kick engine that were used by the UT Austin Villa team in the 2012 RoboCup competition. The walk engine was used largely unmodified from an existing implementation, but several important changes were made to the kick engine in 2012.

5.1 Walk Engine

In 2012, the UT Austin Villa team switched to using the walk engine developed by the B-Human team from the University of Bremen [10]. To interface this walk engine with our codebase, we started from the code released by Bowdoin College's Northern Bites team [8]. This code was used with mostly small changes, the largest being the positioning of the arms. Inspired by the 2011 HTWK team, UT Austin Villa chose to keep the robot's arms behind its back in order to reduce side collisions with other robots. This type of collision is especially hard to avoid when approaching the ball, and it is also difficult to detect, often creating differences between sensed odometry and actual movement. Therefore, keeping the arms behind the robot proved to be very helpful.

Another change was in reducing the height of robot's torso while walking. Lowering this height increases the stability of the robot, leading it to stay upright more often when colliding with other robots. However, walking lower puts more strain on the joints, causing the leg joints to overheat more quickly. This problem was especially apparent at the end of the finals, when UT Austin Villa's robots fell over frequently due to this overheating.

5.2 Kick Engine

Our kick engine was composed of two main kick types: kicks executed from a static standing position and kicks executed directly from the walk engine. The static standing kicks were more accurate and could go longer distances, but required the walk engine to halt and the robot to be in a standing position. The walk engine kicks, although limited to shorter distances, could be executed without stopping the walk.

Static Standing Kicks. The static standing kicks were a simplified, quicker version of the kicks we used at RoboCup 2011 [2]. Specifically, we shortened the interpolation time for executing almost every kick state, and we removed the second align state that we used in 2011.

For RoboCup 2012, the only static standing kicks that the robots used were straight kicks ranging between 1.5 meters and 3.5 meters. The kick engine selects the appropriate parameters for the kick based on the desired kick distance and the current ball position with respect to the kicking foot. The robot's joints are controlled using inverse kinematics to reach each desired state in the allocated time. Splines are used to compute the path for the foot to follow as it moves forward during the kick state. The kick engine obtains the desired kick distance by controlling the amount of time needed to move the foot during the kick state. The relationship between the interpolation time and the distance was determined empirically, and varied for each venue based on the field design and carpet texture.

Walking Engine Kicks. Our strategy in the past couple of years has been to act quickly at the ball and keep the ball moving as much as possible. Hence, once we switched to using the walk engine developed by the B-Human team [10], we began experimenting with executing kicks while the walk engine was still running. We first considered the walk engine kicks that B-Human included in their release, but then set out to improve on both their stability and their consistency.

We built upon B-Human's methodology of adding an offset in the x,y,z directions at selected positions in a normal step to turn the step into a kick. For each of the walk engine kicks, we picked four or five times in a normal step and defined the amount that should be added in the x,y,z directions to the swing foot and the stance foot. We then used splines to fit a smooth curve for the amount to be added to the normal foot positions for both the stance foot and the swing foot at each time when commands are sent to the robot during a walk engine kick.

In this manner, we implemented a 1.25 meter straight kick, a 1.0 meter 45 degree kick, and a 0.75 meter side kick that could be executed from the walk engine. None of the walk engine kicks adapt to the position of the ball, but they are only attempted if the robot approaches the ball such that the ball is in an acceptable position for executing the desired walk engine kick.

6 Behavior and Strategy

Our strategy was focused on three goals: 1) move the ball quickly; 2) move it up-field towards the goal; and 3) keep the ball away from opponents. A key component for our strategy was our module for selecting which of the possible kicks to make [3]. This module iterated through an ordered list of kicks, selecting the first *acceptable* kick, one that would move the ball towards the goal while keeping it away from opponents. This procedure gave the robot speed compared to the option of always turning to kick the ball directly at the goal.

One problem with this strategy is that it focused on quickness, even when there were no opponents around and the robot could have taken more time to make a better kick. One focus for this year was to judge how much time the robot had to kick, based on how close the opponents were to the ball. We introduced three thresholds. If the nearest opponent was very far away, the agent decided it had time to rotate around the ball to make a long kick directly at the goal. If the nearest opponent was closer than this threshold but not very close, then the agent would not rotate at all, but could choose from a set of stronger, slower kicks that required the robot to stop walking. If the opponents were within even a smaller radius, then the robot chose from one of its kicks that it could execute from the walk engine, without having to stop walking. Finally, if the opponent was directly at the ball, the robot would choose to make a quick side kick 90 degrees to one side. Its goal was to choose the side free of opponent robots, or the side with a teammate robot if there were no opponents on either side. This decision making process enabled our agents to take advantage of when they had more space to align for better kicks, while still being very quick at moving the ball when they had to be.

The localization simulator enabled us to perform significant testing and debugging of our strategies and kick selections. One of the main changes that came out of these tests was the positioning of players away from the ball. We split the field into two regions. In the back third of the field, one robot stayed to the side of the ball to receive side kicks, and one forward positioned itself a few meters up from the ball. In the rest of the field, one robot stayed to the side to receive side kicks, and one defender stayed back from the ball a few meters. This positioning proved to be very successful at the competition, as the team completed many side kicks to the support player, and the defender was ready to come up and contest the ball if the opponents got by the attacker.

One major issue at RoboCup every year is handling problems with wireless communication, and this was a problem again in 2012. Our robots depend on wireless communication to share ball information, resolve ambiguous orientations that result from the same-colored goals, and decide on which position each robot should play. Once it became clear that wireless would be an issue for the entire competition, we developed a new positioning strategy for this scenario. If the robots detected that wireless was down, one would permanently chase the ball, while the other two went to static positions on the field. However, these two field players would still go to kick the ball if it reached within a set distance of them. This strategy appeared to be better than the approaches other teams took, such as removing all but one robots from the field, or letting all the robots go to the ball all the time.

7 Competition

In 2012, UT Austin Villa won the Standard Platform League at the 16th International RoboCup Competition in Mexico City, Mexico. 25 teams entered the competition and games were played with four robots on each team. The tournament consisted of two round robin rounds, followed by an elimination tournament with the top 8 teams. UT Austin Villa's scores are shown in Table 1.

In the first round robin, UT Austin Villa was in a group with Nao Team Humboldt and MRL. UT Austin Villa defeated both teams 8-0 to win the group. In the second round robin, UT Austin Villa was placed with the Nao Devils,

Table 1. RoboCup 2012 Results

Round	Opponent	Score
Round Robin 1	Nao Team Humboldt	8-0
Round Robin 1	MRL	8-0
Round Robin 2	Nao Devils	5-1
Round Robin 2	Kouretes	8-0
Round Robin 2	NTU RoboPAL	9-0
Quarterfinal	AUTMan	5-0 (forfeit)
Semifinal	rUNSWift	7-6
Final	B-Human	4-2

Kouretes, and NTU RoboPAL. UT Austin Villa defeated the Nao Devils 5-1, Kouretes 8-0, and NTU 9-0 to place first in the group. UT Austin Villa played AUTMan in the quarter-finals, where UT Austin Villa won 5-0 after AUTMan opted for a forfeit partway through the game due to techinical difficulties. In these games, UT Austin Villa experienced wireless issues but was still able to test various strategy parameters to prepare for the semi-finals and finals.

In the semi-finals, UT Austin Villa faced rUNSWift from The University of New South Wales. rUNSWift went up 2-0 and was ahead by 3-1 before the first half ended with rUNSWift claiming a 3-2 advantage. In the second half, rUNSWift extended its lead to 4-2 before UT Austin Villa tied it up at 4-4. rUNSWift and UT Austin Villa then alternated goals until they were once again tied at 6-6. Then, UT Austin Villa called a 5 minute timeout, during which we implemented and uploaded an untested code change that would keep our goalie from diving immediately after a kickoff. We chose to make this change because rUNSWift's long kickoffs towards the left corner of our goal box were causing our goalie to dive on the kickoff and not recover in time to clear the ball. After the timeout, UT Austin Villa claimed its first lead of the game, 7-6, with about 80 seconds left, and held on to win what several observers felt was the most competitive SPL RoboCup game to date.

Less than two hours after winning their semi-final game, UT Austin Villa faced previously undefeated B-Human from the University of Bremen in the championship game. UT Austin Villa took advantage of an empty field after B-Human's robots were called for ball holding penalties to take an early 1-0 lead. UT Austin Villa's lower walk stance height than B-Human gave them better walk stability. Using this advantage and good positioning of its robots, UT Austin Villa claimed a 2-0 lead in the first half, and extended the lead to 4-0 in the second half. However, the lower walk stance led to UT Austin Villa's robots' knees over-heating, reducing their stability and creating problems with getting up. B-Human took advantage of this weakness to score two late goals, but not enough time remained for them to complete a comeback. UT Austin Villa won the championship game 4-2.

8 Conclusion

This paper describes the technical work done by the UT Austin Villa team that led to its 2012 championship in the Standard Platform League. Importantly, all of the 2012 code was developed on a flexible architecture developed previously, that enables easy testing and debugging of code. This year, we updated the vision code to take advantage of the dual cameras and better processor of the new Nao robots. To improve localization, we developed a localization simulator, allowing us to implement and test a full team solution to the challenge of both goals being the same color. Our 2012 team made use of Northern Bites' port of B-Human's walk engine, combined with novel kicks from the walk. Finally, we implemented new behaviors that would take advantage of when the robots had time to setup for a long kick, and kick very quickly when there were other robots around. The combination of all of these contributions led the team to victory at RoboCup 2012 and gives us a good foundation to field competitive teams in future competitions.

Acknowledgements. This year's team builds upon all of the past UT Austin Villa teams. We thank all the previous team members for their important contributions, especially Michael Quinlan and Mohan Sridharan. This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

References

1. Barrett, S., Genter, K., Hausknecht, M., Hester, T., Khandelwal, P., Lee, J., Quinlan, M., Tian, A., Stone, P., Sridharan, M.: Austin Villa 2010 standard platform team report. Technical Report UT-AI-TR-11-01, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory (January 2011)
2. Barrett, S., Genter, K., Hester, T., Khandelwal, P., Quinlan, M., Stone, P., Sridharan, M.: Austin Villa 2011: Sharing is caring: Better awareness through information sharing. Technical Report UT-AI-TR-12-01, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory (January 2012)
3. Barrett, S., Genter, K., Hester, T., Quinlan, M., Stone, P.: Controlled kicking under uncertainty. In: The Fifth Workshop on Humanoid Soccer Robots at Humanoids (December 2010)
4. Hester, T., Quinlan, M., Stone, P.: UT Austin Villa 2008: Standing on Two Legs. Technical Report UT-AI-TR-08-8, The University of Texas at Austin, Department of Computer Sciences, AI Laboratory (November 2008)
5. Hester, T., Quinlan, M., Stone, P., Sridharan, M.: TT-UT Austin Villa 2009: Naos across Texas. Technical Report UT-AI-TR-09-08, The University of Texas at Austin, Department of Computer Science, AI Laboratory (December 2009)
6. Jochmann, G., Kerner, S., Tasse, S., Urbann, O.: Efficient multi-hypotheses unscented kalman filtering for robust localization. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 222–233. Springer, Heidelberg (2012)
7. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The robot world cup initiative. In: Proceedings of The First International Conference on Autonomous Agents. ACM Press (1997)
8. Neamtu, O., Dawson, W., Googins, E., Jacobel, B., Mamantov, L., McAvoy, D., Mende, B., Merritt, N., Ratner, E., Terman, N., Zalinger, J., Morrison, J., Chown, E.: Northern Bites code release (2012), `https://github.com/northern-bites`
9. Quinlan, M.J., Middleton, R.H.: Multiple model kalman filters: A localization technique for roboCup soccer. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 276–287. Springer, Heidelberg (2010)
10. Röfer, T., Laue, T., Müller, J., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., de Haas, T.J., Härtl, A., Humann, A., Honsel, D., Kastner, P., Kastner, T., Könemann, C., Markowsky, B., Riemann, O.J.L., Wenk, F.: B-human team report and code release (2011), `http://www.b-human.de/downloads/bhuman11_coderelease.pdf`

TUMsBendingUnits from TU Munich: RoboCup 2012 Logistics League Champion

Sören Jentzsch, Sebastian Riedel,
Sebastian Denz, and Sebastian Brunner

Robotics and Embedded Systems, Department of Informatics,
Technische Universität München, Munich, Germany
{jentzsch,riedels,denz,brunnese}@in.tum.de
http://www6.in.tum.de/

Abstract. The new RoboCup Logistics League sponsored by Festo offers a competition within a simulated industrial environment. In order to solve the logistical tasks, all three Robotinos not only have to operate autonomously in a flexible, effective and robust way on their own, they should also collaborate efficiently in order to maximize the overall outcome. In this paper, the first world champion of the Logistics League, TUMsBendingUnits from the Technical University of Munich (TUM), presents their logistical system with focus on approaches concerning robot hardware modifications, software architecture, task planning and execution, multi-robot collaboration, visual perception, motion planning and execution.

1 Introduction

The RoboCup Logistics League sponsored by Festo is a new industrially inspired competition. After two years of demonstration the Logistics League has been added to the approved RoboCup portfolio in 2012 [6], offering a competition with focus on achieving a flexible and efficient material and information flow inside a factory area.

A team consists of three robots based on the robot platform Robotino from Festo [3] which have to operate autonomously without any kind of external computing power, control station or human interference [5]. Each team competes for the highest score in 15 minutes on its own separate competition area with a size of 5.6m $\times$ 5.6m, as shown in Figure 1. The main task is to continuously execute a 3-staged production cycle and to transport the final product to the currently active delivery gate. Whereas the machine positions are known in advance, their types are not and have to be explored by the robots. Each product is represented by a data carrying RFID tag mounted on a red hockey puck and each machine consists of an RFID read/write device and a signal unit in order to show the robot the actual status of this very machine. Alongside this main task more logistical challenges have to be solved, for example dealing with out-of-order machines, handling express goods within a certain timeframe, reacting

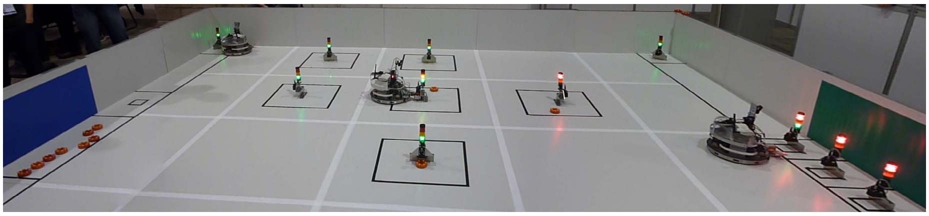


Fig. 1. Factory area of the Logistics League including the raw material zone (left), the production machines (middle), the delivery gates (right) and three autonomously operating Robotinos of the team TUMsBendingUnits

to changing delivery gates and recycling consumed goods in order to obtain new raw materials.

In this paper, the winning team TUMsBendingUnits of the RoboCup Logistics League 2012 championship in Mexico City presents their hard- and software approaches which did master these logistical tasks.

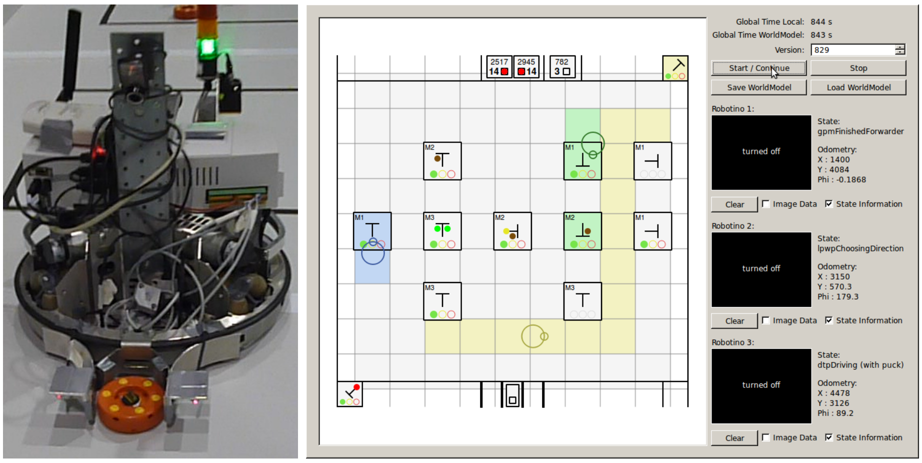
2 TUMsBendingUnits: System Overview

Following the championship of the demonstrational Festo Logistics Competition at RoboCup 2011 in Istanbul, team TUMsBendingUnits of the Technical University of Munich (TUM) could successfully defend their title at the RoboCup Logistics League 2012. After convincingly winning all games at the semi-finals group stage, TUMsBendingUnits won the finals against team Leuphana with a new score record of 160:49.

This section describes the hard- and software of TUMsBendingUnits, which includes the robot sensor equipment, software architecture, task planning and execution, logistical collaboration between the robots, visual perception, as well as motion planning and execution.

2.1 Robot Hardware

The Logistics League is based on the robot platform Robotino, designed and manufactured by Festo [3]. Robotino is actuated by a three-wheeled holonomic drive system. It provides an embedded PC/104 (AMD LX800 processor with 500 MHz) combined with an I/O control board and a LPC2377 32-bit microcontroller to access actuators and sensors. Alongside basic equipment like a WLAN module and the passive puck-pushing device, the teams are free to change or add any kind of sensors. For our Robotinos, we use the default front IR sensor for monitoring the puck possession, two pre-calibrated binary optical sensors to detect and align with black lines on the ground, and two more optical sensors mounted along the customized pushing device in order to align with production machines. A Logitech Webcam C905 is used for visual perception and wheel odometry is assisted by a CruizCore XG1000 gyroscope. Figure 2 (left) shows our final version of the Robotinos.


Global Time Local: 844 s
Global Time WorldModel: 843 s
Version: 829

Robotino 1:
State: gpmFinishedForwarder
Odometry:
X : 1400
Y : 4084
Phi : -0.1868
[ ] Image Data [✓ State Information]

Robotino 2:
State: lpwpChoosingDirection
Odometry:
X : 3150
Y : 570.3
Phi : 179.3
[ ] Image Data [✓ State Information]

Robotino 3:
State: dtpDriving (with puck)
Odometry:
X : 4478
Y : 3126
Phi : 89.2
[ ] Image Data [✓ State Information]

Fig. 2. Robotino of TUMsBendingUnits (left) and their graphical visualization, debugging and controlling tool (right) for the competition

2.2 Software Architecture

The overall software architecture of the Robotino consists of several layers which are shown in Figure 3. In order to control the wheels and to manage analog and digital inputs and outputs, the realtime-based control daemon of the PC/104's Ubuntu uses serial communication to access the I/O board. On top of this architecture we have our application (TBU_ACAPS), programmed in C++ using several external libraries and an internal one (TBU_BASE). Furthermore, we established direct access to the control daemon's shared memory segment, thus effectively bypassing the Robotino-API ComServer and boosting the sensor- and actuator-based communication frequency significantly.

Now let us take a closer look at the software architecture of our application, as visualized in Figure 4.

Starting with the sensorial part, the sensor server has direct access to the shared memory with all the raw sensor and odometry values while retrieving visual perception data from the camera module depending on the currently desired detection mode. Except for the sensor server, each module consists of or operates within own threads, thus resulting in a highly multithreaded architecture. The sensor event generator module constantly polls the sensor values through the server and converts the raw values to discrete events like EvCameraGreenLightDetected or EvHasPuck, depending on certain changes in the data, triggers or thresholds.

The heart of our software architecture is the state machine, which asynchronously processes events from the event queue and executes basic, generalized routines, like grabbing a puck, driving to a certain machine or delivering a puck. These behaviors are realized as own sub-state machines and invoked by the job handler as the robot proceeds in the execution of its current job. Once a job is completed, the job planner decides for a new job, taking the current state of the plant as well as the robot's teammates into account. This information is

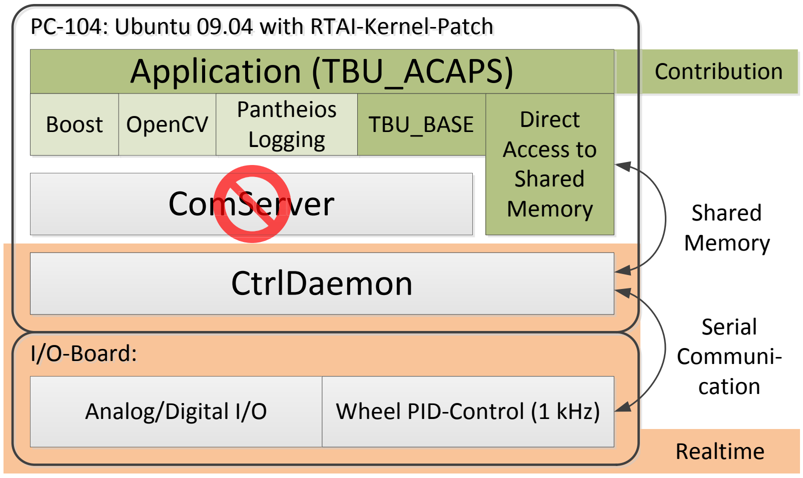


Fig. 3. Overall Robotino software layers with our software code on top of the realtime components of the standardized Robotino platform

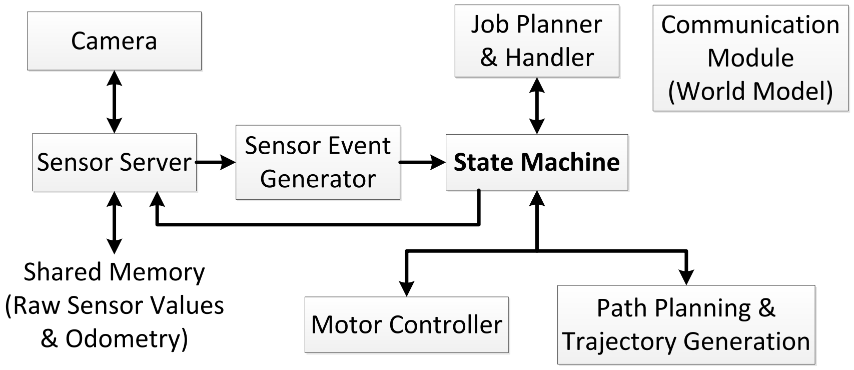

Camera
Job Planner & Handler
Communication Module (World Model)
Sensor Server
Sensor Event Generator
State Machine
Shared Memory (Raw Sensor Values & Odometry)
Motor Controller
Path Planning & Trajectory Generation

Fig. 4. Abstract block diagram of the software architecture, where each block represents a separate software module with the arrows indicating the main information flow and function calls, respectively

subsumed in a world model structure, which is shared and synchronized between the robots via the communication module.

The state machine itself has access to other modules, for example in order to control the camera via the sensor server, ask for the next job, execute a motion in the motion controller module or request the path planning module to plan a certain path. With completion of a task, these modules again utilize event-based communicatation with the state machine.

Finally, a graphical user interface running on an external computer was developed for debugging and basic interaction purposes, like starting or pausing the robots (Figure 2, right). It visualizes the current world model and additional information about each robot, like the current state-machine state, the camera image and odometry values.

2.3 Task Planning and Execution

The Logistics League does not allow teams to utilize an external central controlling system and thus each robot has to be able to plan its next task on its own. This chapter introduces our solution to decentralized planning and execution in a multi-robot team.

Job Planning. The job planning module of each robot takes the current state of the world model and the last executed job as input for determining the next job. We distinguish two different types of jobs: the DeliverPuckJob and the CheckInsertionAreaJob.

The DeliverPuckJob is the basic job type that covers all of the actions which are required to solve the basic production cycle. It always takes a start node, such as the input zone or a machine, a certain puck at that node, and a destination to which the good has to be delivered to.

In order to meet the express good challenge within the official time span of 120 seconds, the CheckInsertionAreaJob periodically checks for an express good within the insertion area. Note that, as all of our robots run the same software code, we avoid delegating a robot only for the express good challenge, thus focusing on the main production cycle.

The job planner creates a list of all possible jobs and then chooses the next task based on the highest priority value for each job. These priority values are assigned for each job as follows:

$$\begin{aligned} priority(dJob) &= available(dJob) \cdot (\, basicPriority(dJob) \\ &\qquad - distPosStart(dJob) - distStartTarget(dJob)\,) \qquad (1) \\ priority(cJob) &= available(cJob) \cdot (timeElapsed() - distPosIA()) \qquad (2) \end{aligned}$$

Equation 1 calculates the priority for the DeliverPuckJob (*dJob*) as follows: At first, the function *available* checks for availability of the job and returns 1 only if the following conditions are met (otherwise 0): the picked up good is available at the starting machine, the target machine has the appropriate machine type, requires this good, has the status of being ready, has no blocking good underneath and both the starting machine and the target machine are not occupied by another robot. Then, we assign a certain basic priority to each job type (*basicPriority*). For example, the basic priority for delivering the final product to the delivery gate is much higher than exploring an unknown machine with a raw material. These basic priorities were tuned manually in order to prioritize certain job types and to boost the overall production of a final product effectively. However, we also consider the distance which the robot has to drive from its current position to the starting machine (*distPosStart*) and from the starting machine to the target machine (*distStartTarget*), including a penalty if the robot's last job did not end on this very starting machine. Thus, in the end, also jobs with a low basic priority but sufficiently short travel distances can be prioritized over jobs with a higher basic priority.

Equation 2 outlines how the priority for the CheckInsertionAreaJob *cJob* is calculated. First of all, the job is only available if at least 25 seconds elapsed since the last check, we already identified at least one suitable target machine, the maximum number of express goods were not reached yet and the express good insertion area is not occupied by another robot (*available*). Then the priority value depends linearly on the time elapsed since the last check (*timeElapsed*) minus a penalty depending on the distance between the robot's current position and the insertion area of the express good (*distPosIA*).

In the end, the job planning module selects the job with the highest priority value and passes it to the job handling module. If there is no job currently available, the robot moves to a random neighboring grid node unblocking the current node in order to avoid deadlock situations, and starts over again after a certain amount of time.

Job Handling. The job handler takes over the selected job and sequentially triggers the execution of appropriate sub-state machines. At first the robot needs to get to the starting machine. This may or may not include leaving the current machine (the current machine could also be the starting machine). If so, the job handler calls the corresponding sub-state machine and passes over all relevant job information, for example the exact starting and target machine. This allows the sub-state machine to leave the current machine in the most suitable and efficient direction. In similar manner sub-state machines are triggered for: actual driving to the starting machine, picking up a puck, leaving the starting machine, driving to the target machine and placing the puck underneath the RFID-device. After completion of each sub-state machine, the world model is updated accordingly and synchronized to the other robots. In case of the target machine showing red light and thus signaling the out-of-order state, the job handler triggers the job planning module in order to find a new target for the carried good. Finally, after successfully executing the job, the next one is planned.

Logistical Collaboration. The logistical collaboration basically consists of breaking the core production cycle of producing and delivering the final product into atomic operations. These are exactly the DeliverPuckJobs, where we deliver a puck from one machine to another, as there is no efficient way to hand over a good from one robot to another. By operating on a synchronized world model, the robots then automatically collaborate and avoid collisions.

Updating the world model has to be reliable. Every time a robot changes the state of world model objects, such as grid nodes and machines and all their properties, it synchronizes changes with the other robots. To prevent simultaneous write operations, every robot is only allowed to make changes when it holds active TCP connections to all robots (including itself). As every robot only allows for one incoming TCP connection and connections are established in a fixed order, this effectively prevents conflicting write operations. A world model version number further prevents from overriding a newer with an older world model in case of rebooting a robot or network failures.

2.4 Visual Perception

In order to safely and precisely grab a puck or navigate to a certain key location (e.g. to a production machine or a delivery gate), the robot's motion must be based on sensor data, instead of relying only on the internal odometry values. We utilize visual perception with our color camera to detect multiple pucks (Figure 5), to perceive the light states of the production machines, and to locate the currently active delivery gate from distance (Figure 6). For each task, we dynamically switch certain camera parameters (e.g. brightness, contrast, saturation, white balance temperature, gain, exposure, no backlight compensation) in order to maximize the visual perception of task-specific key differences in the image. For example, for the light state detection, we heavily increased the exposure value and set the contrast to zero, which results in images shown in Figure 6, where only the lights turned on are perceived with their corresponding color values, as opposed to lights visualized in Figure 5 (right), where basically each light turned on consists of a white core.

Given these color images and the fact that memory and computation power of our robot is very limited, we apply a sliding window technique on color-thresholded HSV images in order to detect pucks and lights. Due to the perspective distortion of the camera, our window size for detecting pucks depends on the current height in the image. To efficiently calculate the matching pixel sum for each window (up to 19,200 windows for the whole 160x120 image), we construct the integral image (or summed area table) for our binary image in advance, as detailed in [7]. For the light detection this entire procedure is done separately for every light color.

In case of detecting and navigating to pucks and the currently active delivery gate, we have to transform pixel coordinates (center points of the matching windows) to metric units in the robot's local coordinate frame. For this task we collected a sufficiently large amount of training data and used the machine learning tool Eureqa [2] to identify an appropriate conversion function.

In order to ensure a robust light state detection including the occurrence of yellow flashing light, we utilize a light state buffer as follows: We process our images as usual, but only after at least a certain amount of time elapsed and a certain amount of images were processed, we make our final decision based on the recognized light states so far. Then, we clear the buffer and start over again.

2.5 Motion Planning and Execution

The regularly oriented and positioned machines allow for significant reduction of the motion planning problem. Paths between points of interest (machines,

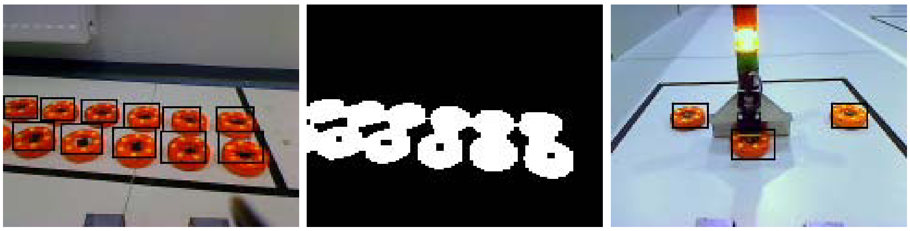


Fig. 5. Puck detection at the raw material zone (left) with the corresponding binary image (middle) and near the production machines (right)

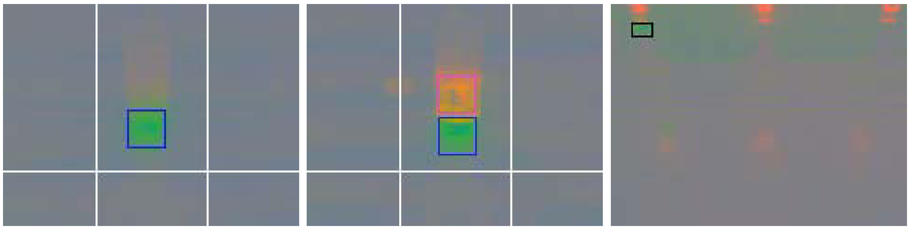
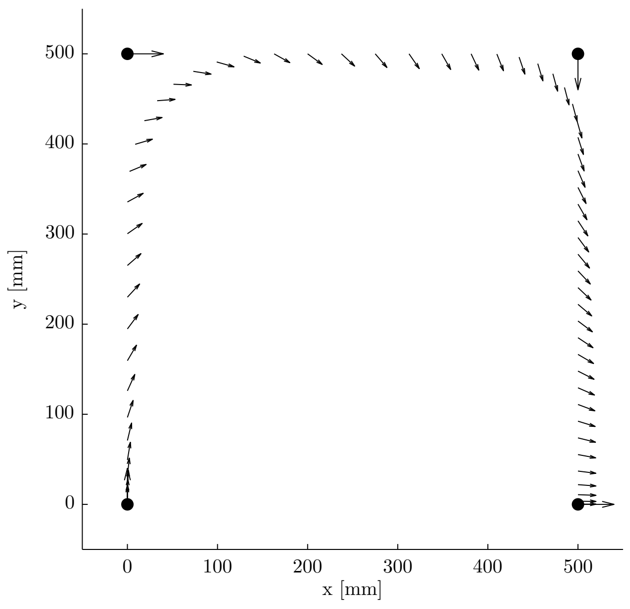


Fig. 6. Light state detection of production machines using modified camera parameters showing green (left), yellow-green (middle), and detecting the currently active delivery gate, that is the one with the green light turned on (right)

raw material zone, delivery gates, etc.) are planned on a coarse 9×9 grid, as visualized in Figure 2 on the right. Each (x, y) grid node expands into four oriented nodes $(x, y, 0°/\pm 90°/180°)$ therefore leading to a shortest path problem in a graph of only 324 nodes. Allowing only four orientations and 81 positions on a rectangular grid, we restrict the motion to right-angled pathways through the factory area. The established coarse "motion corridors" have a slightly larger size than Robotino's diameter. For a smooth trajectory generation along the calculated shortest path, the path is reduced to a list of n via poses $V \in (x, y, \phi)^n$.

Based on the via poses and constraints for maximum velocity and acceleration, a trajectory is calculated following the Linear Segments Parabolic Blends method (LSPB) detailed in [1]. Our trajectories are planned in the world frame, treating x, y and ϕ as independent, but time-synchronized degrees of freedom (Robotino has a holonomic drive system). Generally the LSPB approach is based on linear interpolation between two via points (the "linear segments"). To avoid discontinuous velocity trajectories, parabolic blends are added at each via point. In other words, the robot is allowed to (de)accelerate only near via points, resulting in trapezoidal, continuous velocity trajectories. Finding a trajectory via the LSPB method eventually means to find the minimum linear segment time for each of the $(n - 1)$ path segments and the minimum acceleration time for each via point $v \in V$ under the given dynamic constraints. As you cannot solve for these time intervals in a closed form, Craig suggests a heuristic [1]. We decided for an iterative approach: Starting with the first path segment and zero segment time, we increase the segment time until all constraints are met and the



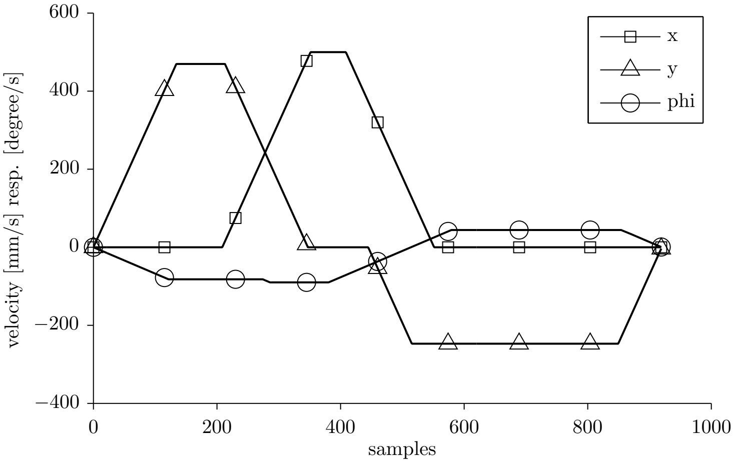


Fig. 7. Generated trajectory sampled at 200 Hz with two intermediate via poses in addition to start and end pose. The quiver plot on top visualizes position and orientation of the sampled trajectory (small arrows at each fifteenth pose), the original via poses are marked by filled dots. Notice how only the start and end pose are actually reached, the intermediate via poses are only approximated. This plays nicely with our coarse grid motion planning. The resulting movement is very smooth, yet still preserves straight line motion on the corridors. The line plot at the bottom represents the corresponding velocity trajectories.

necessary acceleration times are less than half of the segment time. We then move on to the next segment to start with zero segment time again. After defining all time intervals, the trajectory is calculated by sampling it into a look-up table with a sample frequency equal to our desired control loop frequency (an example can be seen in Figure 7).

Trajectory control is then achieved by a position PD-controller targeted at a control loop frequency of 200 Hz. The direct communication with Robotino's real-time subsystem via shared memory allowed us to achieve these short control cycles on Robotino, as discussed in Chapter 2.2 (Figure 3).

3 Conclusion

This paper introduced the RoboCup Logistics League and the 2012 champion TUMsBendingUnits[1].

As the new Logistics League crowned the first world champion this year, our main focus was on establishing a solid and stable overall logistical system. Avoiding cost-intensive sensor equipment like laser scanners, we developed advanced software routines within our event-based state machine, including for example odometry calibration at certain locations or robust steering towards the machines. The overall software architecture allowed us to decouple modules quite easily while the external graphical interface helped us to debug our approaches. Whereas the visual perception algorithms were kept simple, the more sophisticated motion execution, especially the trajectory following, ensured efficient path following, resulting in significant time savings when executing a job. Overall our system allowed all three robots to keep up a stable and coordinated material flow while dealing with the logistical challenges of the competition, including express goods, out-of-order machines, changing delivery gates, and recycling of consumed goods. During the final game no human intervention was necessary and all robots operated autonomously for the whole 15 minutes.

Compared to the other teams of the Logistics League, the communication and collaboration between the robots and the motion routines set us apart the most. Moreover, our software architecture allowed us to easily decouple, debug and visualize our software components, which is of great value in order to achieve the most stable and robust solution. For the next years, we plan to further boost the dynamic and flexible behaviors concerning job handling, collaboration and time-based path planning. Right now, tasks cannot be interrupted and the whole path is reserved when driving to a certain grid location.

With increasing knowledge and performance capabilities of the Logistics League teams, the competition itself will be further improved and refined in the future. Following the roadmap of the Logistics League [4], alongside the core

[1] More information about the team TUMsBendingUnits and video highlights from the competition can be found on the following websites:
`http://www.tumsbendingunits.de/` (under construction)
`http://www.youtube.com/TUMsBendingUnits`

production cycle and the express good challenge, various new production strategies will be introduced, for example Just-in-Time (JIT) and Just-in-Sequence (JIS). Moreover, future dynamic adversarial obstacles will require a much more sophisticated, reactive and flexible behavior of the robots and their collaboration, especially concerning the path planning. These future changes will keep the Logistics League challenging and contribute towards the goal of approaching an industrial application.

Acknowledgments. The authors would like to thank the additional contributing member Peter Gschirr of TUMsBendingUnits, Andre Gaschler from the fortiss GmbH, Dr.-Ing. Gerhard Schrott and Prof. Dr.-Ing. Alois Knoll of the Chair Robotics and Embedded Systems, Department of Informatics, Technische Universität München, for their huge support and commitment.

References

1. Craig, J.J.: Introduction to Robotics: Mechanics and Control, 3rd edn. Prentice Hall (2004)
2. Eureqa, Cornell Creative Machines Lab, `http://creativemachines.cornell.edu/eureqa`
3. Festo Didactic, Education and Research Robots: Robotino, `http://www.festo-didactic.com/int-en/learning-systems/education-and-research-robots-robotino/`
4. Logistics League Roadmap, `http://www.robocup2012.org/pdf/LL_2012_Roadmap.pdf`
5. Logistics League Rulebook, `http://wiki.openrobotino.org/index.php?title=Logistics_League`
6. RoboCup (2012), Sponsored Leagues: Logistics League by FESTO, `http://www.robocup2012.org/comp_SponsoredFesto.php`
7. Richard, S.: Computer Vision: Algorithms and Applications. Springer, London (2010)

Team CHARLI: RoboCup 2012 Humanoid AdultSize League Winner

Coleman Knabe, Mike Hopkins, and Dennis W. Hong

RoMeLa, Mechanical Engineering, Virginia Tech, USA
{knabe,hopkns,dhong}@vt.edu
http://www.romela.org

Abstract. Autonomous soccer-playing humanoid robots have advanced significantly in the past few years. Skill sets elementary to humans such as omnidirectional bipedal walking, path planning, and gameplay strategy have matured enough to allow for dynamic and exciting games. In this paper team CHARLI, the two-time RoboCup Humanoid AdultSize League winner, describes the design and fabrication of essential components such as the spine and mechanical structure, then overviews the increase in performance resulting from recent mechanical upgrades. Finally, we detail the custom walking controller and gameplay module changes responsible for the outstanding performance of our self-constructed lightweight full-sized humanoid platform, CHARLI-2.

1 Introduction

Team CHARLI is a collaborative effort between Virginia Tech's Robotics and Mechanisms Laboratory (RoMeLa) and the University of Pennsylvania's GRASP lab. Stemming from the success of team DARwIn in the KidSize class, team CHARLI (Cognitive Humanoid Autonomous Robot with Learning Intelligence) has participated in the Humanoid AdultSize League since its debut at RoboCup 2010. Having demonstrated the reliability of the 2011 CHARLI-2 platform by winning the Louis Vuitton Best Humanoid Award – the first United States team to secure the trophy in RoboCup history – few modifications were necessary to comply with the 2012 rules. This paper details the design and fabrication issues of selected innovative features of the CHARLI-2 platform, and then overviews the hardware and software changes which steered team CHARLI to its second consecutive AdultSize victory.

2 Design and Fabrication Considerations

The main emphasis of the CHARLI-2 platform, shown in Figure 1, is on a lightweight design to reduce development cost, improve ease of handling, and ensure safe operation. The construction of a reliable full-sized humanoid robot requires several design considerations and manufacturing processes to create multi-purpose subsystems and minimize weight.

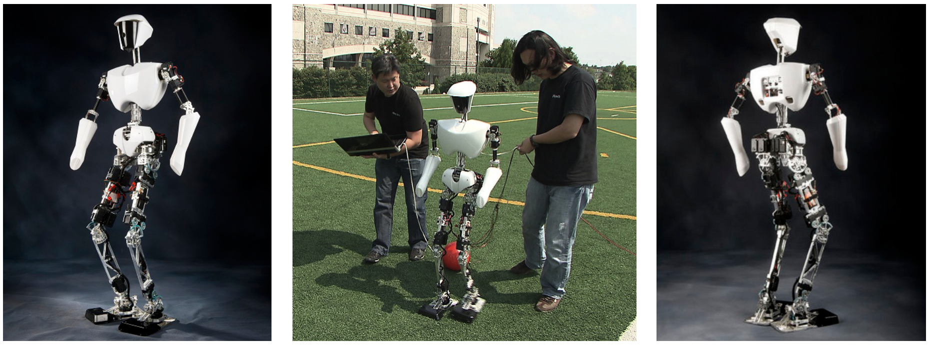


Fig. 1. CHARLI-2 (left, right) and testing on a soccer field (center)

2.1 Spine

Some of the problems faced by full-sized humanoids are ease of handling, operator safety, and a lightweight design; many utilize a bulky handle near the shoulders or require multiple users to maneuver and position the robot. To address these issues, CHARLI employs a unique multifunctional spine design.

Two stainless steel rods are the only load-bearing components needed for the spine. However, an innovative design employing laser-cut acrylic disks resembling human vertebrae not only improves aesthetic appeal, but also allows for routing of wires through the center of the discs. The spine also functions as a comfortable handle well above the robot's center of gravity in an area free of pinch points, which ensures safe, stable, handling and eliminates the added weight of a conventional handle near the head or shoulders.

The base of the spine detaches from the waist, allowing us to separate CHARLI into upper and lower portions for easy transportation. The detachable spine base also permits changes to the waist and pelvis structure, so in the future we plan to implement a waist yaw joint to increase the upper body range of motion.

2.2 Speaker

Design considerations for the speaker system focused on power consumption, packaging, weight, and acoustics. To realize these design requirements a low-power Mylar cone was outfitted in the chest cavity, using the chest covers as a natural enclosure to enhance acoustic quality.

Though we typically use the speaker at demos to communicate and interact with the audience, it is also one of the most useful subsystems for code development and testing. Since it can be difficult to recognize errors in autonomous behavior during runtime, descriptive audio clips — such as *ball found*, *ball lost*, or *orbit right-forward* — were triggered by state machine transitions to provide cues regarding the robot's decision-making. This provides real-time monitoring of autonomous decisions made by the robot, increasing the efficiency with which we can debug the control software.

2.3 Mechanical Structure

Unlike the easily fabricated bent sheet metal components on the KidSize DARwIn-OP, the mechanical structure for CHARLI-2 requires more rigid components to sustain the increased dynamic loads associated with full-sized platforms. Thus, the fabrication of a CHARLI frame is a more involved process than its KidSize counterpart. Several manufacturing processes were considered for their ease of fabrication, time savings, and achievable tolerances, and CNC milling was determined to be the most effective.

The frame is primarily cut from aluminum alloy 6061 due to its machinability, strength-to-weight ratio, and low cost. Figure 2 depicts the procedure used to fabricate the CHARLI frame: we utilize CNC milling machines to cut the aluminum parts and a CAD-CAM program to generate the tool path and G-code required for CNC control. Once milled, parts are removed from the aluminum alloy sheets and post processed – cutting remaining features, threading holes, and removing sharp burrs – to complete fabrication. Using this manufacturing process, we can achieve intricate geometries with the precision and tolerances required for high-performance robotics applications.

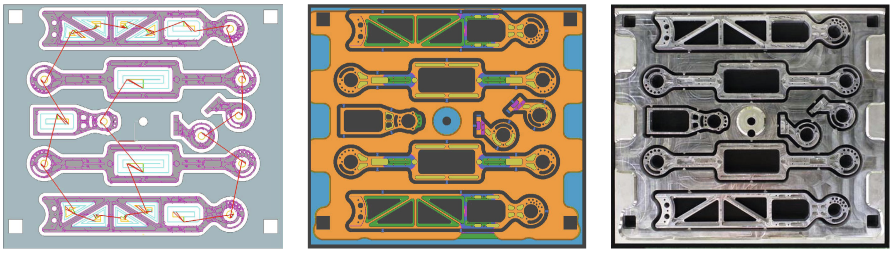


Fig. 2. Left: Designing a tool path. Center: Tool path simulation. Right: Resulting milled aluminum alloy sheet.

2.4 Covers

Covers are a critical robot component, as they improve the safety and reliability of the system and define the basic appearance of the robot. Covers provide an essential barrier between the user and the vital internal circuitry to prevent injury or electronic shorts. We considered several methods of fabricating covers, each with distinct advantages and disadvantages: injection molding is only advantageous for mass production due to its high start-up cost, CNC milling a cover from a large block of stock is extremely time-consuming, and carbon fiber lay ups have low material costs but can result in weakened mechanical properties due to uneven resin distribution.

Instead, fabrication of CHARLI's lightweight covers is accomplished through the cost-efficient, repeatable vacuum forming process depicted in Figure 3 [1]. Molds are carefully designed to avoid overstretching and webbing of the plastic, and are then cut on CNC mills similar to the aluminum parts. Next, thin clear plastic covers are shaped over the molds through vacuum forming. Post processing of the covers involves trimming excess plastic and painting the interior, resulting in the appealing glossy finish characteristic to CHARLI's appearance.

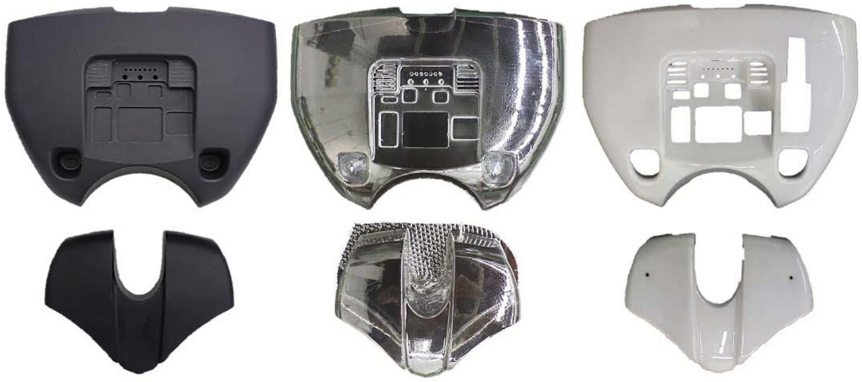


Fig. 3. (Left to right) Cover molds, vacuum formed covers, and post processed covers

3 Mechanical Platform Upgrades

The only major mechanical change to the CHARLI-2 platform from 2011 to 2012 was an upgrade from Robotis's Dynamixel EX-106+ actuators in the legs to the new MX-106 actuators. The MX-series boosts performance over the previous EX-series without a change in actuator cost or dimensions.

A contactless absolute encoder permits 360° rotation of the motors, a 40% increase from the magnetic encoder [2], permitting CHARLI to utilize a larger range of motion. Furthermore, the new actuators require a lower nominal operating voltage, allowing the use of lower voltage leg batteries to reduce weight.

Another new feature is the ability to receive bulk feedback data from the actuators including the actual position, velocity, voltage, and/or current draw. By monitoring the consumed current for each leg actuator, power consumption was analyzed for various walking speeds and trajectories [1].

Despite the increased maximum torque, one tradeoff to using the MX-106 actuators is a 35% reduction in maximum speed inherent to the reduced gear ratio [2], but we did not find this to inhibit walking performance.

4 Codebase Upgrades

One major RoboCup 2012 rule change was a 50% size increase of the AdultSize field, presenting exciting challenges for AdultSize teams. Team CHARLI was able to directly port the majority of the cross-platform software architecture employed by team DARwIn for the KidSize competition [3]. However, CHARLI's custom walking controller and dedicated gameplay module required innovative changes in order to complete the Dribble-and-Kick competition within the effectively reduced timeframe.

4.1 Custom Walking Controller

CHARLI can walk at speeds of up to 0.4 m/s, but this speed was never attained during matches in 2011 due to the inherent reduction in stability at maximum walking velocity. Adapting to the larger field size this year required a faster stable walking gait to minimize the time required to traverse the field. To enhance the ZMP-based walking controller [1],[4] we conducted extensive testing to further understand the multifaceted effects of the walking algorithm parameters on the speed and stability of the gait.

The walking controller is tuned using a number of intuitive parameters such the step period time, double stance phase ratio, and peak foot height. By reducing the step period time and double stance phase ratio, we created a faster, more dynamic walking gait which which was more robust to disturbances. Increasing the proportional feedback gains for the hip and ankle roll and pitch joints provided the stability necessary to maintain balance at higher speeds. Secondary walking controller parameters were then adjusted to fine-tune the walk for various surfaces.

The increased stability of the walking gait improved performance of turning and side-stepping, allowing us to more effectively implement the omnidirectional path planning included in the cross-robot software architecture and improve the aiming accuracy during attacks.

4.2 Gameplay Module

We made several upgrades to the high-level gameplay module to futher reduce the time required to complete each challenge. Ball dribbling was improved by utilizing the side-step to adjust foot alignment during the approach. We also implemented a range of kick speeds to provide more accurate ball placement across the field.

We modified the 2011 goal-scoring algorithm to combine the robot's angle of approach to the ball with an adjustable gain to determine where to aim the kick within the goal. As opposed to consistently kicking towards goal center, this approach can reduce the number of blocked goals; however, shooting accuracy was more sensitive to errors in localization prior to the kick, which can result in missed goals.

Finally, we created a goalie module unprecedented in the AdultSize League. Conventional AdultSize goalkeepers typically remain stationary at the center of the goalie box or, range of motion permitting, temporarily squat with arms extended to block the kick. Inspired by basic human goalie tactics, CHARLI advances within the goalie box toward the incoming attacker in order to block off the angle of attack available to the striker.

5 Conclusions

CHARLI was featured at the 2012 World Expo in South Korea, walking on stage and interacting with the audience 12 hours per day for three months, demonstrating the durability of the CHARLI-2 platform. Team CHARLI is committed to introducing reliable innovative platforms to the humanoid community, and is currently developing a new platform utilizing linear series-elastic actuators and an impedance control walking algoritm capable of safe falling and recovery.

Recognizing the booming success of DARwIn-OP and the unified humanoid robotics codebase, we also have aspirations of releasing an open platform version of CHARLI. We believe this is the most effective method of advancing the field of humanoid robotics and contributes toward the ultimate RoboCup goal of playing soccer with humans.

Acknowledgements. We acknowledge the Office of Naval Research (ONR) and the National Science Federation (NSF) for partially supporting this work through grants ONR 450066 and CNS 0958406.

References

1. Han, J.: Bipedal Walking for a Full-sized Humanoid Robot Utilizing Sinusoidal Feet Trajectories and Its Energy Consumption. PhD thesis, Virginia Polytechnic Institute and State University (April 2012)
2. Robotis Dynamixel Actuators, `http://www.robotis.com/xe/dynamixel_en/`
3. McGill, S.G., Brindza, J., Yi, S.-J., Lee, D.D.: Unified humanoid robotics software platform. In: The 5th Workshop on Humanoid Soccer Robots (2010)
4. Song, S., Ryoo, Y., Hong, D.: Development of an omni-directional walking engine for full-sized lightweight humanoid robots. In: IDETC/CIE Conference (2011)

RoboCup@Work League Winners 2012

Stefan Leibold, Andreas Fregin, Daniel Kaczor, Marina Kollmitz, Kamal El Menuawy, Eduard Popp, Jens Kotlarski, Johannes Gaa, and Benjamin Munske

Institute of Mechatronic Systems, Leibniz Universität Hannover,
Appelstr. 11A, 30167 Hanover, Germany
luhbots@imes.uni-hannover.de
http://www.imes.uni-hannover.de

Abstract. One of today's overall efforts in mobile industrial robotics is the enhancement of autonomy and flexibility considering required safety issues. The new league RoboCup@Work being carried out for the first time in Mexico City, Mexico 2012, focuses on boosting research activities in this field in order to create new, innovative ideas and concepts meeting industrial needs.

This paper introduces the new league. Furthermore, it presents the approaches of the winner team LUHbots, Leibniz Universität Hannover, Hanover, Germany, at each competition in detail.

1 Introduction

RoboCup@Work is a new league of the RoboCup Federation [23]. It was carried out for the first time at the RoboCup 2012 in Mexico City, Mexico. The new league is based on existing RoboCup competitions incorporating proven concepts. However, the applicability and relevance for the industry is high, because deployment of mobile robotics in industrial scenarios is targeted. Furthermore, the new league aims to foster research and development into new and not thoroughly covered areas of industrial robotics, e.g.

– perception, using multiple kinds of sensors and introducing new concepts of sensorics to the industrial environment,
– path and motion planning, adapting established methods and developing new concepts,
– object manipulation,
– planning and scheduling,
– learning, adapting to changing or unknown environments, and
– probalistic modeling.

"Examples for the work-related scenarios targeted by RoboCup@Work include

– loading and/or unloading of containers with/of objects with the same or different size,
– pickup or delivery of parts from/to structured storages and/or unstructured heaps,
– operation of machines, including pressing buttons, opening/closing doors and drawers, and similar operations with underspecified or unknown kinematics,

– flexible planning and dynamic scheduling of production processes involving multiple agents (humans, robots, and machines),
– cooperative assembly of non-trivial objects, with other robots and/or humans,
– cooperative collection of objects over spatially widely distributed areas, and
– cooperative transportation of objects (robots with robots, robots with humans)." [18]

Thus far, a RoboCup@Work competition consists of three parts: Two stages and the finals. Each stage contains multiple tests with varying difficulty. The first stage focuses on basic skills like perception, navigation and manipulation, the second stage consists of more complex tests merging the skills of the first stage and combining them with new elements. The finals consist of one or multiple tests of the previous two stages. Each test is subdivided into single tasks. For each fulfilled task the teams can score a set number of points. These points can forfeit due to collisions or other unwanted behaviors. In addition to the tests, teams can score points within the *Open Challenge*, analogous to RoboCup@Home. The Open Challenge is a free demonstration which is meant to be a playground for innovative ideas and solutions that do not fit into the standard tests. Although the structure of a RoboCup@Work competition is fixed, the contents of the tests varies between competitions.

The mobile edutainment robot KUKA youBot [19] was the basic platform for the teams this time. Any other robot platform meeting the prescribed requirements regarding size and functionality, e.g. having at least one manipulator equipped with a gripper, may be used as well. Since RoboCup@Work is industry orientated, the robots used for the competitions should meet professional quality standards regarding robustness and fashioning. However, a certification for industrial use is not required.

Four teams participated in the first RoboCup@Work 2012 in Mexico City. Our team, the LUHbots, has been founded in 2012 consisting of overall seven diploma, bachelor and master students from the Faculty of Mechanical Engineering at the Leibniz Universität Hannover (Hanover, Germany). The team is promoted by the Institute of Mechatronic Systems, Hanover, Germany.

The remainder of this paper is structured as follows. Section 2 briefly describes the robot platform and the team's modifications of the stock model. Section 3 introduces the tests performed at RoboCup 2012 in detail and our solutions to the posed problems. Section 4 gives an outlook about future work and concludes this paper.

2 Hard– and Software

As mentioned before, the mobile edutainment robot youBot was the basic platform for the teams at RoboCup@Work 2012. By offering the youBot, KUKA has the intention to provide a platform for professional education and development of scalable software components in mobile robotics research [7]. The robot consists of a holonomic platform and a five degrees of freedom (5-DoF) manipulator. Both components can be used separately or connected to each other. It is also possible to use two manipulators on a single moving platform, e.g. for cooperative two arm manipulation. Due to the four Mecanum wheels, the holonomic platform has a high mobility [9]. A Mini-ITX computer with Intel® Atom™ CPU integrated in the robot's base can be used for controlling the actuators. All of them, except for the gripper, provide an EtherCAT® - interface, allowing realtime communication [12]. Initially, the robot consists of rotary encoders and current sensing in each actuator. In order to enable autonomous interaction with the environment and to solve complex tasks, e.g. part handling in unknown environment, the RoboCup@Work rules allow to midify the stock platform with additional sensors. The team's youBots holds additional sensors (see Fig. 1), as descibed in the following. At the front of the youBot a Hokuyo URG-04LX-UG01 laser range finder is attached having a dimension of only 50 mm $\times$ 50 mm $\times$ 70 mm [16]. At 10 Hz, it provides a scan range between 20 mm and 4000 mm and a scan angle of 240°. The pitch angle is 0.36°. Besides others, the laser range finder can be used for mapping, localization, navigation, and safety related applications. A camera mounted at the end effector of the manipulator allows visual servoing. The webcam LifeCam Cinema™ (Microsoft Corporation, Redmond, USA) has been chosen for this task [22]. With its cylindric shape it can be easily mounted on the robot. Providing an appropriate resolution, webcams are much cheaper in comparison to their industrial counterparts.

For RoboCup@Work all robots need to have an emergency stop system. The youBot itself is shipped without any emergency system. Hence, in order to fulfill saftey requirements, a XBee-based wireless emergency system has been developed to stop the robot remotely. In addition to that, an emergency stop button at the back of the robot can be used. Another modification affects the manipulator. Originally it's rotation area covers 169° in both direction assuming that the manipulator points forwards at 0°. For being able to load objects on the youBot's cargo area without any overhead movement the arm is mounted with a static offset of 30°. This enables the up and unloading of objects with only rotational movement along the vertical axis. Thus we are able to reach any point on the cargo area as we get a new coverage of 199° CCW and 139° CW. In conclusion, with this modifications our youBot is technically prepared to cope with the RoboCup@Work tasks.

ROS in combination with the Linux distribution Ubuntu is used as software platform by all teams of this year's RoboCup@Work. Especially ROS as main framework has significant advantages:

- a huge number (more than 3,000) of open source software packages providing various functionalities is available,
- OpenCV [8] is integrated,
- software functionality can easily be split up between team members using ROS nodes,
- developed ROS nodes can be tested independently,
- visualization (rviz) and simulation (gazebo) is already integrated,
- ROS comes with a ready-to-use navigation stack,
- multiple drivers are included such as various webcam drivers and a driver for the hokuyu laser range finder.

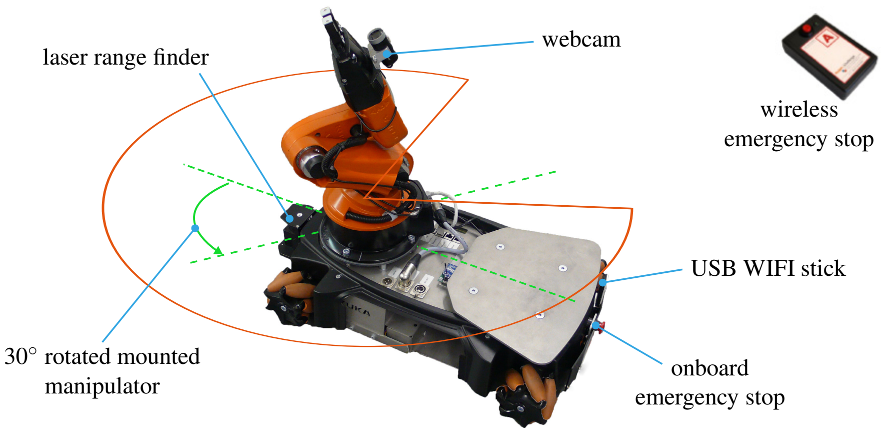


Fig. 1. Modified KUKA youBot equipped with additional sensors

3 RoboCup@Work Tests

The RoboCup@Work tests are processed in an enclosed area (arena) being confined by walls (see Fig. 2). Elevated functional areas with a height of approx. 10cm, also known as service areas e.g. for object manipulation and pick and place assignments, are located immovably. Augmented Reality markers on the walls and floor can be used for navigation (see Fig. 2). Additional static and/or dynamic obstacles may be placed in the arena. Each challenge has to be performed within a specific, fixed time frame. Prior to the start of each test, task specifications are sent to the acting robot by means of a referee-box server. All communication with the robot has to be wireless and any intervention during a run will result in abortion.

In 2012, the first stage was composed of the *Basic Navigation Test* (BNT), the *Basic Manipulation Test* (BMT) and the *Open Challenge* (OC). Combined or competitive tests were not performed this year. Each test was conducted with only one robot in the arena at a time. In the following, the performed tests at RoboCup 2012 are described in detail.

3.1 Basic Navigation Test

Test Description. The purpose of the BNT is to prove the ability of the robot to localize itself and navigate in a known environment. With the use of a given map, it has to navigate autonomously to defined positions within the arena. The positions are tagged by floor markers (see Fig. 2). Points are received for each marker which is completely covered by the robot in a predefined position and orientation. Furthermore, extra points are assigned to the fastest team, presumed that every pose was reached successfully.

Configuration. The ROS navigation stack [21] is adopted to appropriately solve the task. Basically, it makes use of:

– a particle filter based *Adaptive Monte Carlo Localization* algorithm (amcl) [14]
– path planning algorithms with global and local planning according to the starting, current and goal pose

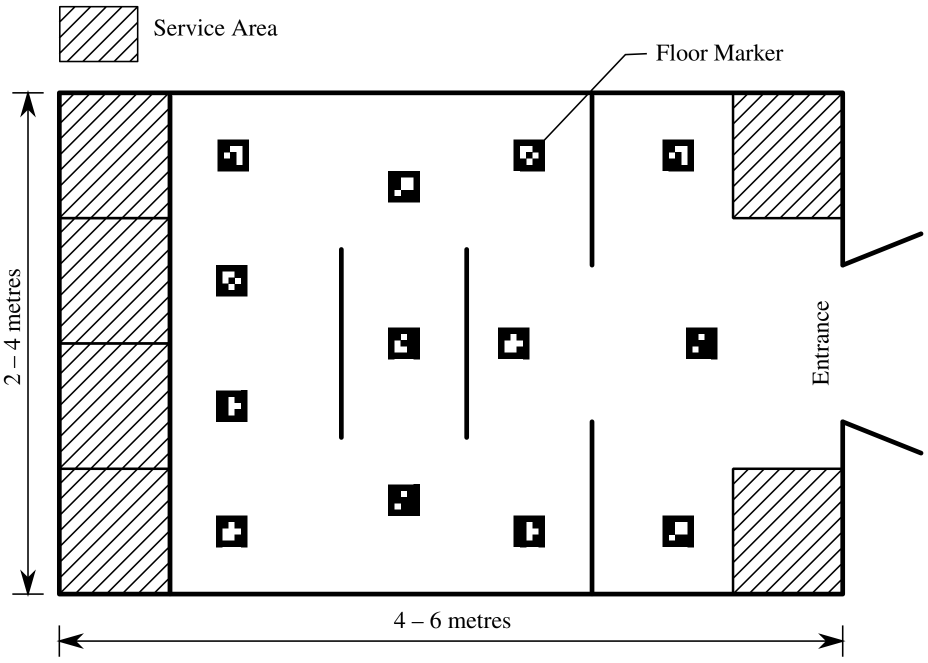


Fig. 2. RoboCup@Work arena 2012 [18]

- an environment map
- the (laser–) sensor and odometry data

The general structure of the stack is visualized in Fig. 3.

The navigation stack is configured for a holonomic robot with rectangular base frame and planar laser scanner input. Furthermore, gmapping [15] is used to create environment maps.

Limitations of the Utilized Navigation Stack. Although the youBot is a holonomic robot, the navigation stack only allows motion in directions that are sensed by the robot, i.e. in the range of the laser scanner in order to prevent collisions. Any backwards or sidewards movements are constrained. Since the floor markers have to be covered completely, the navigation to and the localization at the goal position have to be accurate. Particle spreading and imprecise base-movement result in the need of permanently readjusting the position close to the goal pose. Here, the constrained directions of motion additionally complicate the positioning. The most crucial issues are the positions of floor markers, being close to walls compared to the youBot's dimensions. For obstacle avoidance, the navigation stack inflates the outline of the walls to a certain radius (which ideally represents the longest side of the robot's base) when calculating the safely accessible area. With a radius that is sufficient to prevent collisions, the floor markers are at positions that would demand the robot to trespass the inflated area. This, on the other hand, results in random recovery behavior that handicaps successful navigation.

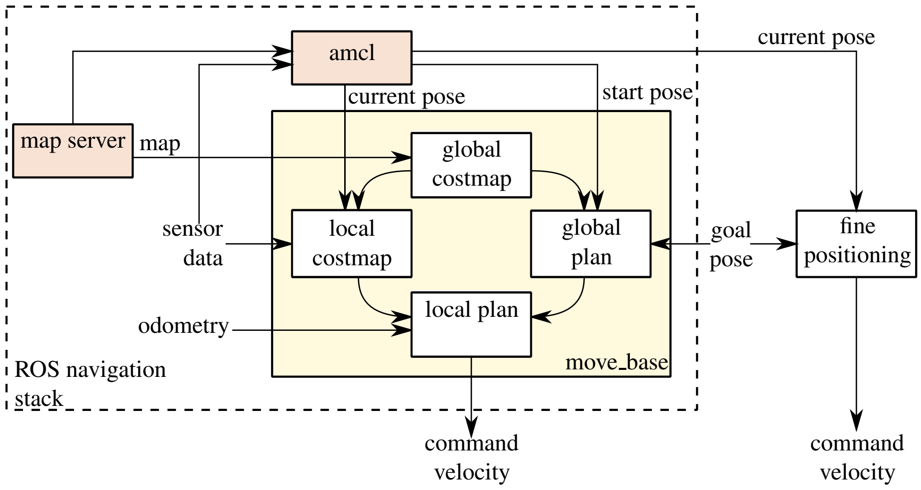


Fig. 3. Scheme of ROS navigation stack [21] and developed fine positioning mode

To sum up, the navigation stack is able to guide the robot close to the goal position. However, it is not possible to reach a desired position with the requested accuracy. In contrast, the disadvantageous goal positions combined with constrained and imprecise movement and particle spreading lead to the robots to perform uncoordinated recovery behavior at the goal positions, not managing to successfully complete the task.

Solutions. The navigation stack is used to guide the robot as close as possible to the requested pose without triggering recovery behaviors. Afterwards, the algorithm switches to an additionally implemented fine positioning mode (Fig. 3). During these time, the current pose is compared to the goal pose on basis of the latest amcl pose estimate. The difference is transformed and the resulting velocity commands are directed to the base actuators. With a constant sampling frequency, the algorithm is looped until a threshold is reached that provides sufficiently accurate positioning.

The developed fine positioning eliminates the constrained movement and reduces the particle spreading thanks to the simple and direct goal approach. The obstacle inflation is neglected because of the very limited space of motion during the fine positioning. Furthermore, no moving obstacles were placed in the arena at the basic navigation test in 2012. Thus, the risk of collisions could be eliminated. As one can see, this approach is able to overcome all the obstacles hindering a successful task completion.

With this considerably simple modification, the LUHbots were able to reach the goals quickly and accurately being the only team performing a complete test run.

3.2 Basic Manipulation Test

Test Description. The aim of the BMT is to prove the ability to recognize different objects and manipulate them. For the RoboCup@Work 2012 the object pool contained ten different objects of silver or black color, e.g. hexagon head and hex socket screws, aluminum profile rails, and screw nuts in various sizes.

In preparation for the test, five objects are nominated and placed in random configuration in a defined service area. The test starts when the product names of three of these objects are sent to the youBot by the referee-box. Those have to be grasped and transported to the neighboring service area.

The task contains three main parts: The recognition of objects, e.g. by means of a camera, manipulation and transportation of the objects with the robot. The scoring rates a robust perception and the ability to distinguish different objects individually. Furthermore, it asks for a fast and correct identification of the objects and for safe manipulation.

Configuration. Many approaches of object recognition with 2D cameras are based on feature detection and matching algorithms, such as SURF or SIFT algorithms [24][20]. Unfortunately, the objects used for the RoboCup@Work do not have enough unique features for a robust identification using these techniques.Therefore, the proposed approach takes the object geometry, i.e. its contour for identification.

For the contour extraction within an image captured using a mono–camera the following image processing pipeline is proposed: The color image is undistorted and converted into a gray scale image. In order to extract the contours, the image is first converted into gray scale and thresholded in order to obtain a binary image. Each object is stored in a database containing specific information concerning the contours, e.g. the aspect ratio. Several following filters delete undefined contours until only geometries within a defined tolerance range remain.

In the following a brief description of each step is given. Fig. 4 gives a overview about the single steps processed by the recognition pipeline. Furthermore, Fig. 5 provides an example of the presented filter stages.

Processing of a Binary Image. For the processing of a binary image three different methods are implemented: A threshold converter [4], an adaptive threshold converter [1] and the canny edge detector [2]. The canny edge detector is the most advanced approach. Nevertheless, regarding the given task, a simple threshold converter achieves the best results in terms of robustness and reliability. With the converter it is possible to separate silver from black objects because pixel that do not fit the target's color are ignored.

Contour Filtering. The distance between the wrist–fixed camera and object is constant. Thus the contour of an object does not vary and a elimination process is capable of identifying the target object. Furthermore the identification does not need to be scale invariant. Nevertheless, since the orientation of the objects is not given, the approach needs to be rotation invariant. The methods used within the proposed contour filtering are a contour size filter, an aspect ratio filter, a size filter and a shape matcher.

– **contour size filter:** To eliminate contours resulting from image interferences a tolerance field is implemented.
– **aspect ratio filter:** The algorithm calculates a minimum rotated bounding rectangle for each remaining closed contour and returns its aspect ratio.
– **size filter:** Objects of the same type and similar aspect ratio, but different size are separated by the absolute size of their bounding box.
– **shape matcher:** For a robust shape matching, this method calculates the Hu-moments [3] up to the third magnitude.

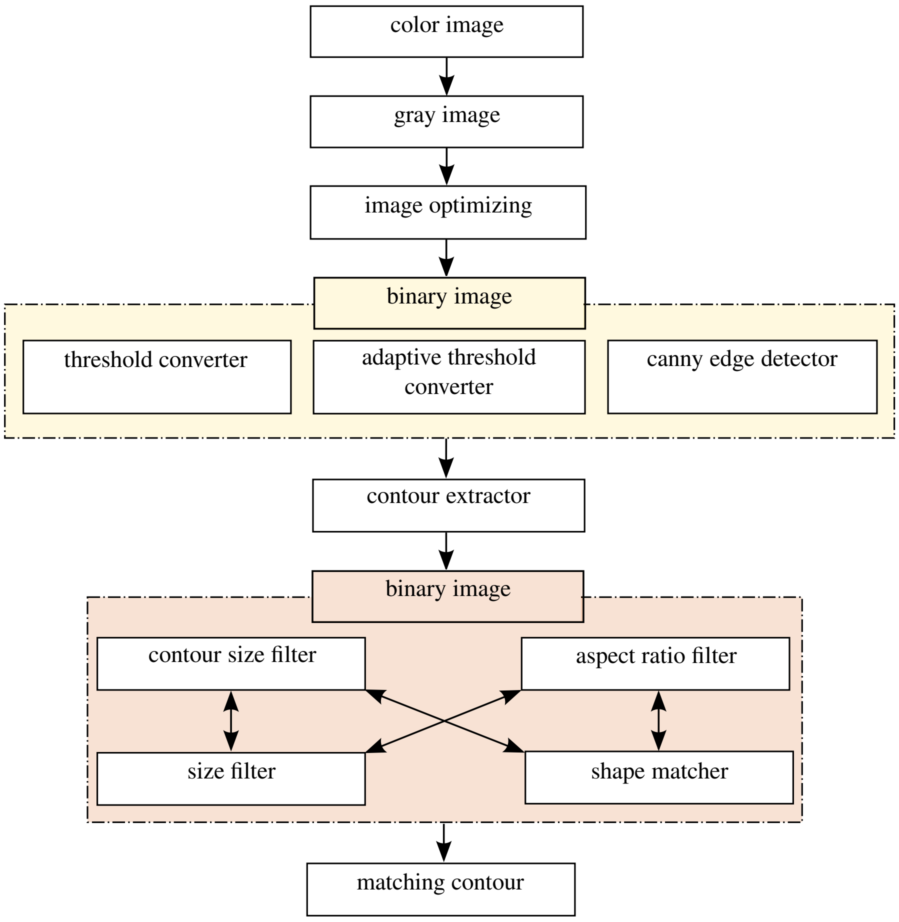


Fig. 4. Different steps of the visions class, that is capable of object recognition and identification using the geometry of an object

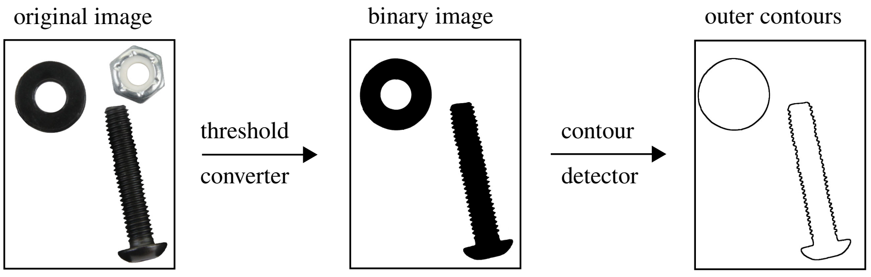


Fig. 5. Example for image processing, from original image to outer contours

Grasping. In order to grasp a detected object, it is necessary to determine its orientation. The aforementioned algorithm generates this information for the filter stages in form of a rotated bounding rectangle. Hence, only the longer side of the rectangle needs to be determined as well as its angle to the horizontal. The centre point of the box serves as the grasping point. Larger screw nuts represent an exception: The space of the gripper jaws is not sufficient to enclose such objects. Therefore, alternative grasping points need to be defined, i.e. by teaching. Since the orientation of the objects can vary the robot has to align itself according to the image before grasping.

Computational Effort. The algorithm works in a resource efficient manner. The image processing is calculated by the on-board computer and, thereby, delivers an average rate of more than 1 Hz. The different filter stages lead to a preselection of suitable objects. Sequencing the filter stages analogous to the needed computing time decreases the overall processing time.

Advanced Features. At the RoboCup@Work 2012 all objects ware placed plain on the service areas. Therefore, a top view on these objects is sufficient. For enabling the identification of arbitrary placed objects in the future, a front view has already been developed. Regarding future requirements, further derivatives of the object recognition class have already been implemented. These extended functions include the removal of image interferences on the binary image as well as the possibility to perform a more detailed image analysis not just per frame but for small sequences. A color filter that constrains a certain colour scale is already included. Furthermore, our vision class contains an implementation of the SIFT algorithm for the identification of textured objects by means of certain characteristics. All methods are modular and can be implemented in different stages of the recognition pipeline.

3.3 Open Challenge

During the Open Challenge each team has the opportunity to demonstrate its own strengths and capacities in five minutes giving a presentation simultaneously. The Open Challenge is evaluated by the following criteria [18]: "

- Relevance and applicability to industrial tasks,
- reuse for different platforms and robustness to different environments,
- professionalism of robot development and use of simulation technologies,
- novelty and scientific contribution,
- difficulty and success of demonstration."

Our presentation is divided into two parts that are described in detail in the following text: The first part presents the *ButlerBot Application* demonstrating object recognition and complex manipulation. The second part contains a human machine interaction.

ButlerBot Application. At the end of a long working day, a cool drink appears like a welcoming refreshment. Treating a robot such an ordinary task is not that simple but requires highly sophisticated image processing and manipulation capabilities. In the proposed application, the youBot retrieves a desired brand of drink and serves it autonomously. The recognition of the bottle requires a two staged solution: In the first step the area is analyzed for objects utilizing a shape detection algorithm. Afterwards, a SIFT algorithm is applied to the preselected shapes. When the correct brand is identified by its label, its position is approximated to grasp the bottle. Finally, the contents of the bottle is poured into a tumbler. This requires a coordinated motion of the manipulator. Therefore, we use a modified path planning based on fourth order polynomial velocity profiles for a smooth and jolt free motion [17]. In order to plan a desired cartesian trajectory considering collisions, endpoints and viapoints [11] are defined. In addition, the path is parameterized in terms of velocity and acceleration. Using the inverse kinematics, the cartesian points are transformed into jointspace, where the path for each joint is calculated. This ensures a safe motion along the desired trajectory with minimized computational load.

Regarding industrial applications, the demonstration shows how hazardous and/or valuable liquids can be handled autonomously by an autonomous robot. In addition, the relevance of the developed methods of image processing and path planning is reinforced.

Human Maschine Interaction. The second part of the LUHbot's Open Challenge presents an approach for a human machine interaction to control the youBot by hand gestures. Therefore, both hands are tracked using an ASUS Xtion Pro Live Camera [6]. It's structed-light 3D sensor uses the aberration of projected light patterns to measure the depth of the environment. The left hand moves the platform and the right hand the robotic manipulator. As a middleware, OpenNI [5] is used to detect the skeleton and return coordinate frames in ROS' tf [13].

After a calibration process, the positions of the current hand fixed coordinate frames are compared to their initial pose. For the base, the difference is used to calculate velocity values for the driver, according to the hand movements (Fig. 6). The youBot arm navigates to teached in positions depending on the performed gesture. A defined final gesture exits the application.

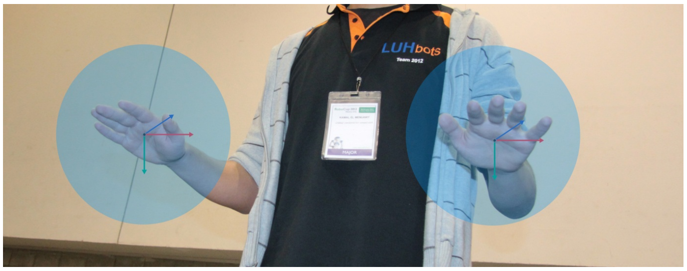


Fig. 6. Hand gesture control of the robot - two coordinate frames are placed inside the user's hand

4 Conclusion and Future Work

In this paper, the new league RoboCup@Work, carried out for the first time in Mexico City, Mexico 2012, was introduced. Furthermore, specific approaches of the RoboCup-@Work team LUHbots were presented.

For enabling new teams to take part in the RoboCup@Work league, the basic tests will remain as a part of the competition. New challenges may include more complex tasks being closer related to real-world problems and interests of the manufacturing industry. Human machine and machine machine interaction will rise in significance [10].

For fullfilling the upcoming requirements, our algorithms as well as our platform will have to get more sophisticated, e.g. the gripper will be modified for grasping objects having greater sizes. The robot is going to be equipped with further sensors for a more robust object detection on structured surfaces. Therefore we currently implement 3D-sensors for extending the perception capabilities to the third dimension. Especially in the logistics context, a huge number of different objects w.r.t. their shape, weight or color, have to be handled. The perception has to be more flexible in order to even manipulate unknown objects. Regarding the fact that in logistics a huge number of objects has to be handled, our target is to become more flexible in handling even unknown objects. An iterative, intelligent grasp algorithm can be considered for this purpose. To see the artifical intelligence in a bigger context, the robot has to be enabled to react autonomously on unpredictable incidents and learn from it. Within the extension of the motion planning functionality of the manipulator, a force and torque controlled solution besides velocity control is in progress in order to implement additional methods for collision avoidance. This extension can be used to increase the joint rigidy in a certain direction to keep the endeffector in a defined workspace. Furthermore the force control can even simplify the uncomfortable process of teaching the path points by conducting the endeffector manually to the goal poses [11].

References

1. OpenCV class description: adaptiveThreshold. Internet (2010), `http://opencv.willowgarage.com/documentation/cpp/imgproc_miscellaneous_image_transformations.html?highlight=threshold#adaptiveThreshold`
2. OpenCV class description: canny. Internet (2010), `http://opencv.willowgarage.com/documentation/cpp/imgproc_feature_detection.html?highlight=canny#Canny`
3. OpenCV class description: matchShapes. Internet (2010), `http://opencv.willowgarage.com/documentation/cpp/imgproc_structural_analysis_and_shape_descriptors.html#matchShapes`
4. OpenCV class description: threshold. Internet (2010), `http://opencv.willowgarage.com/documentation/cpp/imgproc_miscellaneous_image_transformations.html?highlight=threshold#threshold`
5. Introducing OpenNI. Internet (2011), `http://openni.org`
6. ASUS: Xtion pro. Internet (2012), `http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO` Product website
7. Bischoff, R.: KUKA youBot – a milestone for education and research in mobile manipulation. In: IEEE ICRA Workshop – A New Generation of Educational Robots. KUKA Laboratories GmbH, Shanghai International Convention Center, Augsburg, China (2011)

8. Bradski, G.: Open Source Computer Vision Library. Prentice Hall (2004)
9. Campion, G., Bastin, G., Dandrea-Novel, B.: Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. IEEE Transactions on Robotics and Automation 12(1), 47–62 (1996), doi:10.1109/70.481750
10. Celik, I.B.: Development of a robotic-arm controller by using hand gesture recognition. Innovations in Intelligent Systems and Applications, 1–5 (2012)
11. Craig, J.J.: Introduction to Robotics: Mechanics and Control, 3rd edn. Pearson Education Internation (2004)
12. EtherCAT Technology Group: The Ethernet Fieldbus EtherCAT (2009), `http://www.ethercat.org/en/publications.html`
13. Foote, T., Marder-Eppstein, E., Meeussen, W.: tf. Internet (2012), `http://www.ros.org/wiki/tf`
14. Gerkey, B.P.: amcl. Internet (2009), `http://ros.org/wiki/amcl`
15. Grisetti, G., Stachniss, C., Burgard, W.: Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. IEEE Transactions on Robotics 23(1), 34–46 (2007), doi:10.1109/TRO.2006.889486
16. Hokuyo Automatic Co. Ltd.: Scanning Laser Range Finder URG-04LX-UG01 (Simple-URG) - Specifications (2009)
17. Khalil, W., Dombre, E.: Modeling, Identification & Control of Robots, 3. auflage edn. Hermes Penton Science (2002)
18. Kraetzschmar, G., Nowak, W., Hochgeschwender, N., Bischoff, R.: RoboCup@Work Rule Book 2012.4. Internet (2012)
19. KUKA Roboter GmbH: Desktop mobile manipulator for education and research, Product information (2010), `http://youbot-store.com`
20. Labbé, M.: Find-object project homepage (2012), `http://code.google.com/p/find-object/`
21. Marder-Eppstein, E.: Navigation. Internet (2007), `http://ros.org/wiki/navigation`
22. Microsoft Corporation: Microsoft® LifeCam Cinema™. Redmond, USA. Technical data sheet (2011)
23. The RoboCup Federation: Robocup website. Internet (2011), `http://www.robocup.org/`
24. Van de Molengraft, M.: Roboearth project homepage. Internet (2012), `http://www.roboearth.org/`

UT Austin Villa: RoboCup 2012 3D Simulation League Champion

Patrick MacAlpine, Nick Collins, Adrian Lopez-Mobilia, and Peter Stone

Department of Computer Science, The University of Texas at Austin
{patmac,no1uno,alomo01,pstone}@cs.utexas.edu

Abstract. The UT Austin Villa team, from the University of Texas at Austin, won the RoboCup 3D Simulation League in 2012 having also won the competition the previous year. This paper describes the changes and improvements made to the team between 2011 and 2012 that allowed it to repeat as champions.

1 Introduction

UT Austin Villa won last year's 2011 RoboCup 3D simulation competition in convincing fashion by winning all 24 games it played. During the course of the competition the team scored 136 goals and conceded none. This was a vast improvement over the team's previous performance in 2010 when the team finished just outside the top eight. However, despite further improvements for the 2012 competition, the results were much closer this year due to the vast improvement of other teams in the competition.

While many of the components of the 2011 UT Austin Villa agent were reused for the 2012 competition, including that of an optimized omnidirectional walk [1] which was the crucial component in winning the 2011 competition, a number of upgrades were made to the agent to maintain its performance relative to the improvement other teams made between the 2011 and 2012 competitions. Additionally, changes in the rules and format of the 2012 competition, particularly increases in field size and the number of players on a team, necessitated other modifications to the agent be made. This paper is not an attempt at a complete description of the 2012 UT Austin Villa agent, the foundation of which is the same as the 2011 agent fully described in a team technical report [2], but instead focuses on changes made in 2012 that helped the team repeat as champions.

The remainder of the paper is organized as follows. In Section 2 a description of the 3D simulation domain is given highlighting differences from the previous year's competition. Section 3 discusses how a hand-coded get up routine was optimized to make it faster. Section 4 describes an updated kicking system. Changes to a formation and positioning system needed for scaling to 11 agents on a team are detailed in Section 5. Results of the tournament and analysis of improvements to the agent are given in Section 6, and Section 7 concludes.

2 Domain Description

The RoboCup 3D simulation environment is based on SimSpark,[1] a generic physical multiagent system simulator. SimSpark uses the Open Dynamics Engine[2] (ODE) library for its realistic simulation of rigid body dynamics with collision detection and friction. ODE also provides support for the modeling of advanced motorized hinge joints used in the humanoid agents.

The robot agents in the simulation are homogeneous and are modeled after the Aldebaran Nao robot,[3] which has a height of about 57 cm, and a mass of 4.5 kg. The agents interact with the simulator by sending torque commands and receiving perceptual information. Each robot has 22 degrees of freedom: six in each leg, four in each arm, and two in the neck. In order to monitor and control its hinge joints, an agent is equipped with joint perceptors and effectors. Joint perceptors provide the agent with noise-free angular measurements every simulation cycle (20 ms), while joint effectors allow the agent to specify the torque and direction in which to move a joint. Although there is no intentional noise in actuation, there is slight actuation noise that results from approximations in the physics engine and the need to constrain computations to be performed in real-time. Visual information about the environment is given to an agent every third simulation cycle (60 ms) through noisy measurements of the distance and angle to objects within a restricted vision cone (120°). Agents are also outfitted with noisy accelerometer and gyroscope perceptors, as well as force resistance perceptors on the sole of each foot. Additionally, agents can communicate with each other every other simulation cycle (40 ms) by sending 20 byte messages.

For the 2012 competition games consisted of 11 versus 11 agents (up from 9 versus 9 agents in 2011). The field size was also increased to be 20 meters in width by 30 meters in length (the 2011 competition was played on a field 14 meters in width and 21 meters in length).

3 Optimization of the Get Up Routine

A vital skill for an agent in the 3D simulation competition is the ability to stand up from a prone position after having fallen over. In order to get up from a prone position, the robot must choose certain joint angles at certain times to control the movement of its body. The UT Austin Villa team devised a get up routine which iterates through a series of poses, transitioning from one pose to another after a pre-specified period of time. The poses are determined by a series of specified joint angles. Thus, the get up routine can be numerically parameterized by a sequence of time intervals and a set of joint angles. For the 2011 competition these values were chosen and hand-tuned manually.

How quickly a robot is able to recover from a fall is important so that it can rejoin play as fast as possible. For the 2012 competition parameters for the get

[1] `http://simspark.sourceforge.net/`
[2] `http://www.ode.org/`
[3] `http://www.aldebaran-robotics.com/eng/`

up routine were optimized through machine learning to decrease the time needed to stand up. The following subsections explain how this was done.

3.1 Fall Detection and Get Up Motion

The robot detects that it is not upright when its accelerometers indicate that the force of gravity is pulling in a direction not parallel to its torso. When this happens, the robot spreads its arms outward to its side at 90 degree angles. This way, when the robot lands it will fall on either its back or its front. After extending its arms to make sure it falls on its front or its back, the robot pauses for 0.7 seconds before determining which way it has fallen. The agent then proceeds to enter one of two get up routines depending on whether it is lying on its back or front. The get up routine used by a robot lying on its back iterates through the series of poses shown in Figure 1.

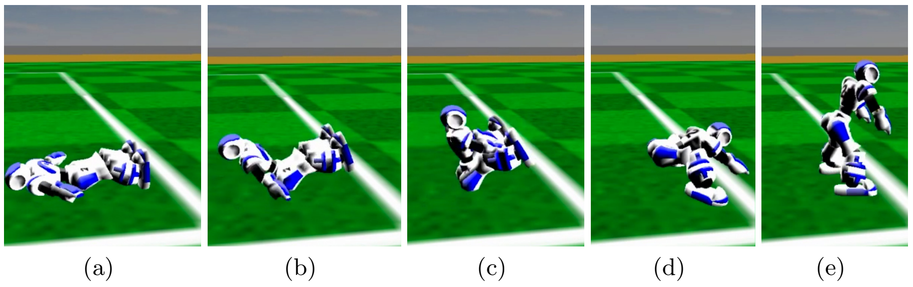


Fig. 1. Routine for getting up after falling backwards. The robot begins lying on its back (a) and then propels itself up with its arms (b). Next the robot throws its arms forward and contracts its legs to get its center of mass in front of its feet (c). Using momentum from the initial push the robot manages to roll into a squatting position (d) after which the robot can get up by extending its knees and hips (e).

3.2 Optimization Process

Parameters for the robot's get up were optimized using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm [3]. CMA-ES was chosen after previously finding it to be the most successful algorithm in optimizing parameters for similar robot skills such as walking forward and turning [4]. CMA-ES is a policy search algorithm that successively generates and evaluates sets of candidates sampled from a multivariate Gaussian distribution. Once CMA-ES generates a group of candidates, each candidate is evaluated with respect to a *fitness* measure. When all the candidates in the group are evaluated, the mean of the multivariate Gaussian distribution is recalculated as a weighted average of the candidates with the highest fitnesses. The covariance matrix of the distribution is also updated to bias the generation of the next set of candidates toward directions of previously successful search steps.

In order to optimize the parameters of the robot's get up routine, the robot is forced to fall and then the time it takes for the robot to get up is measured. A robot that is capable of standing can easily be given a continuous, meaningful fitness based on how long it takes to get up and its stability once standing (as measured by its likelihood of falling back down). Each evaluation begins with the robot being set into an upright, neutral stance by the simulator. The robot is given 1 second to make sure it is stable, and then it is forced to fall backwards. The robot uses accelerometer readings to determine whether it is in an upright position. If it detects that it is not, it will set an internal hasFallen flag and then begin its get up routine. Once the get up routine completes, the agent checks to see if it is upright, and if it is, it clears the hasFallen flag. Otherwise it continues trying its get up routine.

During the evaluation, the robot records how much time the hasFallen flag spends being true, and its fitness for the run is the negation of that. So if it falls once, gets up after 2.5 seconds, and stays up, its fitness is -2.5. After forcing the robot to fall, the evaluation runs for 4 seconds and then ends. If the robot gets up in 2 seconds, but is unstable and falls back down after being up for a second, then its fitness will be -3, since the total time it spent falling or getting up was 3 seconds. Punishing subsequent falls serves to ensure the get up routine is stable.

Additionally, in order to make sure it is stable enough to walk, the robot is asked to perform a movement action after getting up. A complete evaluation trial consists of seven falls and subsequent get ups where after each get up the robot does one of the following: walks forwards, walks backwards, walks left, walks right, turns left, turns right, or stands still. The average across all seven evaluations gives the fitness score for a trial.

3.3 Optimization Results

The optimization process discussed in Section 3.2 was performed on the routines for both the robot getting up from its front and back. Each optimization was run across 200 generations of CMA-ES, using a population size of 150, and was seeded with the original get up sequence hand-tuned parameter values (consisting of joint angle positions and time intervals between poses). Information about the number of parameters optimized, as well as the improvement in speed after optimization are shown in Table 1. Both optimizations were able to reduce the time required to get up to almost a third of their original hand-tuned times.

Table 1. Get up optimizations with the number of parameters optimized and the time in seconds taken to get up before and after optimization

Optimization	Parameters	Hand-tuned Time (s)	Optimized Time (s)
Get Up from Front	9	2.62	0.96
Get Up from Back	26	2.20	0.84

4 Kicking

For the 2012 competition UT Austin Villa switched from a kicking system using directional inverse kinematics based kicks [5] to a hybrid system that has both fixed pose keyframe (better for distance) and inverse kinematics (more robust) based kicks. The parameters for all kicks were optimized using the CMA-ES algorithm described in Section 3.2. The following subsections detail the design and optimization of the kicking system.

4.1 Fixed Pose Keyframe Kicks

Fixed pose keyframe kicks consist of a sequence of body positions, defined by different joint angle positions, which the agent proceeds through in order to kick the ball. Three such kicks were used in the 2012 competition: KickLong, KickMedium, and KickQuick. For each of these the agent first places its support (non-kicking) leg near the ball and shift its weight to the support leg. Next it lifts its kicking leg, and pulls it backward, before finally swinging its kicking leg forward to strike the ball.

KickLong kicks the ball the farthest of the three fixed pose keyframe kicks and is only used on kickoffs. This kick's primary purpose is to push the ball as far as possible into the opponent's end of the field on a kickoff. KickLong also kicks the ball high in the air such that opposing agents can not block the kick as the ball travels over their heads. The extreme motion used by the agent to propel the ball causes the agent to fall flat on its back at the end of the kick.

KickMedium is a kick that also gets a lot of distance, but allows for the agent to remain stable (not fall over) at the end of the kick. KickMedium is used for free kicks as well as during regular play. The kick takes over two seconds to get off thus requiring opponents be at least 2.5 meters away before starting the kick.

Although it doesn't get as much distance as KickMedium, KickQuick is much faster to get off as it takes less than a second to make contact with the ball. KickQuick is designed for quickly kicking the ball when opponents are closing in, and on average gets enough height on the ball to chip it over approaching opponents' heads. KickQuick only requires that the closest opponent be at least 1.0 meters away before starting the kick. Like KickLong, KickQuick destabilizes the agent and causes it to fall over at the conclusion of the kick.

More information about the fixed pose keyframe kicks are found in Table 2.

4.2 Inverse Kinematics Based Kicks

A weakness of the fixed pose keyframe kicks in Section 4.1 is that they require very precise positioning relative to the ball (discussed in Section 4.3) in order for them to be executed. An alternative to this is to define a path relative to the ball that the robot's foot should follow during a kick, and then use inverse kinematics to move the foot along this path. The main advantage gained through such an approach is that a kick is able to adapt to the position of the ball and thus does not require as precise positioning by an agent to line up the kick.

As described in [5], the UT Austin Villa team constructs an inverse kinematics based kick (KickIK) by specifying waypoints relative to the ball for the foot to travel through, and then interpolates between these points using a Cubic Hermite Spline curve to determine the trajectory of the foot's path during a kick. Figure 2 shows the relative waypoints for a forward kick. Inverse kinematics for the agent are computed using OpenRAVE's [6] kinematics solver.

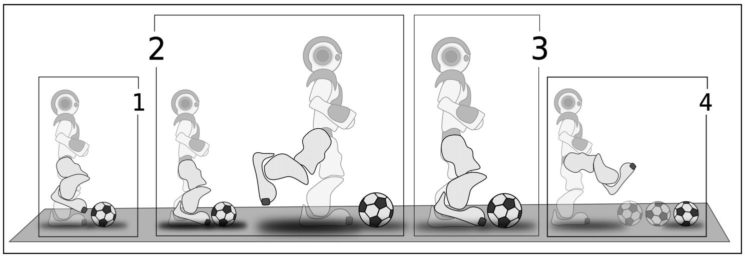


Fig. 2. Waypoints relative to the ball that define the path of the foot for an inverse kinematics based kick. (1) Lift leg to center behind ball. (2) Pull leg back from ball. (3) Bring leg back to position of ball. (4) Kick through ball.

4.3 Kick Positioning

Outside of the kicking motion itself, how a robot positions itself in proximity to the ball before executing a kick is probably the most critical component to successfully striking the ball. When lining up to kick the ball, the UT Austin Villa agent first approaches a target position behind the ball to kick from. The agent is not allowed to proceed with the kick until it is within certain distance thresholds of this target position both along the vectors perpendicular and parallel to the ball from the target position. Before executing a kick the agent must also be within a set angular threshold of facing toward the ball.

Using distance and angle thresholds when positioning to kick is a change from UT Austin Villa's 2011 inverse kinematics based kicking system's positioning [5]. The 2011 agent's kick was triggered as soon as inverse kinematics calculations determined the robot's foot could reach all necessary points along a curve to kick through the ball. After the 2011 competition it was found that using inverse kinematics calculations as a trigger for when to kick is problematic for a moving robot. This is due to the robot's momentum causing its body's position and orientation relative to the ball to change right after deciding to kick. These changes in position and orientation, although often quite small, are enough to prevent the robot's foot from being able to reach the ball and force the robot to reposition itself after aborting the kick.

4.4 Optimization Process

The CMA-ES algorithm discussed in Section 3.2 is used to optimize joint angles for the fixed pose keyframe kicks mentioned in Section 4.1, as well as the X, Y, and Z positions of the waypoints defining a curve for an inverse kinematics based kick detailed in Section 4.2. Roll, pitch, and yaw positions of the foot are also optimized for each of the waypoints of an inverse kinematics based kick. Additionally, for all kicks, the positioning parameters discussed in Section 4.3 of a target point behind the ball, and distance and angle thresholds for being in position to kick, are learned.

When optimizing kick parameters the ball is placed at the center of the field and the agent, placed 1.5 meters behind the ball (or directly behind the ball in the case of KickLong as it is only used for kickoffs), is asked to walk forward and kick the ball toward the center of the opponent's goal. An agent is given a reward for how far it is able to kick the ball in the forwards direction (*distForward*). To promote accuracy a slight penalty is also given for the distance the ball is kicked to either side (*distSideways*). Additionally, as it is important to quickly position behind the ball so as to kick it before an opponent approaches, a penalty is given for the amount of time it takes to position for a kick (*timePositioning*). The following equation gives the reward an agent receives when performing a kick (where distances are in meters and time is in seconds):

$$reward = distForward - .75 * distSideways - timePositioning/8.0$$

If, while positioning to kick, the agent should run into the ball causing the ball to travel greater than .3 meters from its starting spot, a reward of -1 is given for the kick. This is done to ensure the agent doesn't cheat during optimizations by dribbling the ball forward before kicking to gain extra distance. Also the agent is given a reward of -1 when kicking if it falls over while attempting kicks for which it is expected to be stable after performing (KickMedium and KickIK).

All kick optimizations were done across 200 generations of CMA-ES using a population size of 150. Ten kicks were performed for each candidate set of kick parameters being evaluated. Candidates were then assigned a fitness value equal to the average reward of these kicks.

4.5 Optimization Results

Results of optimizing UT Austin Villa's different kicks are shown in Table 2. KickLong was seeded with the 2011 team's kick used for kickoffs. The 2011 kick was optimized in a similar fashion to the 2012 kicks, however for 2012 six additional parameters were learned for adjusting the joint angles of the support (non-kicking) leg. Adding these parameters to the optimization, which increased the number of parameters optimized from 18 to 24, provided a huge performance boost as the kick distance more than doubled from the 2011 kick seed's distance of 5.3 meters. Allowing the agent to fall over after kicking, as opposed to requiring it to be stable as was done in 2011, resulted in a further gain in performance of about a meter as it allowed the agent to throw its body at the ball.

KickMedium and KickQuick were designed from the same seed as KickLong except some of the frames of motion for them were sped up or removed in order to make the kicks faster to get off. As the speed of the seed kick for

KickQuick had to be greatly modified to get it to kick over twice as fast as the original 2011 kick seed, all possible joint angles on the kick were opened up for optimization resulting in more than double the amount of parameters being optimized compared to that of KickLong and KickMedium. Both KickMedium and KickQuick's kick distances were optimized to be well over that of the 2011 kick.

While KickIK has the shortest distance of all the kicks, it is the fastest to get off and, due to its use of inverse kinematics, is generally more robust than the fixed pose keyframe kicks. Additionally, as inverse kinematics allow the kick to adjust to different ball positions, the optimized thresholds for positioning behind the ball can be greater than that of the fixed pose keyframe kicks allowing for faster kick positioning. The time required to position for KickIK is noticeably faster than the time taken to position using the 2011 inverse kinematics based kicks. This is due to the 2011 kicks using inverse kinematics calculations instead of distance and angular thresholds as a trigger for when to kick.

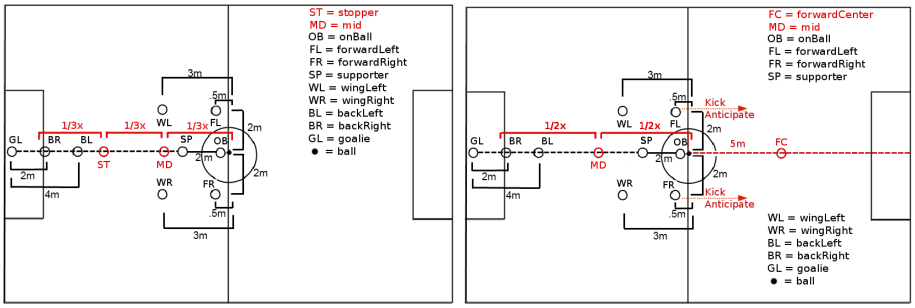
It is worth mentioning that while many of the kicks get a lot of height, which is great for kicking over opponents, height was never something that was optimized for. Optimizing for distance results in the ball being kicked in the air as it is able to travel farther when airborne with no friction from the ground slowing it down.

Table 2. Kick optimizations with the number of parameters optimized, the maximum height and distances recorded from ten kicks (with the median value shown in parentheses), the time taken to execute each, and also whether or not the agent is stable and doesn't fall over after executing the kick

Kick	Parameters	Distance (m)	Height (m)	Time (s)	Stable
KickLong	24	12.20 (11.86)	1.57 (1.36)	2.44	No
KickMedium	24	10.80 (10.46)	1.30 (0.59)	2.12	Yes
KickQuick	51	8.79 (7.30)	1.11 (0.99)	0.92	No
KickIK	42	6.05 (4.82)	0.25 (0.06)	0.25	Yes

5 Dynamic Positioning

With the increase in team size from 9 to 11 agents for the 2012 competition two new role positions were added to UT Austin Villa's base formation: *stopper* and *mid* roles. Both role positions, shown in Figure 3(a), stay on a line running from the center of the goal to the ball. The *stopper* role stays 1/3 of the way between the top of the goal box and the ball while the *mid* role stands 2/3 of the way between these two points. UT Austin Villa also created a more offensive formation designed to take advantage of its new longer kicks described in Section 4. This formation, shown in Figure 3(b), replaces the *stopper* role with a *forwardCenter* role positioned 5 meters beyond the ball along a line from the ball to the center of the opponent's goal. The *mid* role is pushed back to be halfway between the top of the goal box and the ball. A key feature of this formation is the notion of *kick anticipation* where an agent attempting to kick the ball alerts its teammates of the target position the ball is being kicked to. When this target position is



(a) Base Formation (b) Kicking Formation

Fig. 3. Formations used by UT Austin Villa. Added positions for the 2012 competition are shown in red.

broadcasted both *forwardLeft* and *forwardRight* role positions move to the area that the ball is being kicked to in anticipation of the kick.

The UT Austin Villa team used a dynamic role and formation positioning system described in [7] to position its players. This system, at its base, assigns agents to the precomputed role positions on the field so as to avoid collisions and minimize the longest distance any agent has to travel. The positioning system is similar to the one used for the 2011 competition, but was upgraded with enhancements listed in [7]: using path costs and assigning the *supporter* (previously called *stopper* in [7]) role to the nearest agent.

When considering assignments of agents to positions there are $n!$ possible combinations. Using dynamic programming the positioning system only needs to evaluate $n2^{n-1}$ assignments of agents to positions in order to compute an optimal assignment. With the increase from 9 to 11 agents for the 2012 competition scalability becomes a concern, however, because all computations must be performed within 20 ms (the cycle time of the server). As the goalie positions itself only $n = 8$ or 1024 combinations were required to be computed in 2011, but this jumped to $n = 10$ or 5120 combinations to process for the 2012 competition. Despite only needing 3.3 ms to calculate positioning in 2011, the 5X increase in positioning computations for 2012 took up most of an agent's allotted processing time, and left it with little time for other components to do necessary computations.

In order to keep the positioning system from taking too long, a self-monitoring mechanism was put in place where the agent records the amount of time taken to compute positioning role assignments as well as the number n of agents it has assigned to role positions. Should the positioning system take longer than MAX_TIME (set to 10 ms) to run, the agent reduces the maximum number of agents ($maxN$) the positioning system is allowed to evaluate by setting $maxN = maxN - 1$. Alternatively if the positioning system takes less than $MAX\_TIME/2$ to complete then the number of allowed agents to evaluate for positioning is increased by setting $maxN = maxN + 1$. When $maxN$ is less than the number of n agents that need to be positioned then $n - maxN$ agents furthest from the ball are greedily assigned to their nearest role positions. The intuition in greedily assigning agents furthest from the ball to possibly suboptimal role position assignments is that they are less critical to game performance compared to agents closer to the ball. By monitoring the running time of the positioning system, and reducing how many computations it does if it is taking too long, the system can scale to different numbers of agents as well as adapt on the fly to computers with different processors and fluctuating CPU loads.

6 Tournament Results and Analysis

In winning the 2012 RoboCup competition UT Austin Villa finished with a record of 12 wins, 2 losses, and 3 ties.[4] During the competition the team scored 39 goals and only conceded 4. This was not nearly as dominant of a performance as was seen in the 2011 competition when the team won all 24 games it played while scoring 136 goals and conceding none. Several reasons can be attributed to this dip in performance. There was a general decrease in goals scored during the tournament due to the larger field and increase in the number of agents on a team. Additionally early in the tournament there were network problems causing instability that resulted in many teams' agents to lose their balance and have trouble walking. This was very noticeable during the first round when UT Austin Villa suffered both of its losses. UT Austin Villa eventually beat both the teams it lost to (magmaOffenburg and RoboCanes) during the semifinals and finals rounds. A large amount of credit must also be given to the other teams in the tournament as they exhibited a substantial improvement in overall play.

As reported in [5], the 2011 UT Austin Villa team was able to beat all teams in the 2011 competition by at least 1.45 goals on average, and when playing 100 games against every team from the 2011 tournament, UT Austin Villa won every game but 21 of them which were ties (no losses). As seen in Table 3, the 2012 UT Austin Villa team was only able to beat the 2012 2nd place team (RoboCanes) by an average of 0.88 goals and tied them 32 times across 100 games. Although the data in Table 3 shows that UT Austin Villa winning the 2012 RoboCup competition was statistically significant, and that the team didn't lose any games or concede any goals against the other top teams, there was a decent chance of the tournament being decided by penalty kicks due to UT Austin Villa tieing the 2nd place team almost 1/3 of the time. It is thus not surprising that the championship game wasn't decided until the second half of extra time (which UT Austin Villa won 2-0).

It is worth mentioning that the optimized get up mentioned in Section 3 for when an agent is lying on its front (a situation that occurs much less frequently than that of an agent lying on its back) was never used in the competition as the tournament's early network problems, and resulting agent instability, were causing the get up routine to fail. Also it was noticed that kicking was often

[4] Full tournament results, as well as a highlights video of the competition, can be found at
`http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2012/html/results_3d/`

Table 3. UT Austin Villa's released binary's performance when playing 100 games against released binaries of the 2nd, 3rd, and 4th places teams in the tournament: RoboCanes, Bold Hearts, and magmaOffenburg respectively. Values in parentheses are the standard error.

Opponent	Average Goal Difference	Record (W-L-T)	Goals (For-Against)
RoboCanes	0.88 (0.08)	68-0-32	88-0
Bold Hearts	1.64 (0.09)	89-0-11	164-0
magmaOffenburg	1.87 (0.10)	94-0-6	187-0

just turning the ball over to the other team, and so in the later rounds of the tournament the kicking formation was abandoned in favor of the base formation (Figure 3), and the agent only kicked if it thought the kick would score a goal.

In order to quantify gains in performance due to improvements in the agent for the 2012 competition, versions of the UT Austin Villa agent missing different improvements were created and then played against the other teams that made the semifinals. This includes an agent that does not use the optimized get ups in Section 3, an agent without the improved kickoff using KickLong, as well as one that does use KickLong on kickoffs but otherwise just dribbles (shown to be the best performing agent for 2011 in [5]) instead of using any of the other optimized kicks discussed in Section 4, and an agent using the base formation (Figure 3(a)) and positioning system scaled to support 11 agents, but without the improvements to positioning mentioned in Section 5. Additionally an agent missing all improvements was evaluated, as well as a version of the agent using the kicking formation (Figure 3(b)) and kick anticipation. Game performance of different agent variants can be seen in Table 4.

Table 4. Average goal difference when playing 100 games against the released binaries of the other teams in the semifinals: RoboCanes, Bold Hearts, and magmaOffenburg

Agent	Average Goal Difference
2012 UT Austin Villa Released Binary	1.46
No Improved Kickoff	1.37
No Kicking	1.36
No Improved Positioning	1.19
No Improved Get Up	.95
No Improvements (2011 Base)	.92
Kicking Formation with Kick Anticipation	.82

In Table 4 we see that all improvements to the agent were beneficial as missing any single one of them hurt performance and resulted in a lower average goal difference. The most important improvement was that of the optimized get up which resulted in approximately a half a goal increase in performance against the other top teams at the competition. Without any of the improvements for 2012 the team's average goal difference dropped by over half a goal. This includes a 0.59 average goal difference against the 2nd place team (RoboCanes) resulting in a nearly even split between the number of games won and tied against this opponent. It is fortunate that the kicking formation with kick anticipation was abandoned midway through the tournament as it gave the worst performance.

7 Conclusion

UT Austin Villa, bolstered by improvements to its get up routine, kicking, and positioning systems, repeated as 3D simulation league champions at the 2012 RoboCup competition. Although the team was still largely able to lean on dribbling using its stable and fast omnidirectional walk [1] to win the competition, the focus of the team for the 2013 competition will be to continue to improve on its kicking system and integrate passing into the team's strategy. It became clear during the championship match,[5] during which the team was unable to score until the second period of extra time — capped off by a last second goal on a kick from outside the goal box, that kicking will need to be a vital part of the team's strategy if UT Austin Villa is to win a third championship in a row.

Acknowledgments. This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-0917122), ONR (N00014-09-1-0658), and the Federal Highway Administration (DTFH61-07-H-00030). Patrick MacAlpine is supported by a NDSEG fellowship.

References

1. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and optimization of an omnidirectional humanoid walk: A winning approach at the RoboCup 2011 3D simulation competition. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2012) (2012)
2. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011 3D Simulation Team report. Technical Report AI11-10, The Univ. of Texas at Austin, Dept. of Computer Science, AI Laboratory (2011)
3. Hansen, N.: The CMA Evolution Strategy: A Tutorial (2009), `http://www.lri.fr/~hansen/cmatutorial.pdf`
4. Urieli, D., MacAlpine, P., Kalyanakrishnan, S., Bentor, Y., Stone, P.: On optimizing interdependent skills: A case study in simulated 3d humanoid robot soccer. In: Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011) (2011)
5. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012) (2012)
6. Diankov, R., Kuffner, J.: Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA (2008)
7. MacAlpine, P., Barrera, F., Stone, P.: Positioning to win: A dynamic role assignment and formation positioning system. In: Stone, P. (ed.) RoboCup 2012. LNCS (LNAI), vol. 7500, pp. 190–201. Springer, Heidelberg (2013)

[5] Videos of the championship match, as well as more information about the UT Austin Villa team, can be found on the UT Austin Villa team's homepage: `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/`

RoboCup 2012 Best Humanoid Award Winner NimbRo TeenSize

Marcell Missura, Cedrick Münstermann, Malte Mauelshagen, Michael Schreiber, and Sven Behnke

Autonomous Intelligent Systems, Computer Science, Univ. of Bonn, Germany
{missura,behnke}@cs.uni-bonn.de, schreiber@ais.uni-bonn.de
http://ais.uni-bonn.de

Abstract. Over the past few years, soccer-playing humanoid robots advanced significantly. Elementary skills, such as bipedal walking, visual perception, and collision avoidance have matured enough to allow for dynamic and exciting games. In this paper, team NimbRo TeenSize, the winner of the RoboCup 2012 Best Humanoid Award, presents its robotic platform and its approaches to perception and behavior control.

1 Introduction

In the RoboCup Humanoid League, mostly self-constructed robots with a human-like body plan compete with each other. The league comprises three size classes: KidSize (<60 cm), TeenSize (90–120 cm), and AdultSize (>130 cm). The TeenSize robots started to play 2 vs. 2 soccer games in 2010 and moved to a larger soccer field of 9×6 m in the year 2011. This year, a 3 vs. 3 demonstration game showed that –in principle– TeenSize robots are ready to play soccer the way it is done in the KidSize class, given enough participating teams and robots. In addition to the soccer games, the robots face technical challenges, such as throwing the ball into the field from a side line.

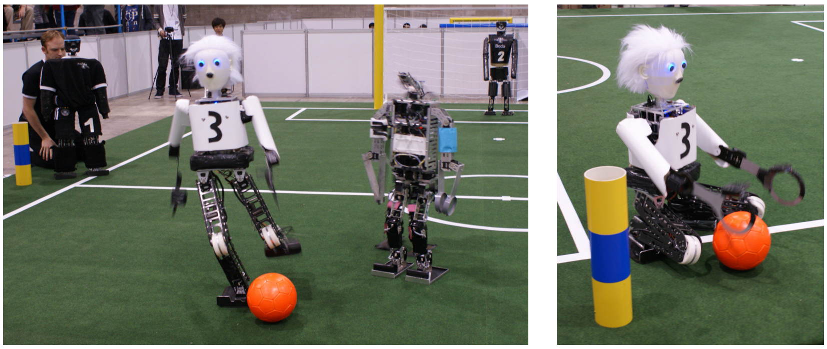


Fig. 1. Left: NimbRo robots Dynaped, Copedo, and Bodo playing in the 3 vs. 3 demo game. Right: Copedo performing the ThrowIn Challenge.

Team NimbRo has a long and successful history in RoboCup with overall ten wins in international Humanoid League competitions since 2005. In 2012, our team won the TeenSize competition for the fourth time in a row and completed the Technical Challenge with the maximum possible score. We have been awarded the Louis Vuitton Best Humanoid Cup for the second time.

2 Mechatronic Design of NimbRo TeenSize Robots

The mechatronic design of our robots, which are shown in Fig.1, is focused on robustness, weight reduction, and simplicity.

Copedo: Our main innovation for the RoboCup competition this year was the construction of a new TeenSize robot that we named "Copedo" (Figure 1). Copedo is 114 cm tall and weighs 8 kg. Its body plan is derived from its successor Dynaped, including the 5-DOF legs with parallel kinematics (Fig. 2(a)) and the spring-loaded passive joint between the hip and the spine (Fig. 2(b)). Copedo, however, is equipped with an additional protective joint in the neck to protect the head. Our new generation of protective joints is now able to snap back into position automatically after being displaced by mechanical stress, such that the robot remains operational after falling to the ground and does not need to be set manually. Copedo is constructed from milled carbon fiber parts that are assembled to rectangular shaped legs and flat arms. The torso is constructed entirely from aluminum and consists of a cylindric tube that contains the hip-spine spring and a rectangular cage that holds the information processing devices.

Most importantly, Copedo is equipped with 3-DOF arms that include elbow joints to enable the robot to get up from the ground and to pick up the ball from the floor and to throw it (Figure 1, right). Including a neck joint to pan the head, Copedo has 17 actuated DOF. The hip roll, hip pitch, and knee DOF are actuated by master-slave pairs of Dynamixel EX-106+ servo motors. All other DOF are driven by single motors including EX-106+ motors for ankle roll, EX-106 motors for hip yaw and shoulder pitch, RX-64 motors for shoulder roll and elbow, and an RX-28 motor for the neck yaw joint.

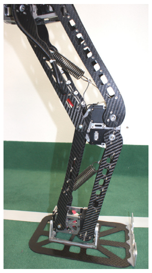
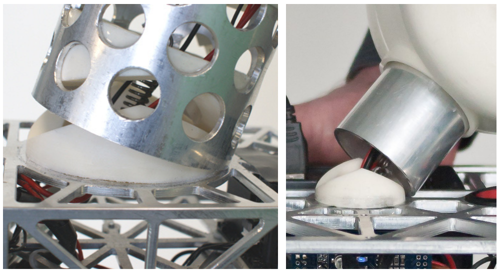
(a)

(b)


Fig. 2. Mechanical construction of Copedo: (a) leg with parallel kinematics; (b) spring-loaded overload protection in the hip and the neck joint

3 Perception

For visual perception of the game situation, we process 752×480 YUV images from a IDS uEye camera with fish eye lens. We detect the ball, goal-posts, poles, penalty markers, field lines, corners, T-junctions, X-crossings, obstacles, team mates, and opponents utilizing color, size and shape information. We estimate distance and angle to each detected object by removing radial lens distortion and by inverting the projective mapping from field to image plane. To account for camera pose changes during walking, we learned a direct mapping from the IMU readings to offsets in the image.

For proprioception, we use the joint angle feedback of the servos and apply it to the kinematic robot model using forward kinematics. Before extracting the location and the velocity of the center of mass, we rotate the kinematic model around the current support foot such that the attitude of the trunk matches the angle we measured with the IMU. Temperatures and voltages are also monitored for notification of overheating or low batteries.

For localization, we track a three-dimensional robot pose (x, y, θ) on the field using a particle filter [1]. The particles are updated using a linear motion model. Its parameters are learned from motion capture data [2]. The weights of the particles are updated according to a probabilistic model of landmark observations (distance and angle) that accounts for measurement noise. To handle unknown data association of ambiguous landmarks, we sample the data association on a per-particle basis. The association of field line corner and T-junction observations is simplified using the orientation of these landmarks. Further details can be found in [3] and [4].

Learning Colors of Unknown Balls: This year, for the first time, the robots had to learn to recognize an unknown ball in the Obstacle Avoidance and Dribbling Challenge. To this end, we defined a region of interest in the field-of-view of the robot, which contained only the field color (green carpet) and the unknown ball (Fig. 3(a)). In this area, we segmented all colors different from the field color, white, and black (Fig. 3(b)). The remaining color histograms were thresholded with a minimum color count and smoothed. We fitted a Gaussian mixture model to the colors of the unknown ball and used its parameters to initialize the ball color in our color table (Fig. 3(c)). Dynaped was the only TeenSize robot to complete this challenge (Fig. 3(d)).

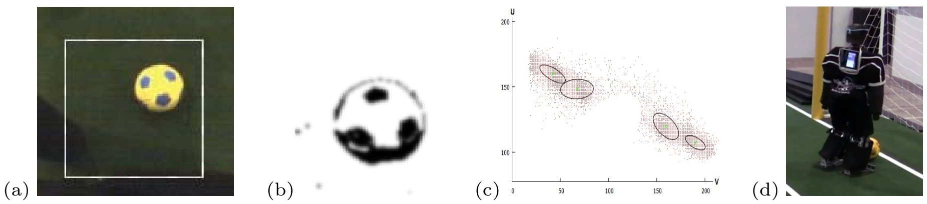


Fig. 3. Ball learning: (a) region of interest with unknown ball; (b) segmented pixels; (c) UV color histogram; (d) Dynaped completing the Dribbling Challenge

4 Behavior Control

We control our robots using a layered framework that supports a hierarchy of reactive behaviors [5]. When moving up the hierarchy, the update frequency of sensors, behaviors, and actuators decreases, while the abstraction level increases. Currently, our implementation consists of three layers. The lowest, fastest layer is responsible for generating motions, such as walking—including capture steps [6], kicking, and the goalie dive. At the next higher layer, we model the robot as a simple holonomic point mass that is controlled with the force field method to generate ball approach trajectories, ball dribbling sequences, and to implement obstacle avoidance. The topmost layer of our framework takes care of team behavior, game tactics and the implementation of the game states as commanded by the referee box. Please refer to [4] for further details.

Get-up Motion: We designed get-up motions for Copedo using a simple, linear interpolated keyframe technique [7]. The motions are executed open-loop after a prone or supine position has been detected. The challenge of performing a get-up motion with parallel kinematics, and thus missing a degree of freedom to pitch the foot, is that the robot is not able to explicitly place its foot flat on the ground. Using its arms, the robot pushes itself up from the floor while retracting its legs and rotating around the front or the back edge of the foot. When the center of mass crosses this edge, the robot will inevitably start tilting quickly towards the other side, pass the pose where the foot is flat on the ground with a relatively high rotational velocity, and is in danger of tipping over again. We found that holding the legs not fully retracted combined with some servo compliance results in a springy leg behavior that quickly dampens the back and forth rocking on the foot edges. Using this technique, active balancing is not required. Once the robot has reached a stable squatting position with the feet flat on the ground, it only has to stretch its legs to regain a standing posture and can continue walking. The get-up motions are illustrated in Figure 4.

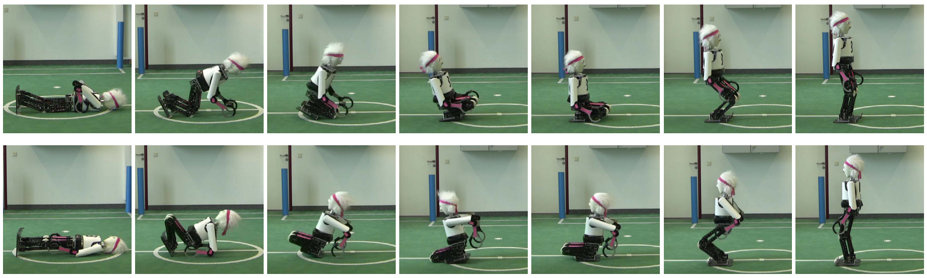


Fig. 4. Top row: Get-up motion from the prone posture. Bottom row: Get-up motion from the supine posture. In both motion sequences, the robot passively rocks back and forth on the foot edges from frames 3 to 5.

5 Conclusions

The 2012 competition showed notable progress in the development of the TeenSize class. Four participating teams were able to play dynamic soccer games and to complete several technical challenges. The highlights this year were the 3 vs. 3 demo game, where six goals were scored, and an exceptionally exciting final game between team NimbRo (Germany) and CIT Brains (Japan). The Japanese team was able to gain a lead in the first half with a surprisingly aggressive strategy. After a tie of 2:2 at half time, NimbRo played more offensively in the second half and achieved a final score of 6:3 for NimbRo.

In the future, the Humanoid League will continue to raise the bar. In the next year, equally colored goals will force the teams to deal with completely symmetric landmarks for localization and new technical challenges will require more sophisticated sensomotoric skills.

In order to make it easier for other teams to participate in the TeenSize class, our team NimbRo developed a modular open TeenSize robot platform, which will be released open-source and which will be made available to other teams for an affordable price [8].

Acknowledgement. This work is supported by Deutsche Forschungsgemeinschaft (German Research Foundation, DFG) under grant BE 2556/6.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2001)
2. Schmitz, A., Missura, M., Behnke, S.: Learning footstep prediction from motion capture. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS, vol. 6556, pp. 97–108. Springer, Heidelberg (2010)
3. Schulz, H., Behnke, S.: Utilizing the structure of field lines for efficient soccer robot localization. Advanced Robotics 26, 1603–1621 (2012)
4. Lee, D.D., Yi, S.-J., McGill, S., Zhang, Y., Behnke, S., Missura, M., Schulz, H., Hong, D., Han, J., Hopkins, M.: RoboCup 2011 Humanoid League winners. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 37–50. Springer, Heidelberg (2012)
5. Behnke, S., Stückler, J.: Hierarchical reactive control for humanoid soccer robots. International Journal of Humanoid Robots (IJHR) 5, 375–396 (2008)
6. Missura, M., Behnke, S.: Lateral capture steps for bipedal walking. In: Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids) (2011)
7. Stückler, J., Schwenk, J., Behnke, S.: Getting back on two feet: Reliable standing-up routines for a humanoid robot. In: Proceedings of The 9th International Conference on Intelligent Autonomous Systems (IAS-9) (2006)
8. Schreiber, M., Behnke, S.: Humanoid TeenSize Open Platform. In: Projects for Promoting RoboCup, 2012. Poster shown at 16th International RoboCup Symposium, Mexico City (2012)

NimbRo@Home: Winning Team of the RoboCup@Home Competition 2012

Jörg Stückler, Ishrat Badami, David Droeschel, Kathrin Gräve, Dirk Holz, Manus McElhone, Matthias Nieuwenhuisen, Michael Schreiber, Max Schwarz, and Sven Behnke

Rheinische Friedrich-Wilhelms-Universität Bonn
Computer Science Institute VI: Autonomous Intelligent Systems
Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
{stueckler,droeschel,graeve,holz,nieuwenhuisen,schreiber}@ais.uni-bonn.de,
{badami,mcelhone,schwarz,behnke}@cs.uni-bonn.de
http://www.NimbRo.net/@Home

Abstract. In this paper we describe details of our winning team NimbRo@Home at the RoboCup@Home competition 2012. This year we improved the gripper design of our robots and further advanced mobile manipulation capabilities such as object perception and manipulation planning. For human-robot interaction, we propose to complement face-to-face communication between user and robot with a remote user interface for handheld PCs. We report on the use of our approaches and the performance of our robots at RoboCup 2012.

1 Introduction

The RoboCup@Home league [16,17] was established in 2006 to foster the development and benchmarking of dexterous and versatile service robots that can operate safely in everyday scenarios. The robots have to show a wide variety of skills including object recognition and grasping, safe indoor navigation, and human-robot interaction. At RoboCup 2012, which took place in Mexico City, 21 international teams competed in the @Home league.

With our team NimbRo@Home we compete in the RoboCup@Home league since 2009. We improved the performance of our robots in the competitions, from third place in 2009 to second place in 2010 to winning in 2011 and 2012.

So far, we focused on hardware design and a system that balances indoor navigation, mobile manipulation, and human-robot interaction. In this year, we further advanced object recognition, modelling, and pose tracking capabilities. We also integrated motion planning for manipulation in complex scenes into the system. Last but not least, we developed a novel remote user interface on handheld computers that allows the user to control the autonomous capabilities of the robots on three levels.

In the following, we will give a short overview on the ruleset of the RoboCup-@Home competition 2012. We then detail our system with a focus on the novel components, compared to 2011. Finally, we will report on the performance of our robots at the 2012 competition.

2 Design of the RoboCup@Home Competition 2012

2.1 Overview

The competition consists of regular tests, i.e., tests with a predefined procedure, open demonstrations, and a technical challenge [5]. In two preliminary stages, the five best teams are selected for the final that is conducted as an open demonstration.

Regular tests cover basic mobile manipulation and human-robot interaction skills that all robots shall be able to demonstrate. The storylines of the regular tests are embedded in application scenarios. In these tests, the robots must act autonomously and fulfill the tasks within a limited amount of time. In the open demonstrations, the teams can choose their own task for the robot in order to demonstrate results of their own research. Finally, the technical challenge has been introduced to test a specific technical aspect in a benchmark. In this year, the robots had to demonstrate object recognition in cluttered scenes.

While the rules and the tests are announced several months prior to the competition, the details of the competition environment are not known to the participants in advance. During the first two days of the competition, the teams can map the competition arena, which resembles an apartment, and train object recognition on a set of 25 objects which are used as known objects with names throughout the recognition and manipulation tests. The arena is subject to minor and major changes during the competition and also contains previously unknown objects.

Performance is evaluated according to objective measures in the regular tests. Juries assess the quality of the open demonstrations based on score sheets. In the final, the jury consists of members of the league's executive committee and external jury members from science, industry, and media.

2.2 Tests and Skills

In Stage I, the teams compete in the tests *Robot Inspection and Poster Session*, *Follow Me*, *Clean Up*, *Who Is Who*, and the *Open Challenge*. During the *Robot Inspection and Poster Session*, the robots have to navigate to a registration desk, introduce themselves, and get inspected by the league's technical committee, while the team gives a poster presentation. In the *Follow Me* test, the robots must keep track of a previously unknown guide in an unknown (and crowded) environment. This year, the robots had to keep track of the guide despite a person blocking the line-of-sight. Then, they had to follow the guide into an elevator and demonstrate that they can find the guide after he/she went behind a crowd. *Clean Up* tests object recognition and grasping capabilities of the robots. They have to retrieve as many objects as possible within the time limit, recognize their identity, and bring them to their designated locations. The *Who Is Who* test is set in a butler scenario, where the robot first has to learn the identity of three persons. Then it has to take an order of drinks for each person, to grasp the correct drinks among others, and to deliver them to the correct person. The

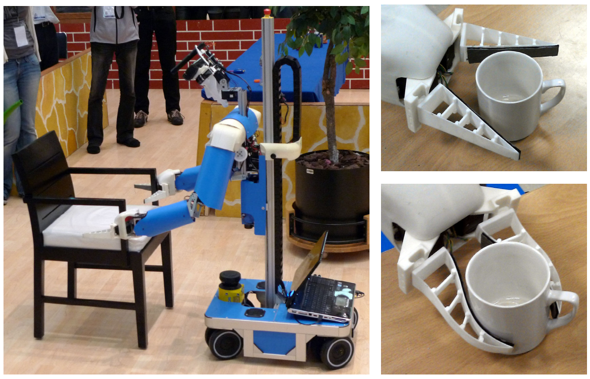


Fig. 1. The cognitive service robot *Cosero*. Left: Cosero moves a chair during the RoboCup@Home Final 2012 in Mexico City. Right: Cosero's grippers feature Festo FinRay fingers that adapt to the shape of objects.

Open Challenge is the open demonstration of Stage I. Teams can freely choose their demonstration in a 5 min slot.

Stage II consists of the *General Purpose Service Robot* test, the *Restaurant* test and the *Demo Challenge*. In the *General Purpose Service Robot* test, the robots must understand and act according to complex, incomplete or erroneous speech commands which are given by an unknown speaker. The commands can be composed from actions, objects, and locations of the regular Stage I tests. In the *Restaurant* test, the robots are deployed in a previously unknown real restaurant, where a guide makes them familiar with drink, food, and table locations. Afterwards, the guide gives an order to deliver three objects to specific locations. Finally, the *Demo Challenge* follows the theme "health care" and is the open demonstration of Stage II.

3 Hardware Design

We designed our service robots Cosero and Dynamaid [13] to cover a wide range of tasks in human indoor environments (see Fig. 1). They have been equipped with two anthropomorphic arms that provide human-like reach. Two torso joints extend the workspace of the arms: One joint turns the upper body around the vertical axis. A torso lift moves the whole upper body linearly up and down, allowing the robot to grasp objects from a wide range of heights—even from the floor. Its anthropomorphic upper body is mounted on a mobile base with narrow footprint and omnidirectional driving capabilities. By this, the robot can maneuver through narrow passages that are typically found in indoor environments, and it is not limited in its mobile manipulation capabilities by holonomic constraints.

In 2012, we improved Cosero's gripper design. We actuate two Festo FinGripper fingers using RX-64 Dynamixel actuators on two rotary joints (see Fig. 1).

When the gripper is closed on an object, the bionic fin ray structure of the fingers adapts its shape to the object surface. By this, the contact surface between fingers and object increases significantly, compared to a rigid mechanical structure. A thin layer of anti-skidding material on the fingers establishes a robust grip on objects.

For perceiving its environment, we equipped the robot with diverse sensors. Multiple 2D laser scanners on the ground, on top of the mobile base, and in the torso measure objects, persons, or obstacles for navigation purposes. The lasers in the torso can be rolled and pitched for 3D obstacle avoidance. We use a Microsoft Kinect RGB-D camera in the head to perceive tabletop objects and persons.

The human-like appearance of our robots also supports intuitive interaction of human users with the robot. For example, the robot appears to look at interaction partners while it tracks them with its head-mounted RGB-D camera. With its human-like upper body, it can perform a variety of gestures.

4 Mobile Manipulation

Some regular tests in the RoboCup competition involve object handling. Currently, objects are placed separated on horizontal surfaces such as tables and shelf layers. The robot needs to drive to object locations, to perceive the objects, and to grasp them.

We further advanced our mobile manipulation and perception pipelines. We developed means for object grasping in complex scenarios such as bin picking, and to track the pose of arbitrary objects in RGB-D images, for example, for moving chairs.

4.1 Motion Control

We implemented omnidirectional driving controllers for the mobile base of our robots [10]. The driving velocity can be set to arbitrary combinations of linear and rotational velocities. We control the 7-DoF arms using differential inverse kinematics with redundancy resolution. The arms also support compliant control in task-space [11].

4.2 Indoor Navigation

During the tests, the setup of the competition arena can be assumed static. We acquire 2D occupancy grid maps of unknown environments using GMapping [4]. We then employ state-of-the-art methods for localization and path planning in grid maps [10]. For obstacle-free driving along planned paths, we support the incorporation of all distance sensors of our robots. Point measurements are maintained in an ego-centric 3D map and projected into a 2D occupancy grid map for efficient local path planning.

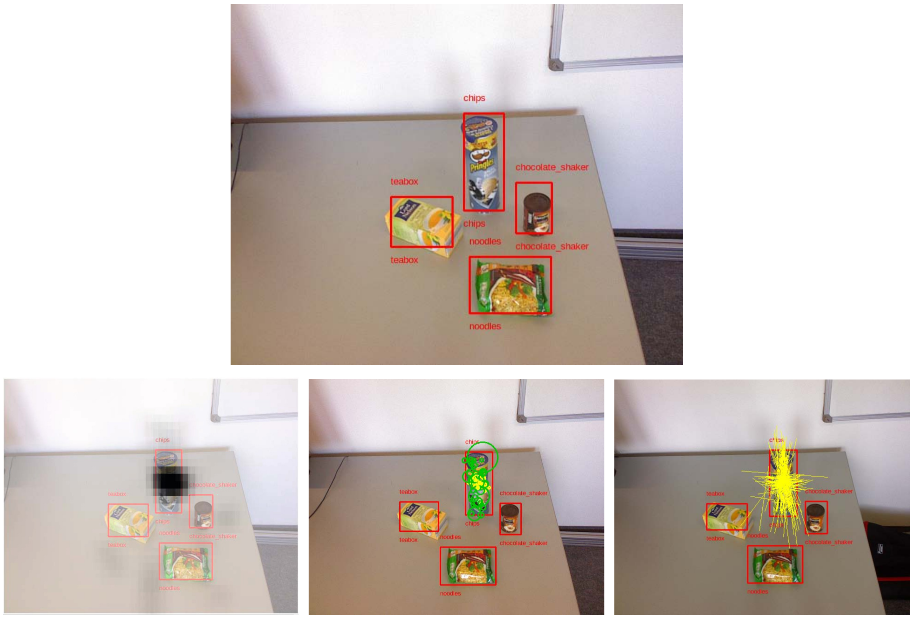


Fig. 2. Object recognition. Top: We recognize objects in RGB images and find location and size estimates. Bottom: Matched features vote for position in a 2D Hough space (left). From the features (middle, green dots) that consistently vote at a 2D location, we find a robust average of relative locations (middle, yellow dots) and principal directions (right, yellow lines).

4.3 Grasping Objects from Planar Surfaces

We developed efficient segmentation of RGB-D images to detect objects on planar surfaces [14]. On the raw measurements within the object segments, we plan top or side grasps on the objects. A collision-free grasp and reaching motion is then executed using parametrized motion primitives. Our method allows to grasp a large variety of typical household objects with cylindrical or box-like shapes. We implemented such highly efficient detection and motion planning to spend only little time for object manipulation during a test.

4.4 Object Recognition

Our robots recognize objects by matching SURF features [1] in RGB images to an object model database [10]. We improved our previous approach by enforcing consistency in the spatial relations between features (see Fig. 2).

In addition to the SURF feature descriptor, we store feature scale, feature orientation, relative location of the object center, and orientation and length of principal axes in the model. During recall, we efficiently match features between an image and the object database according to the descriptor using kd-trees.

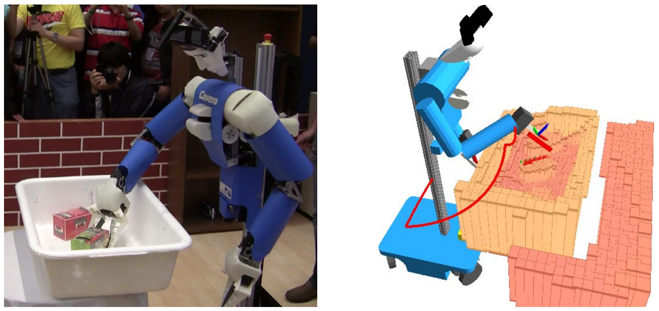


Fig. 3. Motion planning in a bin-picking scenario. We extend grasp planning on object segments with motion planning (reaching trajectory in red, pregrasp pose as larger coordinate frame) to grasp objects from a bin. For collision avoidance, we represent the scene in a multi-resolution height map. We decrease the resolution in the map with the distance to the object. This reduces planning time and models safety margins that increase with distance to the object.

Each matched feature then casts a vote to the relative location, orientation, and size of the object. We consider the relation between the feature scales and orientation of the features to achieve scale- and rotation-invariant voting.

With this object recognition method, our robots can recognize and localize objects in an RGB image as evaluated in this year's technical challenge. When unlabelled object detections are available through other modalities such as planar RGB-D segmentation (Sec. 4.3), we project the detections into the image and determine the identity of the object in these regions of interest.

4.5 Motion Planning in Complex Scenes

Our grasp planning module finds feasible, collision-free grasps at the object. The grasps are ranked according to a score which incorporates efficiency and stability criteria. The final step in our grasp and motion planning pipeline is now to identify the best-ranked grasp that is reachable from the current posture of the robot arm.

In complex scenes, we solve this by successively planning reaching motions for the found grasps ([9], see Fig. 3). We test the grasps in descending order of their score. For motion planning, we employ LBKPIECE [15].

To speed up the process of evaluating collision-free grasp postures and planning trajectories, we employ a multiresolution height map that extends our prior work on multiresolution path planning [2]. Our height map is represented by multiple grids that have different resolutions. Each grid has $M \times M$ cells containing the maximum height value observed in the covered area (Fig. 3). Recursively, grids with quarter the cell area of their parent are embedded into each other, until the minimal cell size is reached. With this approach, we can cover the same area as a uniform $N \times N$ grid of the minimal cell size with only $\log_2((N/M)+1)M^2$ cells. Planning in the vicinity of the object needs a more exact environment

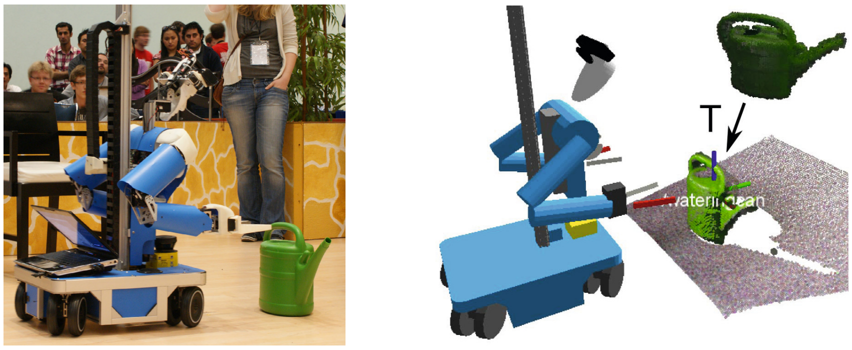


Fig. 4. Object pose tracking. We train multi-view 3D models of objects using multi-resolution surfel maps. We estimate the pose of objects in RGB-D images through real-time registration towards the model. We apply object tracking, for instance, to track the model (upper right) of a watering can for approaching and grasping it.

representation as planning farther away from it. This is accomplished by centering the collision map at the object. This approach also leads to implicitly larger safety margins with increasing distance to the object.

4.6 Object Modelling and Pose Tracking

Many object handling tasks assume object knowledge that cannot be deduced from a single view alone. If an object model is available, the robot can infer valid grasping points or use the model to detect objects and to keep track of them. For example, to implement the handling of a watering can or the moving of a chair with our robot, we teach-in grasping and motion strategies. These grasps and motions are specified in the local reference frame of an object model. To be able to reproduce the motions, the robot needs to perceive the pose of the object. While the robot moves, we register RGB-D images to the model at high frame rates to keep track of the object. This way, the robot does not require a precise motion model.

In our approach, we train a multi-resolution surfel map of the object ([12], see Fig. 4). The map is represented in an octree where each node stores a normal distribution of the volume it represents. In addition to shape information, we also model the color distribution in each node.

Our object modelling and tracking approach is based on an efficient registration method. We build maps from RGB-D images and register these representations with an efficient multi-resolution strategy. We associate each node in one map to its corresponding node in the other map using fast nearest-neighbor look-ups. We optimize the matching likelihood for the pose estimate iteratively to find the most likely pose.

We acquire object models from multiple views in a view-based SLAM approach. During SLAM, we generate a set of key frames that we register to each other. We optimize pose estimates of the key frames to best fit the spatial relations that we obtain through registration. While the camera is moving, we register the current RGB-D image to the closest key frame. Each time the translational or angular distance is above a threshold, we include the current frame as a new key frame into the map. For SLAM graph optimization, we employ the g2o framework [6]. Finally, we merge all key frames based on their pose estimate in a multi-view map.

Once we have a model, we can register RGB-D camera images against it to retrieve the pose of the object. We initialize the pose of the tracker to a rough estimate using our planar segmentation approach.

5 Human-Robot Interaction

5.1 Intuitive Direct Human-Robot Interaction

Domestic service robots need intuitive user interfaces so that laymen can easily control the robots or understand their actions and intentions. Speech is the primary modality of humans for communicating complex statements in direct interaction. For speech synthesis and recognition, we use the commercial system from Loquendo [7]. Loquendo's text-to-speech system supports natural and colorful intonation, pitch and speed modulation, and special human sounds like laughing or coughing.

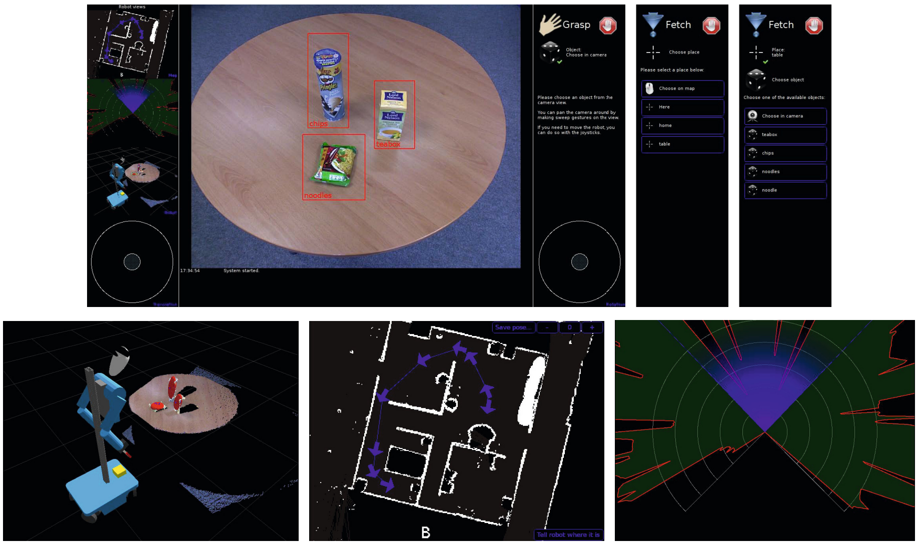
We also implemented pointing gesture synthesis as a non-verbal communication cue. Cosero performs gestures like pointing or waving. Pointing gestures are useful to direct a user's attention to locations and objects. The robots also interpret gestures such as waving or pointing [3].

5.2 Convenient Remote User Interfaces

We develop handheld user interfaces to complement natural face-to-face interaction modalities [8]. Since the handheld devices display the capabilities and perceptions of the robot, they improve common ground between the user and the robot (see Fig. 5). They also extend the usability of the robot, since users can take over direct control for skills or tasks that are not yet implemented with autonomous behavior. Finally, such a user interface enables remote interaction with the robot, which is especially useful for immobile persons.

The user interface supports remote control of the robot on three levels of autonomy. The user can directly control the drive and the gaze using joystick-like control UIs or touch gestures. The user interface also provides selection UIs for autonomous skills such as grasping objects or driving to locations. Finally, the user can configure high-level tasks such as fetch and delivery of specific objects.

The user interface is split into a main interactive view in its center and two configuration columns on the left and right side (see Fig. 5, top). In the left column, further scaled-down views are displayed that can be dragged into the main view. In this case, the dragged view switches positions with the current main view. One view displays live RGB-D camera images with object perception



Please choose an object from the camera view.

You can pan the camera around by making sweep gestures on the view.

If you need to move the robot, you can do so with the joysticks.

Fig. 5. Handheld User Interface. The user interface provides controls on three levels of autonomy. Top: Complete GUI with a view selection column on the left, a main view in the center, and a configuration column on the right. We placed two joystick control UIs on the lower and right corners for controlling motions of the robot with the thumbs. Lower right: 3D external view generated with Rviz. Lower middle: The navigation view displays the map, the estimated location, and the current path of the robot. Lower right: The sensor view displays laser scans and the field-of-view of the RGB-D camera in the robot's head.

overlays (Fig. 5, top). The user may change the gaze of the robot by sweep gestures, or select objects to grasp. A further view visualizes laser range scans and the field-of-view of the RGB-D camera (Fig. 5, bottom right). The navigation view shows the occupancy map of the environment and the pose of the robot (Fig. 5, bottom center). The user can set current pose and goal pose. While the robot navigates, the view shows the current path. Finally, we also render a 3D external view (Fig. 5, bottom left).

On the right (Fig. 5, top), high-level tasks such as fetch and delivery can be configured. For fetching an object, for instance, the user either selects a specific object from a list, or chooses a detected object in the current sensor view.

6 Competition Results at RoboCup 2012

With our robot system, we achieved scores among the top rankings in almost every test of the competition[1]. In Stage I, Cosero and Dynamaid registered for the competition in the *Robot Inspection and Poster Session*. In the new *Follow Me* test, Cosero learned the face of the guide and was not disturbed later by

[1] A video can be found at `http://www.NimbRo.net/@Home`

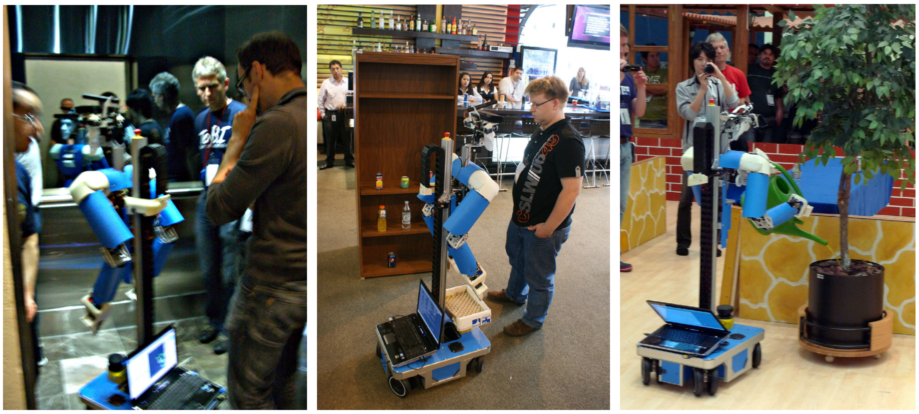


Fig. 6. Left: Cosero follows a guide into an elevator during the *Follow Me* test. Middle: In the *Restaurant* test, a guide shows Cosero drink and food locations in a real and previously unknown restaurant. Right: Cosero waters a plant in the final.

another person blocking the line-of-sight. It followed the guide into the elevator (see Fig. 6) and left it on another floor. Unfortunately, it falsely detected a crowd of people and could not finish the test. In *Who Is Who*, Cosero learned the faces of three persons, took an order, fetched three drinks in a tray and each of its arms, and successfully delivered two of them within the time limit. In the *Clean Up* test, our robot Cosero had to find objects that were distributed in the apartment, recognize them, and bring them to their place. Our robot detected three objects, from which two were correctly recognized as unknown objects. It grasped all three objects and deposited them in the trash bin. In the *Open Challenge*, we showed a "housekeeping" scenario. Cosero demonstrated that it could recognize a waving person. It took over an empty cup from this person and threw it into the trash bin. Afterwards, it approached a watering can and watered a plant. After finishing all tests of Stage I, our team lead the competition with 5,071 points, followed by WrightEagle (China) 3,398 points and ToBi (Germany) 2,627 points.

In the second stage, Cosero recognized speech commands from two out of three categories in the *General Purpose Service Robot* test. It recognized a complex speech command consisting of three actions. While it successfully performed the first part of the task, it failed to recognize the object in a shelf. It also understood a speech command with incomplete information and posed adequate questions to retrieve missing information. The third speech command was not covered by the grammar and, hence, could not be understood. Overall, Cosero achieved the most points in this test. In the *Demo Challenge* with the theme "health care", an immobile person used a handheld PC to teleoperate the robot. The person sent the robot to fetch a drink. The robot recognized that the requested drink was not available and the user selected another drink in the transmitted camera image. After the robot delivered the drink, it recognized a pointing gesture and navigated to the referenced object in order to pick it up from the ground. In the

Restaurant test, our robot Cosero was guided through a previously unknown bar (see Fig. 6). The guide showed the robots where the shelves with items and the individual tables were. Our robot built a map of this environment and took an order. Afterwards, it navigated to the food shelf to search for requested snacks. The dim lighting conditions in the restaurant, however, prevented Cosero from recognizing the objects. After both stages, we accumulated 6,938 points and entered the final with a clear advantage towards WrightEagle (China, 4,677 points) and eR@sers (Japan, 3,547 points).

In the final, our robot Cosero demonstrated the approaching, bi-manual grasping, and moving of a chair to a target pose. It also approached and grasped a watering can with both hands and watered a plant (see Fig. 6). After this demonstration, our robot Dynamaid fetched a drink and delivered it to the jury. In the meantime, Cosero approached a transport box, from which it grasped an object using grasp planning. This demonstration convinced the high-profile jury, which awarded the highest number of points in all categories (league-internal jury: scientific contribution, relevance, presentation and performance; external jury: originality, usability, difficulty and success). Together with the lead after Stage II, our team received 100 normalized points, followed by eR@sers (Japan, 74 points) and ToBi (Germany, 64 points).

7 Conclusion

In this paper, we presented the contributions of our winning team NimbRo to the RoboCup@Home competition 2012 in Mexico City. Since the 2011 competition, we improved object recognition, developed model learning and tracking, and implemented motion planning to further advance the mobile manipulation capabilities of our robots. We also developed a novel remote user interface on handhelds to complement natural face-to-face interaction through speech and gestures.

Our robots scored in all the tests of the competition and gained a clear advantage in the preliminary stages. In the final, our robots convinced the high profile jury and won the competition.

In future work, we will further develop robust object recognition in difficult lighting conditions. More fluent and flexible speech and non-verbal cues will improve the naturalness of human-robot interaction. Finally, we also plan to investigate tool-use and learning for object handling.

Acknowledgments. This project has been partially supported by the FP7 ICT-2007.2.2 project ECHORD (grant agreement 231143) experiment ActReMa.

References

1. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)

2. Behnke, S.: Local multiresolution path planning. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 332–343. Springer, Heidelberg (2004)
3. Droeschel, D., Stückler, J., Holz, D., Behnke, S.: Towards joint attention for a domestic service robot – Person awareness and gesture recognition using time-of-flight cameras. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2011)
4. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with Rao-Blackwellized particle filters. IEEE Trans. on Robotics 23(1) (2007)
5. Holz, D., Mahmoudi, F., Rascon, C., Wachsmuth, S., Sugiura, K., Iocchi, L., del Solar, J.R., van der Zant, T.: RoboCup@Home: Rules & regulations (2012), `http://purl.org/holz/rulebook.pdf`
6. Kuemmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: A general framework for graph optimization. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2011)
7. Loquendo S.p.A. Vocal technology and services (2007), `http://www.loquendo.com`
8. Muszynski, S., Stückler, J., Behnke, S.: Adjustable autonomy for mobile teleoperation of personal service robots. In: Proc. of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN) (2012)
9. Nieuwenhuisen, M., Stückler, J., Berner, A., Klein, R., Behnke, S.: Shape-primitive based object recognition and grasping. In: Proc. of the 7th German Conference on Robotics (ROBOTIK) (2012)
10. Stückler, J., Behnke, S.: Integrating indoor mobility, object manipulation, and intuitive interaction for domestic service tasks. In: Proc. of the IEEE Int. Conf. on Humanoid Robots (Humanoids) (2009)
11. Stückler, J., Behnke, S.: Compliant task-space control with back-drivable servo actuators. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 78–89. Springer, Heidelberg (2012)
12. Stückler, J., Behnke, S.: Model learning and real-time tracking using multi-resolution surfel maps. In: Proc. of the AAAI Conference on Artificial Intelligence (AAAI 2012) (2012)
13. Stückler, J., Droeschel, D., Gräve, K., Holz, D., Kläß, J., Schreiber, M., Steffens, R., Behnke, S.: Towards robust mobility, flexible object manipulation, and intuitive multimodal interaction for domestic service robots. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 51–62. Springer, Heidelberg (2012)
14. Stückler, J., Steffens, R., Holz, D., Behnke, S.: Efficient 3D object perception and grasp planning for mobile manipulation in domestic environments. In: Robotics and Autonomous Systems (2012)
15. Şucan, I.A., Kavraki, L.E.: Kinodynamic motion planning by interior-exterior cell exploration. In: Chirikjian, G.S., Choset, H., Morales, M., Murphey, T. (eds.) Algorithmic Foundation of Robotics VIII. STAR, vol. 57, pp. 449–464. Springer, Heidelberg (2009)
16. van der Zant, T., Wisspeintner, T.: A proposal for a new league where RoboCup goes real world. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 166–172. Springer, Heidelberg (2006)
17. Wisspeintner, T., van der Zant, T., Iocchi, L., Schiffer, S.: RoboCup@Home: Scientific competition and benchmarking for domestic service robots. Interaction Studies 10(3), 393–428 (2009)

How Much Worth Is Coordination of Mobile Robots for Exploration in Search and Rescue?

Francesco Amigoni[1], Nicola Basilico[2], and Alberto Quattrini Li[1]

[1] Politecnico di Milano, Milano, Italy
francesco.amigoni@polimi.it, quattrini.li@elet.polimi.it
[2] University of California, Merced, USA
nbasilico@ucmerced.edu

Abstract. Exploration of unknown environments is an enabling task for several applications, including map building and search and rescue. It is widely recognized that several benefits can be derived from deploying multiple mobile robots in exploration, including increased robustness and efficiency. Two main issues of multirobot exploration are the *exploration strategy* employed to select the most convenient observation locations the robots should reach in a partially known environment and the *coordination method* employed to manage the interferences between the actions performed by robots. From the literature, it is difficult to assess the relative effects of these two issues on the system performance. In this paper, we contribute to filling this gap by studying a search and rescue setting in which different coordination methods and exploration strategies are implemented and their contributions to an efficient exploration of indoor environments are comparatively evaluated. Although preliminary, our experimental data lead to the following results: the role of exploration strategies dominates that of coordination methods in determining the performance of an exploring multirobot system in a highly structured indoor environment, while the situation is reversed in a less structured indoor environment.

Keywords: search and rescue, exploration, coordination, multirobot.

1 Introduction

Robotic exploration of unknown environments is fundamental for several real-world applications, including map building and search and rescue. It is widely recognized that several benefits can be derived from deploying multiple mobile robots in exploration, ranging from an increased robustness of the whole system to a more efficient exploration [1–3]. Two important issues of multirobot exploration are exploration strategies and coordination methods. An *exploration strategy* is employed to select the most convenient observation locations the robots should reach in a partially known environment [4]; in short an exploration strategy is used to answer the question "where to go next?". A *coordination method* is employed to manage the interferences between the actions performed by robots [5]; in the context of exploration, a coordination method is used to allocate tasks to robots and basically to answer the question "who goes where?". Prior work evaluates these two issues mostly in a separated way, making it difficult to assess their relative effects on exploration.

In this paper, we contribute to fill this gap by comparatively evaluating some coordination methods and exploration strategies in a search and rescue setting according to their contribution to an efficient exploration of indoor environments. We selected the search and rescue application because there is an international competition, namely the RoboCup Rescue Virtual Robot Competition[1], that provides a simulated common ground (e.g., metrics and software tools) for assessing the performance of exploring multirobot systems, enabling comparison and reproduction of results.

The general setting we consider is the following. A team of robots has to search an initially unknown environment for victims. Since no *a priori* knowledge about the possible locations of the victims is assumed to be available, we can reduce the problem of maximizing the number of victims found in a given time interval to the problem of maximizing the amount of area covered by robots' sensors in the same time interval. Broadly speaking, the robots operate according to the following steps: (a) they perceive the surrounding environment, (b) they integrate the perceived data within a map representing the environment known so far, (c) they decide where to go next and who goes where, and (d) they go to their destination locations and start again from (a). In our experiments, we employ a publicly available simulator [6] and controller [7]. In this way, we can focus on the exploration strategies and coordination methods (step (c)) exploiting an already tested framework for steps (a), (b), and (d).

The original contribution of this paper is not in proposing new exploration strategies or coordination methods, but in taking some initial steps in shedding light on their relative impact on the performance of multirobot systems employed in search and rescue applications. We contribute to answer the following question: With limited computing or time resources, should developers spend more efforts on developing an effective exploration strategy or coordination method?

2 Coordinated Multirobot Exploration

Robotic exploration can be defined as a process that discovers unknown features in environments by means of mobile robots. Coordinated multirobot exploration has been mainly studied for map building [8, 9] and for search and rescue [10]. Previous works on coordinated multirobot exploration have focused in a rather separated way on evaluation of either coordination methods or exploration strategies.

Exploration strategies are used to select locations that autonomous robots should reach in order to discover the physical structure of environments that are initially unknown. In the following, we survey a representative sample of the several exploration strategies that have been proposed in literature.

[1] http://www.robocuprescue.org/virtualsim.html

Unsurprisingly, most of the work on exploration strategies for discovering the physical structure of environments has been done for map building. The mainstream approach models exploration as an incremental Next Best View (NBV) process, i.e., a repeated greedy selection of the next best observation location. Usually, at each step, an NBV system considers a number of candidate locations on the frontier between the known free space and the unexplored part of the environment (in such a way they are reachable from the current position of the robot) and selects the best one [11]. The most important feature of an exploration strategy is the *utility function* it uses to evaluate candidate locations in order to select the best one.

In evaluating candidate locations, different criteria can be used. A simple one is the distance from the current position of the robot [12], according to which the best observation location is the nearest one. Most works combine different criteria in more complex utility functions. For example, in [13] the cost of reaching a candidate location is linearly combined with its benefit. Another example of combination of different criteria is [14], in which the distance of a candidate location from the robot and the expected information gain of the candidate location are combined in an exponential function. In [15], a technique based on relative entropy is used to combine traveling cost and expected information gain. In [16], several criteria (such as uncertainty in landmark recognition and number of visible features) are combined in a multiplicative function.

The above strategies aggregate different criteria in utility functions that are defined *ad hoc* and are strongly dependent on the criteria they combine. In [17], the authors proposed a more theoretically-grounded approach based on multi-objective optimization, in which the best candidate location is selected on the Pareto frontier. Following the same theoretically-grounded approach, decision theoretical tools have been applied to the definition of exploration strategies [18]. More details on this approach will be illustrated in Section 3.2.

Compared with exploration strategies for map building, relatively few exploration strategies for autonomous search and rescue have been proposed. A work that explicitly addressed this problem is [7], which proposes to combine some criteria in an *ad hoc* utility function that will be described in Section 3.2. In [10], traveling cost to reach a location is used as the main criterion for evaluating candidate locations, while the utility of the locations (calculated according to the proximity of other robots) is used as a tie-breaker. The exploration strategy for search and rescue of [19] uses a formalism based on Petri nets for exploiting *a priori* information about the victims' distribution to improve the search.

In this paper, we evaluate, relatively to some coordination methods, the exploration strategies proposed in [18] and [7], as representative samples of theoretically-grounded and *ad hoc* exploration strategies, respectively.

Coordination methods are used to manage the interactions between multiple robots. Here we are interested in coordination methods that are used to allocate locations to the robots during exploration. One of the earliest works in the field of multirobot exploration is by Yamauchi [12], in which robots navigate, in an uncoordinated way, to the closest accessible unvisited frontiers and integrate their local maps in a global map of the environment.

A series of works [1, 2] (and, partially, [20]) propose an interesting approach in which the coordination method is embedded within the exploration strategy. In particular, the utility value of a candidate location is reduced according to the number of robots that can view it. In this way, robots are pushed to select different locations to reach. Experimental results show that this coordinated behavior has better performance than uncoordinated behavior (in which different robots can select the same location to reach) and slightly worse performance than a method that finds the optimal allocation of candidate locations to robots.

Coordination methods based on market mechanisms have been extensively studied. For example, in [21] coordination of mobile robots is performed by a central executive that, beyond collecting local maps and combining them into a single global map, manages an auction by asking bids to the robots and assigning tasks (i.e., locations to reach) according to the received bids. Bids contain information about expected utility for pairs robot-location; utility are calculated as the expected information gain at the location minus the cost for reaching it. A similar coordination method is presented in [22] in connection with three techniques for generating the locations that the robots should reach (a random technique, a closest-point greedy technique, and a quadtree-based technique). These points are evaluated using an utility function similar to that used in [21]. Experimental results show that the auction-based coordination method performs better with a random and a quadtree-based generation of locations, while (as expected) outperforming the uncoordinated methods. Qualitatively similar findings are reported also in [23], which proposes an auction-based coordination method not only for task assignment, but also for coalition formation.

In this paper, we evaluate, relatively to some exploration strategies, some variants of the coordination method employed in [7], which produces the same allocation of the market-based coordination method of [21]. Our results complement those of [22], by considering more complex ways for generating the locations allocated to robots.

3 The Search and Rescue Setting

In this section, we describe the search and rescue setting in which we investigated the relative impact of exploration strategies and coordination methods on performance of exploring multirobot systems. In our setting, the goal is to explore an initially unknown indoor environment for finding the largest number of human victims within a given time. Assuming no *a priori* knowledge about the possible locations of the victims, the problem of maximizing the number of victims found in a given time interval is equivalent to the problem of maximizing the amount of area covered by robots' sensors in the same interval. We consider a time interval of 15 minutes. We first describe the adopted simulation environment and robot controller. Then, we describe the exploration strategies and the coordination methods we consider.

3.1 The Simulation Environment and the Robot Controller

In order to perform repeated tests under controlled conditions, we use a robot simulator. We selected USARSim [6] because it is a high fidelity 3D robot simulator and it is employed in the RoboCup Rescue Virtual Robot Competition.

From an analysis based on availability of code and performance obtained in the RoboCup Rescue Virtual Robot Competition, we selected the controller developed by the Amsterdam and Oxford Universities (Amsterdam Oxford Joint Rescue Forces, AOJRF[2]) for the 2009 competition [24]. The main reason for using an existing controller is that we can focus only on the exploration strategies and on the coordination methods, exploiting existing and tested methods for navigation, localization, and mapping. The controller manages a team of robots. The robotic platform used is a Pioneer P2AT equipped with range scanner sensors and sensors able to detect human victims. The map of the environment is maintained by a base station, whose position is fixed in the environment, and to which robots periodically send data. The map is two-dimensional and represented by three occupancy grids. The first one is obtained with a small-range (3 meters) scanner and constitutes the *safe area*, i.e., the area where the robots can safely move. The second one is obtained from maximum-range scans (20 meters) and constitutes the *free area*, i.e., the area which is believed to be free but not yet safe. Moreover, a representation of the *clear area* is maintained as a subset of the safe area that has been checked for the presence of victims (this task is accomplished with simulated sensors for victim detection). Given a map represented as above, a set of (connected) boundaries between safe and free regions are extracted. The center point of the free area beyond each boundary is considered as a *candidate location* to reach. The utility $u(p, r)$ of a candidate location p for a robot r is evaluated as discussed in the next section.

3.2 Exploration Strategies

As discussed in Section 2, exploration strategies differ in the utility functions they use to evaluate and select the candidate locations. The following criteria are combined in our utility functions:

- $A(p)$ is the amount of free area beyond the frontier of p computed according to the free area occupancy grid;
- $P(p)$ is the probability that a robot, once reached p, will be able to transmit information (e.g., the perceived data or the locations of victims) to the base station (whose position in the environment is known), this criterion depends on the distance between p and the base station;
- $d(p, r)$ is the distance between p and current position of robot r, this criterion can be calculated with two different methods: $d_{EU}()$, using an approximate method that calculates the Euclidean distance, and $d_{PP}()$, using a path planner procedure that returns the exact value of the distance (if no safe path completely contained in the explored area can be found, then $d_{PP}() = \infty$); obviously, calculating $d_{PP}()$ requires more time than calculating $d_{EU}()$;

[2] `http://www.jointrescueforces.eu/`

– $b(r)$ is the battery level of robot r (from 0, full, to 1, empty); the larger its value, the smaller the amount of residual energy in the battery.

Given these criteria, we define two exploration strategies. The first one is a slight variation of the strategy proposed in [7] and is called *AOJRF strategy.* It integrates the above criteria in an *ad hoc* utility function:

$$u(p,r) = \frac{A(p)P(p)}{d(p,r)^{b(r)}}. \tag{1}$$

The second exploration strategy is called *MCDM strategy* and combines the criteria of the set $N = \{A, P, d, b\}$ using the Multi-Criteria Decision Making (MCDM) approach. Refer to [18] for a complete description; here we just sketch how the approach works. We call $u_j(p,r)$, with $j \in N$, the utility value for candidate location p and robot r according to criterion j. To apply MCDM, utilities have to be normalized to a common scale $I = [0,1]$. We use a linear relative normalization. For example, given a robot r, the utility of a candidate p related to the distance $d()$ is normalized using $u_d(p,r) = 1-(d(p,r)-\min_{q\in C} d(q,r))/(\max_{q\in C} d(q,r)-\min_{q\in C} d(q,r))$, where C is the set of candidate locations. Note that the larger $u_j(p,r)$, the better the pair p and r.

Basically, the MCDM strategy replaces function (1) with the following function:

$$u(p,r) = \sum_{j=1}^{4} (u_{(j)}(p,r) - u_{(j-1)}(p,r))\mu(A_{(j)}), \tag{2}$$

where $\mu : \mathcal{P}(N) \to [0,1]$ ($\mathcal{P}(N)$ is the power set of set N) is such that $\mu(\{\emptyset\}) = 0$, $\mu(N) = 1$, and, if $A \subset B \subset N$, then $\mu(A) \leq \mu(B)$. That is, μ is a normalized *fuzzy measure* on the set of criteria N that will be used to associate a weight to each group of criteria. $u_{(j)}$, with $(j) \in N$, indicates the j-th criterion according to an increasing ordering with respect to utilities, i.e., after that criteria have been ordered to have, for candidate p and robot r, $u_{(1)}(p,r) \leq \ldots \leq u_{(n)}(p,r) \leq 1$. It is assumed that $u_{(0)}(p,r) = 0$. Finally, the set $A_{(j)}$ is defined as $A_{(j)} = \{i \in N | u_{(j)}(p,r) \leq u_i(p,r) \leq u_{(n)}(p,r)\}$.

Using (2) is a more principled way than (1) to compute utilities, because it allows to consider criteria's importance and their mutual dependency relations. Criteria belonging to a group $G \subseteq N$ are said to be redundant if $\mu(G) < \sum_{i\in G} \mu(i)$, synergic if $\mu(G) > \sum_{i\in G} \mu(i)$, and independent otherwise.

We use the weights reported in the following table, which have been manually set in order to obtain good performance (according to [18]).

criteria	A	d	P	b	A,d	A,P	A,b	d,P	d,b	P,b	A,d,P	A,d,b	A,P,b	d,P,b
$\mu()$	0.4	0.3	0.05	0.25	0.75	0.55	0.55	0.4	0.32	0.28	0.9	0.8	0.85	0.4

The two exploration strategies have been selected because they are representative of the two main classes of strategies that have been proposed for exploration of unknown environments (see Section 2). In particular, the AOJRF strategy represents *ad hoc* strategies, while the MDCM strategy represents more theoretically-grounded strategies.

3.3 Coordination Methods

While exploration strategies evaluate the goodness of a candidate location p for a robot r, coordination methods are used to assign candidate locations to robots. We define three coordination methods for allocating candidate locations to robots. They start from a set of candidate locations (generated as discussed in Section 3.1) and a set of robots, and their goal is to assign a location to each robot.

The first coordination method, which is executed by each robot independently, knowing (from the base station) the current map and the positions of the other robots is derived directly from [7]:

1. compute the global utility $u(p,r)$ of allocating each candidate p to each robot r (using (1) or (2)) where $d(p,r)$ is calculated using the Euclidean distance $d_{EU}()$ (namely using an underestimate of the real distance),
2. find the pair (p^*, r^*) such that the previously computed utility is maximum, $(p^*, r^*) = \arg\max_{p,r} u(p,r)$,
3. re-compute the distance between p^* and r^* using $d_{PP}()$ with the path planner (namely considering the real distance) and update the utility of (p^*, r^*) using such exact value instead of the Euclidean distance,
4. if (p^*, r^*) is still the best allocation, then allocate location p^* to robot r^*, otherwise go to Step 2,
5. eliminate robot r^* and candidate p^* and go to Step 2.

This first coordination method is called *AOJRF original coordination*. The reason behind the utility update of Step 3 is that computing $d_{PP}()$ requires a considerable amount of time. Calculating it for all the candidate locations and all robots would be not affordable in the rescue competition, since a maximum exploration time is enforced. Although in pathological cases all pairs (p,r) could be re-evaluated, in practice this is done only for few of them. Note that, being $d_{EU}()$ an underestimate of the real distance and being (1) and (2) monotonically decreasing with $d()$, the method is guaranteed to select the best pair (p^*, r^*) according to $u()$ calculated with $d_{PP}()$.

The AOJRF original coordination method produces the same results of the market-based mechanism proposed in [21] and is applied considering (1) or (2) to calculate utilities for bids. Both methods first select the pair (p^*, r^*) with the largest utility $u()$, then, among the pairs left after elimination of those involving p^* and r^*, they select the pair (p^{**}, r^{**}) with the largest utility, and so on.

The second coordination method, called *AOJRF simplified coordination*, is similar to the previous one, but does not re-compute the distance in the Step 3. It selects the best pair (p^*, r^*) only on the basis of the Euclidean distance.

In the third coordination method, called *no coordination*, each robot selfishly selects its best candidate location, without considering the presence of other robots. This means that Steps 1-4 are performed only for one robot r^* (the robot that is running the method) and that Step 5 is skipped. Note, however, that Step 3 is executed and distance re-computed.

The three coordination methods are in decreasing order of "optimality" in allocating locations to the robots, with the AOJRF original coordination method producing the best allocation and the no coordination method the worst. The last two methods can end up with sub-optimal allocations in which a robot is assigned a location that is supposed to be close but is actually far (AOJRF simplified coordination method) or in which two robots are assigned the same location (no coordination method).

4 Experimental Results

We consider teams of two and three robots (plus the base station) deployed in the "DM-compWorldDay4b_250" and "DM-VMAC1" environments, called *office* and *open* environments, respectively (see Fig. 1). Both the environments are indoor with the office environment (about 800 m^2) presenting an intricate cluttered structure and the open environment (about 1300 m^2) presenting more open spaces. We define a configuration as an environment, a number of robots, an exploration strategy, and a coordination method. For each configuration, we execute 5 runs (with randomly selected starting locations for the mobile robots such that they are separated by about 20 meters) of 15 minutes each. We assess performance by measuring the amount of free, safe, and clear area every 30 seconds of the exploration. Due to space limitations, we report only data on safe area at the end of runs (free area is less significant and clear area is similar to the safe area). Of course, the larger the mapped safe area within 15 minutes, the better the performance. Under the assumption that victims are uniformly spread in the environment, this metric is basically equivalent to the metric that counts the number of victims found. Experiments have been run in real-time as in the competition, to realistically account for time spent in movements and in computation.

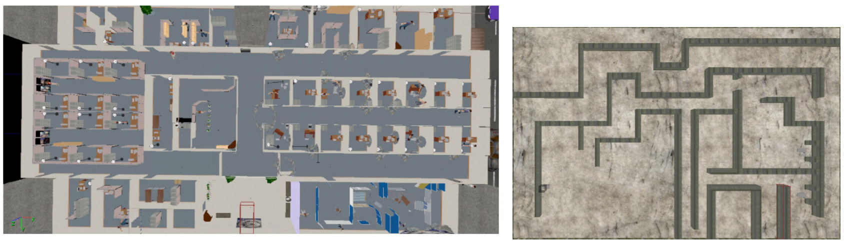


Fig. 1. The office environment (left) and the open environment (right)

To have a base line in comparing the results, we consider a *random coordination* method that randomly assigns robots to candidate locations, without evaluating them. We expect this random method to perform worse than other combinations of exploration strategies and coordination methods.

Tab. 1(a) shows results for the office environment. With all the three coordination methods, the MCDM strategy seems to behave better than the AOJRF strategy, although differences are not statistically significant, according to an ANOVA analysis with a threshold for significance p-value < 0.05 [25]. The difference between the safe area mapped at the end of the 15 minutes is more evident with the AOJRF original coordination method. Conversely, the difference between the two exploration strategies is less evident with the AOJRF simplified coordination method. These results can be explained by saying that MCDM better exploits the more precise information used with the AOJRF original coordination method (a precise distance value obtained with path planning procedures instead of an approximate Euclidean distance value). Multirobot exploration introduces some benefits, as shown by the configurations with three robots that consistently outperform those with two robots (consider that a single robot maps approximately 250 m^2 with the MCDM strategy and 230 m^2 with the AOJRF strategy). Finally, the random method has, as expected, the worst performance (we tested it only with two robots).

Table 1. Average safe area (and standard deviation) mapped after 15 minutes (units are m^2)

(a) office environment

	2 robots		**3 robots**	
	AOJRF strategy	MCDM strategy	AOJRF strategy	MCDM strategy
AOJRF original coordination	299.77(53.60)	341.95(12.54)	341.58(98.62)	387.41(66.67)
AOJRF simplified coordination	257.53(54.65)	262.43(15.62)	320.40(63.71)	325.14(42.21)
no coordination	306.36(65.91)	330.27(46.38)	332.58(42.03)	374.28(40.31)
random	211.68(18.86)	211.68(18.86)		

(b) open environment

	2 robots		**3 robots**	
	AOJRF strategy	MCDM strategy	AOJRF strategy	MCDM strategy
AOJRF original coordination	430.18(78.86)	498.45(51.12)	483.46(130.14)	511.83(118.35)
AOJRF simplified coordination	586.77(72.16)	678.27(48.77)	673.48.77(85.61)	690.16(36.69)
no coordination	356.92(65.97)	425.05(99.01)	458.55(80.30)	498.08(81.03)
random	472.71(115.48)	472.71(115.48)		

The performance of the AOJRF original coordination method and that of the method without coordination are very similar and better than that of the AOJRF simplified coordination method. The difference between the safe area mapped at 15 minutes with the AOJRF original coordination and with the AOJRF simplified coordination methods is statistically significant for the MCDM strategy (p-value$= 2.05 \cdot 10^{-5}$ for two robots and p-value$= 0.04321$ for three robots), but not for the AOJRF strategy (p-value$= 0.25$ for two robots and p-value$= 0.6972$ for three robots). Similarly, the difference between the no coordination and the AOJRF simplified coordination methods is statistically significant for the MCDM strategy (p-value$= 0.0147$ for two robots and p-value$= 0.0485$ for three robots), but not for the AOJRF strategy (p-value$= 0.23797$ for two robots and p-value$= 0.7304$ for three robots).

Tab. 1(b) shows the results for the open environment. Also in this case, the MCDM strategy seems to behave better than the AOJRF strategy with all the three coordination methods, although differences are not statistically significant. The difference between the safe area mapped at the end of the 15 minutes is more evident with the no coordination method, suggesting that a theoretically-grounded exploration strategy like MCDM can be more effective in limiting the problems of uncoordinated robots in the open environment.

In the open environment, the AOJRF simplified coordination method outperforms the other methods. The difference between the safe area mapped at 15 minutes with the AOJRF simplified coordination and with the AOJRF original coordination methods is statistically significant both for the MCDM strategy (p-value$= 5.00 \cdot 10^{-4}$ for two robots and p-value$= 0.0123$ for three robots) and for the AOJRF strategy (p-value$= 0.0113$ for two robots and p-value$= 0.02594$ for three robots). Similarly, the difference between the AOJRF simplified coordination and the no coordination methods is statistically significant both for the MCDM strategy (p-value$= 9.00 \cdot 10^{-4}$ for two robots and p-value$= 0.00131$ for three robots) and for the AOJRF strategy (p-value$= 8.00 \cdot 10^{-4}$ for two robots and p-value$= 0.0035$ for three robots).

The results for the office environment are rather surprising: coordinately allocating tasks to robots and allocating tasks without any coordination lead to the same performance. Although the initial separation of robots could help to decompose the problem, this observation can be explained by saying that what is predominantly important in exploring the highly structured office environment is the quality of the information used to evaluate the candidate locations (like the distance returned by path planning procedures instead of the Euclidean distance). This result does not contradict previous results that concluded that coordinated robots perform better than uncoordinated robots (see Section 2). It seems rather to complement previous works, which considered much simpler exploration strategies than those used in this paper. The use of exploration strategies, like MCDM and AOJRF strategies, that efficiently exploit good quality information to select observation locations effectively balances computational effort and accuracy of information. Indeed, although obtaining more accurate information (i.e., planning a path between the current location of the robot and the candidate location) requires more time and could represent a problem with the 15 minutes deadline, the resulting selection of a good observation location has a global benefit in highly structured environments.

The results for the open environment suggest that coordination becomes more important when the environment is less structured. This can be explained by noting that, in the office environment, robots can choose from many candidate locations and the intricate structure of the environment "pushes" robots to spread, while, in the open environment, the number of candidate locations is smaller and robots need to be coordinated to effectively spread across the environment and map it. Accordingly, in the open environment, the worst performance is obtained with the no coordination method, which is outperformed also by the random method, suggesting that assigning candidate locations randomly to robots is more effective than letting robots independently choosing their best candidate locations. In the open environment, the quality of information seems not so important (AOJRF simplified coordination method using Euclidean distance outperforms AOJRF original coordination method using distance returned by path planning procedures), mainly because obtaining accurate information requires some efforts, thus leaving less time to exploration, which can be performed quickly in uncluttered open environments.

5 Conclusion

This paper offered a first contribution to assess the relative influence of exploration strategies and coordination methods on the performance of multirobot systems employed in search and rescue applications. One of our results is that the quality of information used to evaluate candidate locations seems more relevant than assigning locations to robots in a coordinated way for a highly structured indoor environment. We are not claiming that coordination is useless, but that, in some settings, its impact on the exploration performance is less important than that of exploration strategies. From the other hand, in a less structured environment, coordination methods have a stronger impact than exploration strategies on the amount of area discovered.

The above conclusions are not yet definitive and need more efforts to be further assessed. For example, larger multirobot systems and other environments, exploration strategies, coordination methods, and integrated approaches will be considered. Also, more realistic situations involving real physical robots (with issues like damaged robots and loss of communication) and human disaster response teams will be considered. Finally, generalization of the outcomes of this paper to other applications involving exploration (like map building, where the quality of the map is an issue) could be investigated.

Acknowledgement. The authors gladly thank Chiara Nichetti for her contributions to the experimental part of this work.

References

1. Burgard, W., Moors, M., Schneider, F.: Coordinated multi-robot exploration. IEEE T. Robot. 21(3), 376–378 (2005)
2. Burgard, W., Fox, D., Moors, M., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proc. ICRA, pp. 476–481 (2000)
3. Sariel, S., Balch, T.: Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments. In: Proc. AAAI Workshop on Integrating Planning and Scheduling, pp. 27–33 (2005)
4. Amigoni, F.: Experimental evaluation of some exploration strategies for mobile robots. In: Proc. ICRA, pp. 2818–2823 (2008)
5. Gerkey, B., Mataric, M.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. 23, 939–954 (2004)

6. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: A robot simulator for research and education. In: Proc. ICRA, pp. 1400–1405 (2007)
7. Visser, A., Slamet, B.: Including communication success in the estimation of information gain for multi-robot exploration. In: Proc. WiOPT, pp. 680–687 (2008)
8. Lopez-Sanchez, M., Esteva, F., Lopez de Mantaras, R., Sierra, C., Amat, J.: Map generation by cooperative low-cost robots in structured unknown environments. Auton. Robot. 5, 53–61 (1998)
9. Ko, J., Stewart, B., Fox, D., Konolige, K., Limketkai, B.: A practical, decision-theoretic approach to multi-robot mapping and exploration. In: Proc. IROS, pp. 3232–3238 (2003)
10. Marjovi, A., Nunes, J., Marques, L., de Almeida, A.: Multi-robot exploration and fire searching. In: Proc. IROS, pp. 1929–1934 (2009)
11. Tovey, C., Koenig, S.: Improved analysis of greedy mapping. In: Proc. IROS, pp. 3251–3257 (2003)
12. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proc. Int'l Conf. Autonomous Agents, pp. 47–53 (1998)
13. Stachniss, C., Burgard, W.: Exploring unknown environments with mobile robots using coverage maps. In: Proc. IJCAI, pp. 1127–1134 (2003)
14. Gonzáles-Baños, H., Latombe, J.C.: Navigation strategies for exploring indoor environments. Int. J. Robot. Res. 21(10-11), 829–848 (2002)
15. Amigoni, F., Caglioti, V.: An information-based exploration strategy for environment mapping with mobile robots. Robot. Auton. Syst. 5(58), 684–699 (2010)
16. Tovar, B., Munoz, L., Murrieta-Cid, R., Alencastre, M., Monroy, R., Hutchinson, S.: Planning exploration strategies for simultaneous localization and mapping. Robot. Auton. Syst. 54(4), 314–331 (2006)
17. Amigoni, F., Gallo, A.: A multi-objective exploration strategy for mobile robots. In: Proc. ICRA, pp. 3861–3866 (2005)
18. Basilico, N., Amigoni, F.: Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. Auton. Robot. 31(4), 401–417 (2011)
19. Calisi, D., Farinelli, A., Iocchi, L., Nardi, D.: Multi-objective exploration and search for autonomous rescue robots. J. Field. Robot. 24(8-9), 763–777 (2007)
20. Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., Stewart, B.: Distributed multirobot exploration and mapping. Proc. IEEE 94(7), 1325–1339 (2006)
21. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: Proc. AAAI, pp. 852–858 (2000)
22. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proc. ICRA, pp. 3016–3023 (2002)
23. Hawley, J., Butler, Z.: Hierarchical distributed task allocation for multi-robot exploration. In: Proc. DARS, pp. 445–458 (2010)
24. Visser, A., de Buy Wenniger, G., Nijhuis, H., Alnajar, F., Huijten, B., van der Velden, M., Josemans, W., Terwijn, B., Sobolewski, R., Flynn, H., de Hoog, J.: Amsterdam Oxford joint rescue forces - Team description paper - RoboCup 2009. In: Proc. RoboCup (2009)
25. Pestman, W.: Mathematical Statistics: an Introduction. de Gruyter (1998)

Robot Localisation Using Natural Landmarks

Peter Anderson, Yongki Yusmanthia,
Bernhard Hengst, and Arcot Sowmya

School of Computer Science and Engineering,
University of New South Wales, UNSW Sydney 2052 Australia

Abstract. This paper introduces an optimised method for extracting natural landmarks to improve localisation during RoboCup soccer matches. The method uses modified 1D SURF features extracted from pixels on the robot's horizon. Consistent with the original SURF algorithm, the extracted features are robust to lighting changes, scale changes, and small changes in viewing angle or to the scene itself. Furthermore, we show that on a typical laptop 1D SURF runs more than one thousand times faster than SURF, achieving sub-millisecond performance. This makes the method suitable for visual navigation of resource constrained mobile robots. We demonstrate that by using just two stored images, it is possible to largely resolve the RoboCup SPL field end ambiguity.

1 Introduction

In the RoboCup soccer Standard Platform League (SPL), the field set-up has changed over the years to progressively remove navigation beacons and other colour coded visual cues. In keeping with this trend, in the 2012 SPL competition the goal-posts at either end of the field are to be made the same colour for the first time. This implies that a robot forced to localise from an unknown starting position will not be able to resolve one end of the field from the other. In RoboCup matches this requirement can arise after a complicated fall, for example when robots become entangled, slip, and are rotated unwittingly.

B-Human's 2011 Open Challenge demonstration addressed the field-end ambiguity challenge by using a team-wide ball model, enabling a kidnapped robot to recover by fusing their own ball observations with those of their team-mates [10]. The authors acknowledged, however, that this approach could fail in situations where a robot is alone, unaware that it has been kidnapped, or if the team-wide ball model is incorrect. An own goal is the potentially disastrous result of one of these localisation failures. To avoid these problems and to allow a single robot to localise, a method for extracting unique natural landmarks from images of the unspecified environment beyond the field is required. In this context, a natural landmark is defined as a set of scale-invariant local features that can be used to find point correspondences, and ultimately a perspective transformation, between two images containing the same object.

SURF (Speeded Up Robust Features) [2], [1] and SIFT (Scale-Invariant Feature Transform) [9] are two existing methods for extracting invariant local features from images. However, these methods are relatively computationally expensive and difficult or impossible to implement in real time on a resource constrained robot. To overcome these resource limitations, this paper introduces an optimised feature detector consisting of a modified one dimensional SURF algorithm (1D SURF), applied to a single row of grey-scale pixels captured at the robot's horizon. The horizon image is chosen for analysis because, for a robot moving on a planar surface, the identified features cannot rotate or move vertically, and must always remain in the same order. The use of a 1D horizon image and other optimisations dramatically reduces the computational expense of the algorithm, while exploiting the planar nature of the robot's movement and still providing acceptable repeatability of the features.

By using 1D SURF we show that a resource limited mobile robot is able to memorise and recognise natural landmarks seen at the horizon in typical indoor environments in real time. Consistent with the original SURF algorithm, the extracted landmarks are robust to lighting changes, scale changes, small scene changes and small changes in viewing angle. We have used the Aldebaran Nao humanoid robot to evaluate the 1D SURF algorithm, but the method could be applied to other vision-based robot localisation problems where the robot moves on a planar surface and can estimate the position of the horizon in images. The remainder of this paper is organised as follows: section 2 outlines related work, section 3 describes the 1D SURF algorithm and section 4 presents experimental results.

2 Background

Both SIFT [9] and SURF [2], [1] are feature representations that are designed to be stable under scale and viewpoint changes. Each method identifies potential features by searching for extrema at all possible scales of a grey-scale image. In SIFT, this step is implemented efficiently by using the difference of Gaussians function applied in scale-space to a series of smoothed and re-sampled images. Once features have been identified, they are accurately localised in both scale and location by interpolating from a 3D quadratic function fitted to local sample points. Next, feature points that are poorly located along an edge are eliminated and an orientation is assigned to each feature, so all future operations can be performed in a rotation invariant manner. SIFT calculates a 128-dimension descriptor vector for each identified feature based on the 8-bin histogram of the image gradient in 4x4 subregions around the feature point location. This, combined with the use of a Gaussian weighting function and normalisation of the descriptor vector, produces features that are invariant to scaling and rotation, as well as small viewpoint and illumination changes.

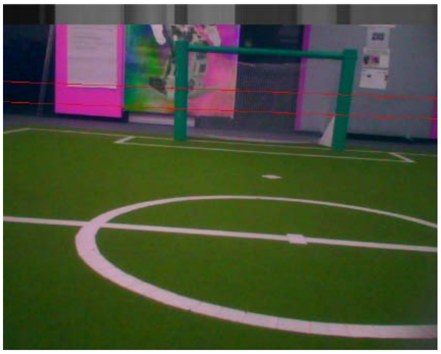
SURF is related to SIFT, but instead of using a Difference of Gaussian filter, SURF uses simple box filters which can be evaluated very efficiently using integral images. Box filters are used to approximate Gaussian second order partial derivatives and find the determinant of the Hessian matrix, which is referred to as the blob response at a particular location and scale. Features are yielded at local maxima of this response, found by thresholding the response and applying non-maximal suppression in a 3x3x3 neighbourhood over the image and in scales. Like SIFT, SURF also involves feature localisation by interpolating from a fitted 3D quadratic function, and orientation assignment. The SURF feature descriptor uses integral images in conjunction with Haar wavelets to calculate a 64-dimension descriptor vector. This is calculated by summing both the signed and absolute values of both the horizontal and vertical Haar wavelet response over 25 sample points to generate a 4-dimension vector in each of 4x4 subregions around the feature point location.

A comparison of SIFT and SURF using a standard testing procedure based on a range of real-world images found SURF to be faster and more accurate than SIFT [8]. For this reason we have chosen SURF as the basis of our 1D feature representation. Several other papers have adapted SIFT methods to operate on 1D data. The closest work to ours [3], [4], [5], use a 1D variation of SIFT to localise a mobile robot fitted with an omni-directional camera. This was achieved by identifying SIFT-like features in a 1D circular panoramic image, calculating feature descriptors based on colour and curvature information, and using a circular dynamic programming algorithm to match features between images. Compared to this work, we target a robot camera with a horizontal viewing angle of only 47.8 degrees, rather than 360 degrees, which dramatically reduces the amount of information available in a 1D horizon image. Furthermore, for reasons of computational efficiency we do not use colour information and use SURF rather than SIFT as the basis for our method.

3 1D SURF

In many respects the 1D SURF algorithm represents the equivalent of SURF, but using only one image dimension rather than two. SURF searches for blob response extrema in a 3D scale-space consisting of horizontal location, vertical location and scale. In 1D SURF, the search is conducted in a 2D scale-space consisting of horizontal location and scale only. However, there are also some other significant modifications and simplifications which were made to the original algorithm, as outlined below.

As indicated in Figure 1 Left, the input to the 1D SURF algorithm is a single row of grey-scale image pixels. The intensity values of these pixels are calculated by sub-sampling every 4 pixels along the robot's horizon, and taking the sum over a band of 30 vertical pixels at each sample point. The vertical sum minimises the sensitivity of extracted features to errors in the location of the horizon, and the sum is faster to compute than the mean. The resulting increase in pixel intensity values can be compensated in the response threshold. The position of the horizon in the image is determined by reading the robot's limb position sensors and calculating the forward kinematic chain from the foot to the camera, in accordance with the Denavit-Hartenberg parameters previously determined by the rUNSWift team [7].



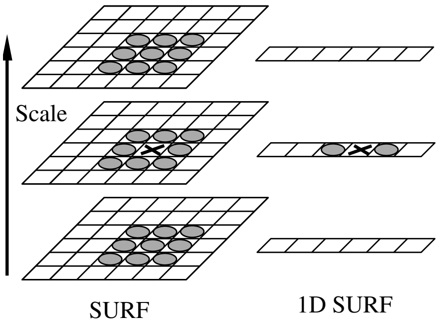


Fig. 1. Left: Image captured by the Nao robot showing superimposed 30 pixel horizon band in red, and the extracted grey-scale horizon pixels at the top of the image. Right: Identification of local maxima in scale-space. Pixel 'X' is selected as a maxima if it is greater than the marked pixels around it.

To identify local maxima of the blob response in scale-space, SURF thresholds the responses, then each pixel in 3D scale-space is compared to its 26 neighbours in a 3x3x3 neighbourhood to determine if it is a local maximum. In the case of 1D SURF, rather than searching for local maxima in a 3x3 scale-space neighbourhood, we apply a weaker test and only require that responses be extrema in the single space dimensional, as illustrated in Figure 1 Right. This relaxation ensures that sufficient feature points will be detected. It is an important aspect of the approach that a large number of relatively poor-quality features are generated, rather than relying on a small number of very distinctive features. A typical 1D horizon image containing 640 pixels might generate 50 - 70 features, depending on the parameter values chosen. In our case we use a scale-space consisting of 4 octaves of 3 intervals each.

Since in 1D SURF all features are defined with reference to the horizon, the SURF orientation assignment step is no longer necessary and can be disregarded. SURF interpolates the location of features in both space and scale to sub-pixel accuracy by fitting a 3D quadratic curve to the local image function. In our application, we found that the additional accuracy provided by this step was not worth the computational burden, and it was also discarded. Finally, although the 1D SURF feature descriptor is calculated analogously to the SURF feature descriptor, due to the reduction in sample space and by using 3 subregions instead of 4, we produce a 6-dimension feature descriptor rather than a 64-dimension feature descriptor, allowing for much faster matching of descriptors across images.

3.1 Application to Natural Landmark Recognition

A simple method is presented to memorise, and subsequently recognise, natural landmarks using 1D SURF features. Given a test image and a stored image, landmark recognition is performed by matching features in the test image to their nearest neighbours in the stored image, based on the Euclidean distance between feature descriptors. As before [9], feature matches are considered to be valid if the nearest-neighbour distance ratio is less than 0.7. Similarly [5], we then assign a recognition score to the test image calculated as the sum over all valid matched features of the inverse distance between feature descriptors. A high recognition score indicates that the test image contains the same natural landmark as the stored image with high likelihood.

The above method, which we will refer to as nearest neighbour (NN) matching, does not preclude feature matches that are out of order, or otherwise inconsistent in terms of scale or horizontal displacement. Therefore a second matching method is presented, which first matches nearest neighbour features, and then uses RANSAC [6] to discard feature matches that do not agree on a consistent landmark pose, before recalculating the recognition score. A consistent pose is defined as a set of matched features that conform to a straight line matching function as follows, where $x_{test,i}$ and $x_{stored,i}$ represent the horizontal pixel location of the ith matched feature in the test and stored images respectively, and β_s and β_d are scaling and displacement parameters:

$$x_{test,i} = \beta_s x_{stored,i} + \beta_d \tag{1}$$

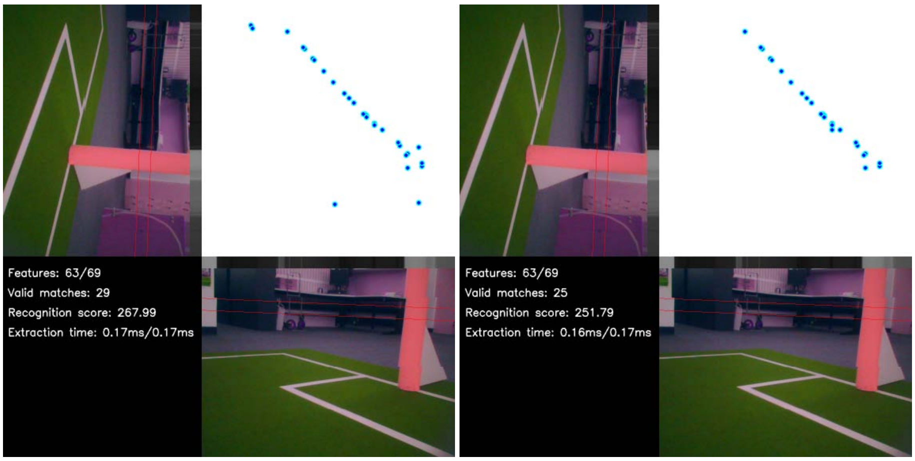
Given the robot's limited horizontal field of view, we find a straight line matching function is a reasonable approximation of the true feature matching function, which is curved in the presence of translation. Compared to NN matching, recognition scores calculated with this method will be lower, but have potentially greater discriminatory power. We will refer to this method as nearest neighbour matching with RANSAC (NN with RANSAC). The further advantage of this method is that it provides useful information about the robot's motion between the two images. The use of RANSAC to discard inconsistent matches generated by NN matching is shown in Figure 2.

4 Experimental Results

Two experiments were used to evaluate the performance of 1D SURF for robot localisation. In each experiment, the images used were captured using the Aldebaran Nao RoboCup edition v3.2, a humanoid robot equipped with a 500MHz AMD Geode LX800 processor. The Nao has two 640x480 pixel 30 fps digital cameras, each with a horizontal field of view of 47.8 degrees, which can be accessed one at a time.

4.1 Classification Experiment

The first experiment was designed as a classification task, to assess whether the recognition score between two images could be used by the robot to determine whether both images contained the same landmark. Data for the experiment



```
Features: 63/69
Valid matches: 29
Recognition score: 267.99
Extraction time: 0.17ms/0.17ms
```

```
Features: 63/69
Valid matches: 25
Recognition score: 251.79
Extraction time: 0.16ms/0.17ms
```

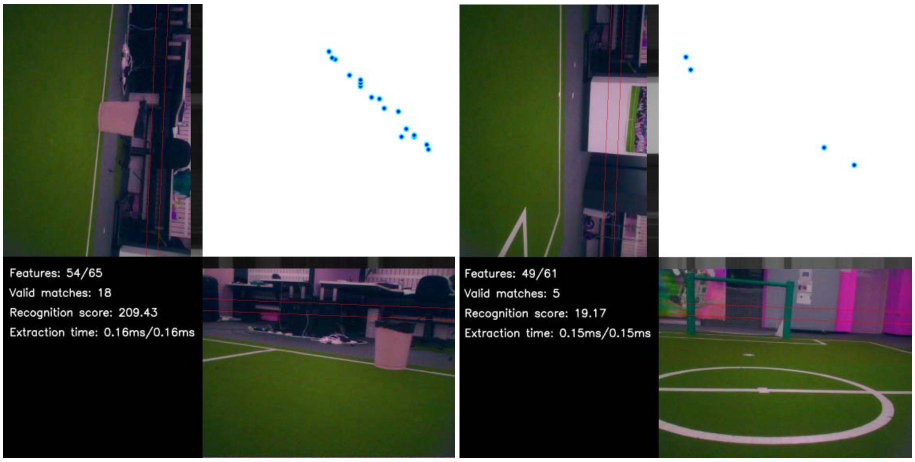
Fig. 2. Left: Matching features in two similar images based on nearest neighbour (NN) matching. Right: Matching features in the same two images after using RANSAC to discard matches that don't agree on a consistent pose (NN with RANSAC). As in Figure 1, each image displays the horizon band in red and the extracted grey-scale horizon pixels at the top of the image. Matching features are plotted in the top-right panel against their horizon location in each image. The text panel illustrates the number of features detected in each image, the number of matches, the recognition score and the time taken to extract the features on a 2.4GHz laptop.

was captured by rotating the robot at a single location on the field, and capturing 88 images at approximately 4 degree increments. During this process the background around the field consisted of a typical office environment. From this image library we generated a test bank of 480 matched images and 2,065 unmatched images. Two images were considered to match if the angle between them was less than 20 degrees, implying at least 58% of each image horizon overlapped with the other image. Example images from the test bank and the resulting recognition scores are shown in Figure 3.

Although this experiment contains no changes in scale, illumination or viewing angle, it provides a useful baseline against which to tune parameters and assess the likely rate of false positive landmark recognitions. Feature extraction and matching was performed off-board the robot using a 2.4GHz Core 2 Duo Processor laptop. This enabled the classification accuracy and speed of 1D SURF to be easily compared against SURF, for which we used the OpenSURF[1] library implementation.

The sensitivity and specificity of SURF (using NN matching) and 1D SURF (using NN matching, and NN with RANSAC matching) with variation in the recognition score discrimination threshold is shown in Figure 4. 1D SURF (using the horizon pixels only) is clearly less robust than SURF (processing the entire

[1] `http://www.chrisevansdev.com/computer-vision-opensurf.html`



```
Features: 54/65
Valid matches: 18
Recognition score: 209.43
Extraction time: 0.16ms/0.16ms
```

```
Features: 49/61
Valid matches: 5
Recognition score: 19.17
Extraction time: 0.15ms/0.15ms
```

Fig. 3. Left: Two images that almost completely overlap. Although the features on the horizon are not very distinctive, a high recognition score is generated using 1D SURF and NN with RANSAC feature matching. Right: Two images with no overlap, resulting in a low recognition score using the same method. As before, matching features are plotted in the top-right panel against their horizon location in each image, and the text panel contains key statistics.

image). However, as illustrated in Table 1, 1D SURF uses only a fraction of the features, and runs more than one thousand times faster than SURF in this experiment. With the mean extraction time below 0.2ms, real-time feature extraction on the Nao during RoboCup soccer matches is a clear prospect. Also, using RANSAC to enforce a consist landmark pose results in a small improvement in classification accuracy.

Table 1. Running time of feature extraction and matching algorithms evaluated on a 2.4GHz Core 2 Duo laptop

Feature extraction technique	Feature matching technique	Mean no. features	Mean extraction time (ms)	Mean matching time (ms)	Area under ROC curve
SURF	Nearest neighbour (NN)	429	222.3	19.1	98.8%
1D SURF	Nearest neighbour (NN)	59.2	0.158	0.069	88.0%
1D SURF	NN with RANSAC	59.2	0.158	0.076	89.6%

4.2 Field Experiment

Having validated the performance of 1D SURF on highly similar images, the second experiment was designed to assess the performance of 1D SURF under

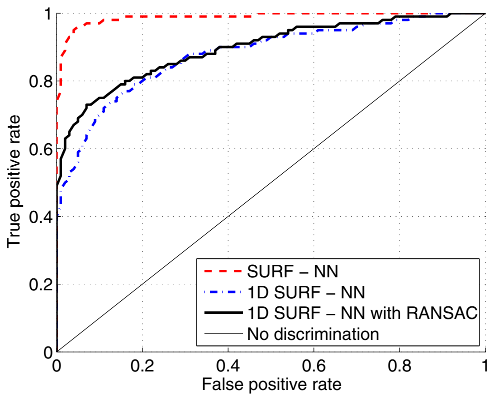


Fig. 4. ROC curve for classifying test images as matched or unmatched using the recognition score. Using NN with RANSAC matching on this data set, a threshold recognition score of 100 captured 70% of true positives with a 5% false positive rate.

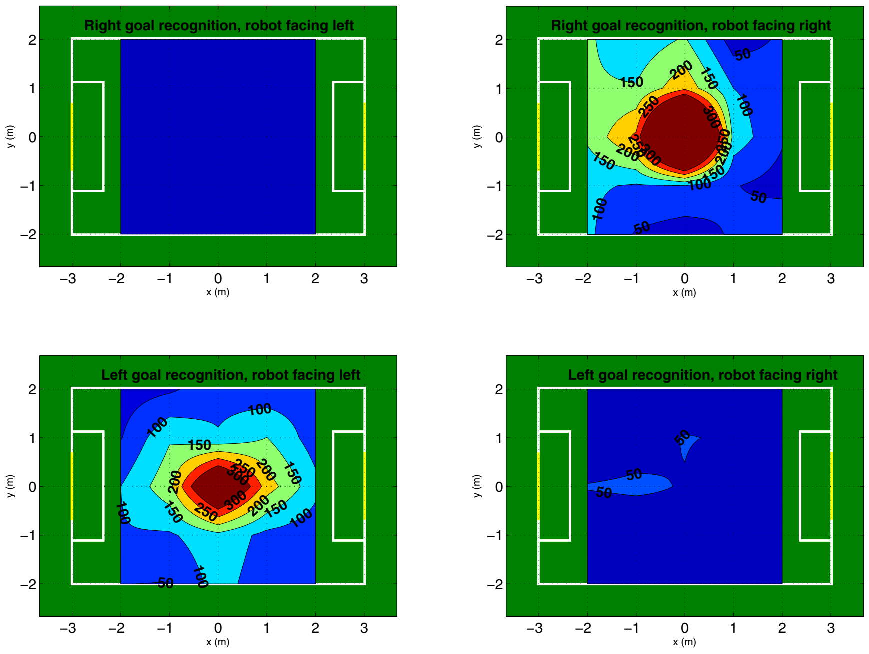


Fig. 5. Recognition scores of a single goal image from different areas of the field. Clockwise from top left: Recognition of right-hand goal when facing left, recognition of right-hand goal when facing right, recognition of left-hand goal when facing right, recognition of left-hand goal when facing left.

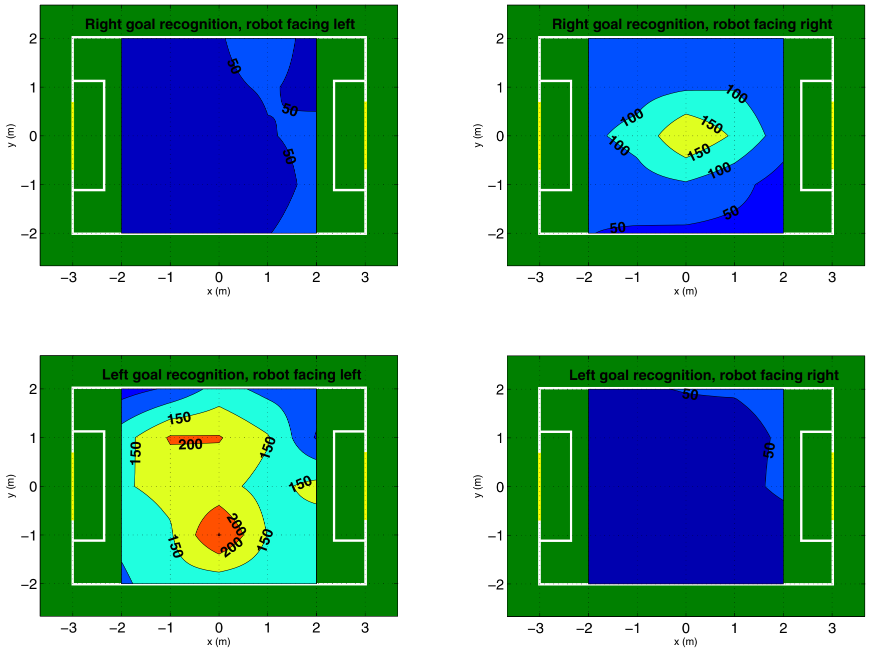


Fig. 6. Recognition scores of a single goal image from different areas of the field, with the overhead field lighting turned off, and the goals themselves removed. Clockwise from top left: Recognition of right-hand goal when facing left, recognition of right-hand goal when facing right, recognition of left-hand goal when facing right, recognition of left-hand goal when facing left.

changes in scale, viewing angle, illumination and with small scene changes. It was performed on-board the Nao robot to provide a clearer assessment of the processing speed of the method with constrained hardware. In this experiment, we used NN with RANSAC matching and evaluated 1D SURF the way it might be used in a SPL match; to distinguish one end of the field from the other. To do this, we positioned the Nao in the centre of the field, captured one image of each goal area, and stored the extracted feature vectors. Next, we moved the Nao through a 1m grid of positions covering a 4m x 4m area of the field (25 positions in total), and recorded the recognition scores at each point when manually positioned to face approximately towards each goal. By moving the Nao around the field, large changes in scale and viewing angle were generated. At each point we hoped to observe a large recognition score for the stored goal the robot was actually facing, and a low recognition score for the other goal, indicating that this technique could be used to reliably distinguish field ends during a match. During this entire experiment both goals were coloured yellow, and background objects were approximately 2m behind the goals themselves.

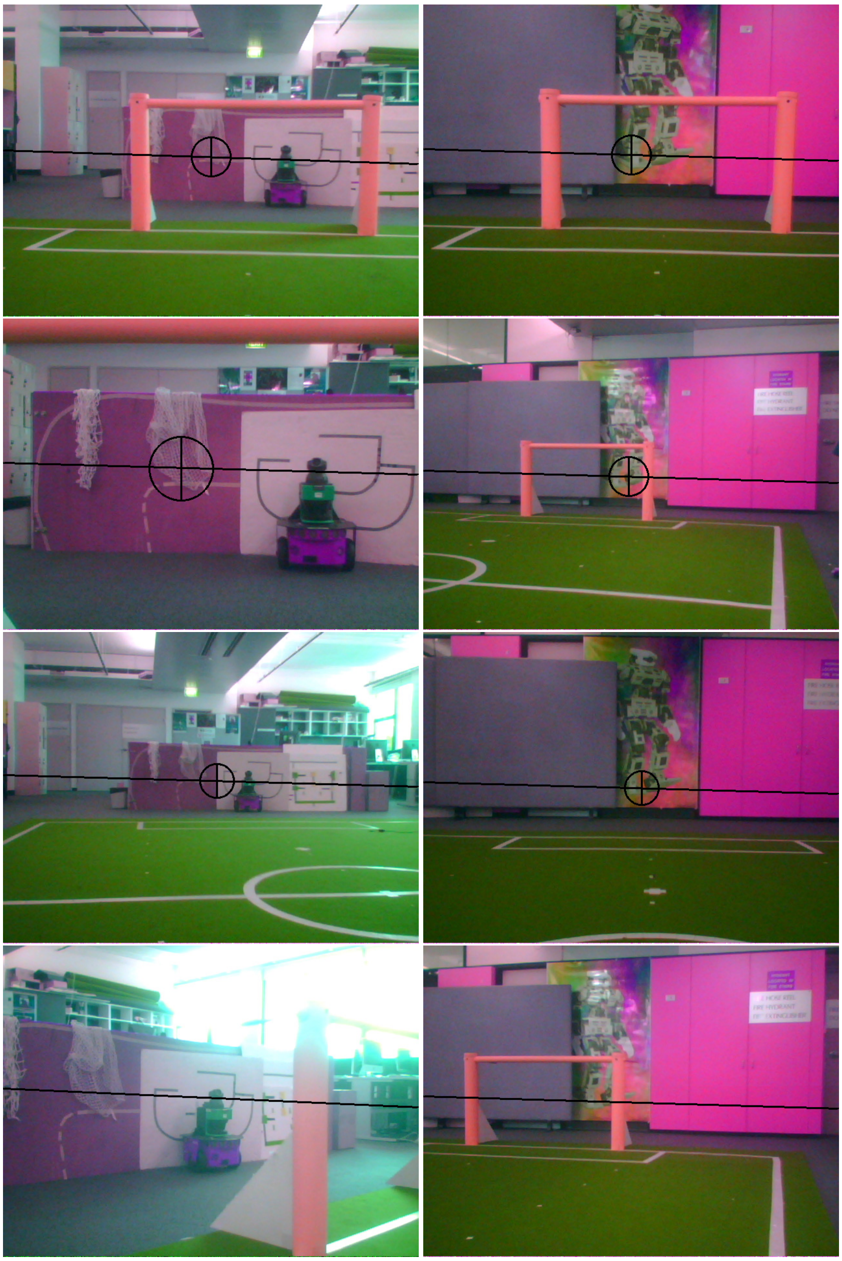


Fig. 7. Top row: Stored images of the left-hand and right-hand goal areas respectively. Row 2: Examples of correctly matched field views. Markers indicate the scale and position of the match. Row 3: Examples of correctly matched views with goals removed and overhead lights turned off. Bottom row: Some field views that could not be confidently matched to the stored images, possibly due to overexposure and occlusion of key features respectively.

The recognition scores recorded during this exercise are overlaid on a field map in Figure 5. Using a recognition threshold of 100 (as determined during the classification experiment), each field end is correctly recognised from the single stored image in more than half of the 4m x 4m test area. A very strong recognition response (greater than 200) is observed in a radius of approximately 1m around the location of the original stored image. Finally, there were zero false positives recorded when facing the opposite end of the field. The recognition response to the opposite end of the field is almost always less than 50. Overall, these results indicate that even with just two stored images, a kidnapped robot could resolve one end of the field from the other from most mid-field positions. To provide a clearer indication of the field environment used during the experiment, Figure 7 depicts the stored goal images and examples of views from different areas of the field.

In many robot navigation applications, including RoboCup SPL, robots are subject to varied lighting conditions and the natural landmarks in a given scene will change over time. To test the robustness of 1D SURF in the face of these challenges, we repeated the experiment with the overhead field lighting turned off and both goals removed (to simulate some measurable change to the original scene). The stored features extracted from the original goal images were not changed. As shown in Figure 6, the recognition response to the correct field end is less peaked than before, but the recognition area is still large and again there are no false positives. It is interesting to note that the recognition area for the left-hand goal actually increases once the goal itself is removed. The goal itself can actually be something of a nuisance in the recognition process, since with large perspective changes it occludes features in the background that might otherwise be identified. Using a representative sample of field locations, the mean execution time to extract 1D SURF features on the Nao robot was 12ms. Although this is considerably slower than the 0.158ms extraction time achieved on the laptop, it is still fast enough to enable features to be extracted in real time at the full 30 fps frame rate of the Nao camera.

5 Evaluation and Conclusion

This paper has presented an optimised method for extracting local features from 1D images of a mobile robot's horizon. The extracted 1D SURF features are robust to lighting changes, scale changes, and small changes in viewing angle or to the scene itself, making them suitable for robot navigation in indoor environments. Using 1D SURF features and a NN with RANSAC matching technique, we demonstrate that (in a relatively distinctive environment with few scene changes) it is possible to resolve the RoboCup SPL field end ambiguity in real time using just two stored images.

In actual RoboCup matches, it is likely that the background environment will be more challenging than our laboratory tests due to the coming and going of spectators during the match. As such, we anticipate that in practise it will be necessary to store more than two images, and to update them during the match as the natural landmarks around the field change. In future work we will investigate methods to simultaneously localise and map changing natural landmarks around the field, rather than relying on a fixed set of stored images.

Acknowledgements. The authors gratefully acknowledge the support of the School of Computer Science and Engineering, past and present members of the rUNSWift team, and associated staff and students in the school's robotic laboratory.

References

1. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Computer Vision and Image Understanding 110(3), 346–359 (2008)
2. Bay, H., Tuytelaars, T., Van Gool, L.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
3. Briggs, A., Detweiler, C., Mullen, P., Scharstein, D.: Scale-space features in 1d omnidirectional images. In: Omnivis 2004, the Fifth Workshop on Omnidirectional Vision, Prague, Czech Republic, pp. 115–126 (2004)
4. Briggs, A., Li, Y., Scharstein, D., Wilder, M.: Robot navigation using 1d panoramic images. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, pp. 2679–2685. IEEE (2006)
5. Briggs, A.J., Detweiler, C., Li, Y., Mullen, P.C., Scharstein, D.: Matching scale-space features in 1d panoramas. Computer Vision and Image Understanding 103(3), 184–195 (2006)
6. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)
7. Hartenberg, R.S., Denavit, J.: A kinematic notation for lower pair mechanisms based on matrices. Journal of Applied Mechanics 77, 215–221 (1955)
8. Juan, L., Gwun, O.: A comparison of sift, pca-sift and surf. International Journal of Image Processing (IJIP) 3(4), 143–152 (2009)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
10. Röfer, T., Laue, T., Müller, J., Fabisch, A., Feldpausch, F., Gillmann, K., Graf, C., Haas, T.J.d., Härtl, A., Humann, A., Honsel, D., Kastner, P., Kastner, T., Könemann, C., Markowsky, B., Riemann, O.J.L., Wenk, F.: B-human team report and code release 2011 (2011), `http://www.b-human.de/publications/`

Solving Multi-agent Decision Problems Modeled as Dec-POMDP: A Robot Soccer Case Study

Okan Aşık and H. Levent Akın

Boğaziçi University, Department of Computer Engineering, 34342, İstanbul, Turkey

Abstract. Robot soccer is one of the major domains for studying the coordination of multi-robot teams. Decentralized Partially Observable Markov Decision Process (Dec-POMDP) is a recent mathematical framework which has been used to model multi-agent coordination. In this work, we model simple robot soccer as Dec-POMDP and solve it using an algorithm which is based on the approach detailed in [1]. This algorithm uses finite state controllers to represent policies and searches the policy space with genetic algorithms. We use the *TeamBots* simulation environment. We use score difference of a game as a fitness and try to estimate it by running many simulations. We show that it is possible to model a robot soccer game as a Dec-POMDP and achieve satisfactory results. The trained policy wins almost all of the games against the standard *TeamBots* teams, and a reinforcement learning based team developed elsewhere.

Keywords: DEC-POMDP, genetic algorithms, robot soccer, simulation, high-level planning.

1 Introduction

Robots are physical agents which interact with their environment via their sensors and actuators. The main problem of a robot is finding a method to map its sensor inputs to actuator outputs to achieve its designated goal. This can be modeled as a decision making problem. There are many methods to solve decision making problems. Approaches based on *Markov Decision Process* (MDP) models are widely used compared to other methods.

There are some tasks which require the cooperation of agents, such as robot soccer. All robots act autonomously, but they should be coordinated. Decision making is a more complicated problem for such multi-robot situations because individual actions of the robots should result in the completion of the task of the team, such as scoring. Decentralized Partially Observable Markov Decision Process (Dec-POMDP) model is one of the promising approaches to solve multi-agent decision making under uncertainty. There are different formalizations for Dec-POMDP, in our study we use Bernstein's model [2].

In this paper, we model robot soccer as a Dec-POMDP problem and use the GA-FSC algorithm in [1]. The algorithm represents policies as finite-state controllers and searches the policy space with genetic algorithms. We use *TeamBots* [3] 2D robot soccer simulator as the simulation environment. We show that it is possible to develop a successful team that defeats all the predefined teams in the *TeamBots* environment and also a reinforcement learning based team developed in another study [4].

The organization of the rest of the paper is as follows. Section 2 introduces related work. Section 3 presents the algorithm we used to solve Dec-POMDP problem. Section 4 introduces our experiments and results. We present our conclusions and intended future work in Section 5.

2 Related Work

We can categorize Dec-POMDP algorithms as exact and approximate algorithms. Optimally solving Dec-POMDP problems, have been shown to be NEXP-complete [5]. Therefore, exact solutions are not feasible for almost all real-world applications, and the current research is mainly about finding approximate solutions. The algorithms developed so far are generally tested on benchmark Dec-POMDP problems such as Dec-Tiger, multi-access broadcast channel, meeting in a grid, box pushing, and fire fighting problems [1]. They are used to compare and contrast the performances of different algorithms.

Wu and Chen solves the soccer problem modeled as a Dec-POMDP with Correlation-MDPs in the RoboCup domain [6]. They base their work on the memory-bounded dynamic programming algorithm proposed by Bernstein *et al* [2]. Their main contribution is proposing an approximate algorithm to calculate the correlation device. They used the algorithm to improve the coordination of soccer playing agents in the RoboCup 2006 Soccer 2D Simulation Competitions, and they won all the matches except one. This study is important in terms of showing the capabilities of the Dec-POMDP framework in the robot soccer domain.

Keepaway soccer was put forth as a testbed for machine learning [7], and there is a wide variety of reinforcement algorithms which are tested with keepaway soccer [8, 9, 10, 11]. Di Pietro *et al* used evolutionary algorithms to learn a policy which results in coordinated behavior [12]. They formulate the problem so that the agent decisions are based on parameters such as the distance to the recipient. The evolutionary algorithm searches for the optimal parameters to keep the ball as long as possible which is the ultimate goal of keepaway soccer. This work is close to our work in terms of using an evolutionary algorithm and trying to solve the soccer problem, but their solution is problem specific which is a sub-problem of robot soccer.

Although there are many studies on how to learn to play soccer, they have either combined their solution with the existing planning framework or solved a subset of soccer problem such as keepaway soccer [7, 13]. In this paper, we model robot soccer as a Dec-POMDP and represent the policy as a finite state controller. The robots execute the trained policy represented as finite state controllers throughout the game.

3 Solving Problems Modeled as Decentralized Markov Decision Processes

The *Decentralized Partially Observable Markov Decision Process (DEC-POMDP)* [5] model consists of 7-tuple $(n, S, A, T, \Omega, Obs, R)$ where:

- n is the number of agents.
- S is a finite set of states.
- A is the set of joint actions which is the Cartesian product of A_i $(i = 1, 2..., n)$ i.e. the set of actions available to $agent_i$.
- T is the state transition function which determines the probabilities of the possible next states given the current state S and the current joint action a.
- Ω is the set of joint observations which is the Cartesian product of Ω_i $(i = 1, 2..., n)$ i.e. the set of observations available to $agent_i$. At any time step the agents receive a joint observation $o = (o_1, o_2, ..., o_n)$ from the environment.
- Obs is the observation function which specifies the probability of receiving the joint observation o given the current state S and the current joint action a.
- R is the immediate reward function specifying the reward taken by the multiagent team given the current state and the joint action.

3.1 Dec-POMDP Policies and Finite State Controllers

A Dec-POMDP policy is a mapping of the observation history to the actions. Generally, policies are represented as a policy tree where observations lead to actions. However, the tree representation is not sufficiently compact. The Finite state controller (FSC) representation is one of the viable candidates to represent policies. A FSC is a special finite state machine. It consists of a set of states and transitions. The main difference here is that those states called FSC nodes, and are abstract and different from the environment states. Every FSC node corresponds to one action which is the best action for that particular state. Transitions take place when a particular observation is taken at a particular FSC node. An example finite state controller can be seen in Figure 1. This finite state controller is designed for a problem having only two observations and three actions. In a FSC, there is always a starting state. Let us assume that the starting state is $S1$ so that $A1$ is executed first. If the robot gets an observation $O2$, it updates its current FSC node to $S2$ and executes the action $A2$. Action execution and FSC node update continues until the the end of the episode. This finite state controller represents the policy of a single robot. The critical point about the finite state controller representation is that we can model a Dec-POMDP policy with different numbers of nodes. Since every node corresponds to one action, the minimum number of nodes is the number of actions. Since having greater number of nodes than the number of actions does not improve the performance of the algorithm[1], in our experiments, the number of FSC nodes is equal to the number of actions.

3.2 Genetic Algorithms

In genetic algorithms, a candidate solution is encoded in a chromosome and the set of all chromosomes is called a *population*. The fitness of a candidate solution determines how good the candidate is. Through the application of evolutionary operators such as selection, crossover, and mutation, a new population is created from the current population. When the convergence criteria are met, the algorithm terminates and the best candidate becomes the solution of the algorithm [14].

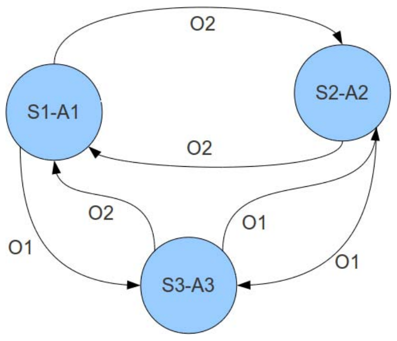


Fig. 1. An Example Finite State Controller

Encoding. In order to solve a Dec-POMDP using genetic algorithms, we should encode the candidate solution, the policy. In this study, the encoding of a FSC as a chromosome is as follows: the first n genes represent node-action mapping and their values are between 1 and the number of actions (A). Then, for each node, there is an observation-node mapping which denotes the transition when an observation is taken as seen in Figure 2. The value of this range is between 1 and S which represents the number of nodes. The whole chromosome of the Dec-POMDP policy is constructed by concatenating every robot's policy.

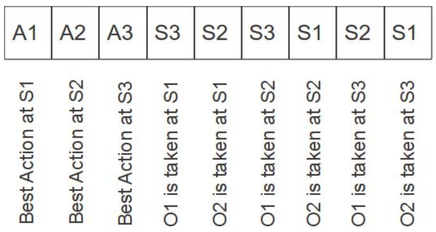


Fig. 2. An Example FSC Encoding

Fitness Calculation. Fitness calculation is one of the most critical parts of any genetic algorithm. For Dec-POMDP problems for which transition and reward functions can be stated, it is possible to calculate fitness values for a given policy. However, for problems with unknown transition and reward functions, only approximate fitness calculation is possible.

One method of calculating fitness approximately is by running a large number of simulations with a given policy. The fitness of a policy have been shown to stabilize after 1000 simulations for Dec-POMDP benchmark problems [1]. However, for a stable fitness calculation, we should run as many simulations as possible, but the reasonable number of simulations is highly problem dependent. There is a trade-off between the precision of the calculation and the running time complexity of the calculation. One of the most important factors that have an effect on choosing the number of simulations is accuracy. We need to estimate the fitness value sufficiently accurately so that the chromosomes can be ranked.

3.3 The GA-FSC Algorithm

Even though an evolutionary strategy based approach has been proposed in [15], it has been shown to be not sufficiently scalable with the number of agents. In [1] it has been shown that the finite state controller based approach performs better than the previous approach in [15]. For this reason we use the genetic algorithms based approach proposed in [1].

This algorithm has two major components :

- **Encoding the candidate policy:** A policy is represented as a FSC and is encoded as an integer chromosome whose details will be given below.
- **Searching the policy space for the best policy with genetic algorithm:** In [1], two fitness calculation approaches are proposed: exact and approximate. For the robot soccer problem considered here, exact calculation is not possible since the dynamics of the environment are not known exactly. The approximate calculation method, however, relies on running many simulations with a given policy and taking the average reward of those simulations as the fitness of the policy.

The algorithm has three stages: pre-evolution, during evolution, and post-evolution. After a random population is formed, the k best chromosomes are selected based on their fitnesses. Those k chromosomes are copied to the best chromosomes list. At the end of each generation, the best k chromosomes of the population are compared to the chromosomes in the best chromosomes list, if it one of the best chromosomes of this generation is better than one of the current best chromosomes, its fitness is calculated more precisely by running additional simulations. If it is still better, it is added to the current best chromosomes list. At the end of the evolution which is determined by setting a maximum generation number, the best of best chromosomes list is determined by running additional simulations. In this study, we keep 10 chromosomes in the best chromosomes list.

3.4 Robot Soccer Dec-POMDP Model

We use the *TeamBots* simulation environment [3] as a testbed for our Dec-POMDP algorithm. The model is directly related to the simulation environment. Different models are required for different simulation environments. Since we have already used *TeamBots* simulation in different studies, we have a well-established MDP model. To model the robot soccer as Dec-POMDP model, we need to define the set of actions, set of observations and the number of states. The finite set of actions is as follows:

$$A = \{\textit{Go to ball}, \textit{Go to support position}, \textit{Go to defense position}, \\ \textit{Pass to the closest teammate}, \textit{Pass to the teammate closest to the opponent goal}\}$$

The finite set of observations is as follows: The *TeamBots* field is divided with 2 equally spaced lines from the narrow edge and 3 equally spaced lines from the wide edge. In total there are 12 grid cells as seen in Figure 3. The *Location* information is based on this grid.

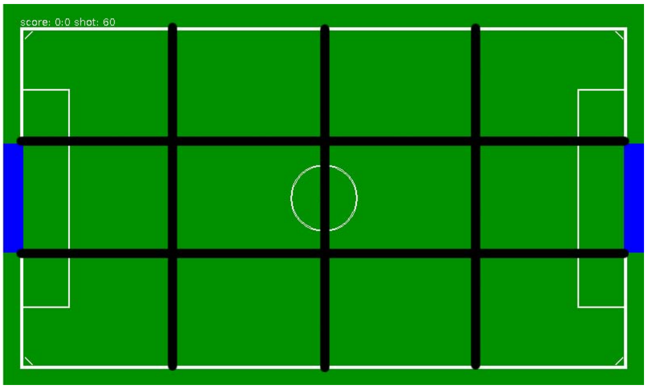


Fig. 3. *TeamBots* Field

We define two observation metrics in those grid cells. The first observation metric called *Dominance* has three possible values based on the number of players in the cell the ball resides:

– *Equal number of players*,
– *The opponent team has more players*, and
– *Our team has more players*.

The other observation metric is called *Closeness*. It also has three possible values which are based on which player is the closest to the ball:

– *An opponent player is the closest*,
– *A teammate is the closest*, and
– *The robot itself is the closest*.

Therefore, the observation set includes three critical pieces of information about the environment: The location of the the ball in the grid, the player the closest to the ball, and the team which is the dominant one in the cell where the ball resides.

$$Observation = Location \times Closeness \times Dominance$$

4 Experiments and Results

All the experiments in this study are done with the *TeamBots* simulation environment using the $JGAP$ genetic algorithms package [16]. In the standard *TeamBots* package there are four standard teams. They are in the order of increasing power: *BrianTeam*, *Kechze*, *SibHeteroG*, *AIKHomoG*. In addition there is a team called *NullTeam* which is used for learning very basic behaviors such as dribbling the ball. The players of the *NullTeam* are immobile during the game. The matches are played with teams of 5 players.

We train against all teams iteratively starting from the easiest team up to the hardest team. Our ultimate goal is to fine tune the algorithm so that it is best suited for solving the robot soccer problem modeled as a Dec-POMDP. Since we need a stable fitness calculation, the number of simulations used for estimating the fitness of a candidate policy is one of the parameters we need to determine.

4.1 Genetic Algorithm

When we define our problem as a Dec-POMDP and use GA-FSC as a solver, the quality of the solution is highly dependent on the parameters of the genetic algorithm. We determined the genetic algorithm parameters shown in Table 1 empirically.

Table 1. Parameters of the Genetic Algorithm

Parameter	**Value**
Population Size	50
Mutation Rate	0.1
Crossover Rate	0.5
N_B : Number of Simulations Before Evolution	100
N_D : Number of Simulations During Evolution	50
N_A : Number of Simulations After Evolution	500
Fitness Metric	Score
Maximum Number of Generations	50
Convergence Limit	20

The evolution cycle for training the Dec-POMDP team against a selected standard team is as follows. The first population is initialized randomly. Then, we determine the best chromosomes of the evolution by running N_B simulations. In each generation, we determine the fitness of the chromosomes in the population by running N_D simulations. At the end of every generation, we get the top 10 chromosomes of the population and recalculate their fitness by running N_B simulations. If any one of them is still good enough to be in the best chromosomes list, it is added to the list and the evolution continues. As the termination criteria we use reaching the maximum number of generations or the maximum fitness not changing for a specified number of generations. When the evolution ends we calculate the best solution from the best chromosomes list by running N_A simulations.

Training is carried out in stages. We first train against the *NullTeam*, then against the other standard *TeamBots* teams, in the order of increasing difficulty. The population of a previous team is used for the next team except the *NullTeam* whose population is randomly initialized.

4.2 Fitness Calculation

The main problem about the fitness calculation is that we try to estimate the fitness of a policy by taking many simulation runs. Therefore, we need to find the number of simulation runs which is enough to rank the chromosomes so that the genetic algorithm can converge. In Figure 4, we show the change in the rank of 50 chromosomes over the number of simulations. The change in rank is calculated by summing the change of all chromosomes between two consecutive runs. It is found that 50 simulation runs are enough to distinguish good solution candidate since after 50 simulations the rank of chromosomes do not oscillate much. However, we need to determine two more numbers for simulation runs to achieve higher precision when deciding whether the policy is good enough to be kept as one of the best solutions, and when deciding what is the best of all best candidates. By considering running time limitations, we choose 100 simulation runs to decide whether a policy is good enough to be in the best chromosome list, and we choose 500 simulation runs to decide what is the best solution of best chromosomes list.

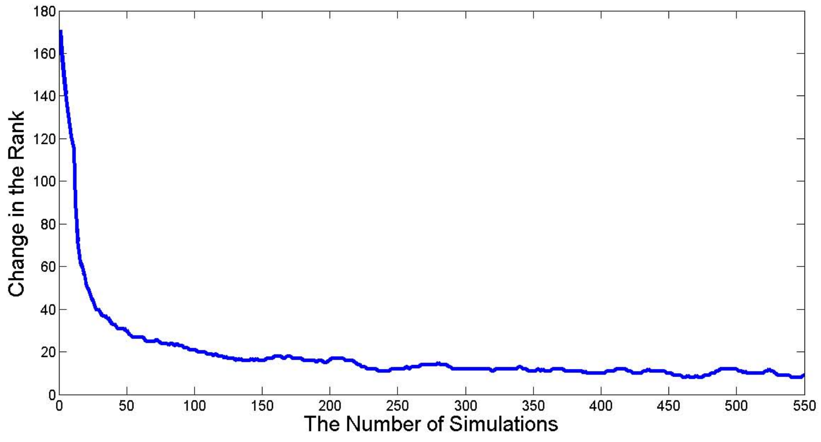


Fig. 4. The Change in the Rank of Chromosomes by the Number of Simulations

In robot soccer, the fitness of a policy can be calculated in different ways. One of the possible fitness calculation methods is the score difference. However, score difference may not be a good method since it may not be selective enough to differentiate a good soccer player policy from a bad one when their score is the same. When policies are randomly initialized, none of the policies in the population scores goals against the good teams so that they all have the same fitness. We know that some policies are more successful at playing soccer, but they cannot score. Those chromosomes should be selected for next generations. Therefore, to solve this problem, we train policies iteratively starting with the weaker teams and continuing with the stronger teams. The performance of the method can be seen in Table 2.

Table 2. The Performance of Iterative Training with Score Difference Fitness Method

Opponent	**Average Score Difference of 500 Evaluation Runs**	**Average Score Difference at The End of Evolution for That Team**	**Best Score Difference**	**Win**	**Draw**	**Loss**
NullTeam	8.42	43.96	19	499	1	0
BrianTeam	7.04	22.9	13	500	0	0
Kechze	3.68	4.97	9	493	7	0
SibHeteroG	1.31	1.74	4	399	90	11
AIKHomoG	2.48	3.77	7	460	37	3
Mericli *et al* team (RL-Based)	1.74	N.A.	6	421	78	1

The difference between the average scores at the end of evolution and the average scores of 500 evaluation runs is high for weak teams such as *NullTeam*, and *BrianTeam*. Since the policies trained against those teams easily converge to successful policies which are a series of simple actions, the score of the evaluation run is lower than the score at end of evolution for that team. Another reason for this difference is that the final best policy is highly adapted to the last teams it is trained against.

One of the most important performance measures for the algorithm is the number of wins and losses. As it is seen in Table 2, the trained policy never loses against *NullTeam*, *BrianTeam*, *Kechze*, and loses only 11 games against *SibHeteroG*, 3 games against *AIKHomoG* out of 500 games. Although, the average score difference against *SibHeteroG*, and *AIKHomoG* is not very high, the number of wins are quite satisfactory.

In addition to the standard *TeamBots* teams, we also report the average scores against the team trained by Mericli *et al* [4]. Even though our team was trained only against the *TeamBots* teams we have a positive average score against the Mericli *et al* team and we win most of the games as seen in Table 2.

4.3 Evaluation of DEC-POMDP Policies

Although there is no benchmark for the *TeamBots* simulation environment, in order to assess the performance of our method we compare our average score with the scores reported in [4]. Although the focus of the work reported in [4] is different from our work, both studies use the same MDP model and the simulation environment, i.e., the same basic actions, state definition, and observation definition. They use the reinforcement learning approach with soccer metrics developed by Mericli *et al* [17]. In Table 3, we compare our results with the scores reported in [4]. Although, our average scores are lower, we achieve positive average scores against all teams and win most of the games against *SibHeteroG*. However, the reinforcement learning based team has a negative average score against *SibHeteroG*.

Table 3. The Comparison of Average Scores

Opponent Team	Average Scores of Dec-POMDP Based Approach	Average Scores of Reinforcement Learning Based Approach [4]
NullTeam	8.42	28.25
BrianTeam	7.04	17.80
Kechze	3.68	12.67
SibHeteroG	1.31	-4.90
AIKHomoG	2.48	N.A.

5 Conclusions

Robot soccer is one of the best testbeds for studying a variety of different techniques in the multi-robot domain. In this paper, we propose the application of a Dec-POMDP algorithm for developing team strategies for robot soccer. We implemented the algorithm in the *TeamBots* 2D simulator and compared the results with the previous work. We found that the algorithm is quite suitable for solving robot soccer decision problems since we get positive average scores against teams of different strength and win almost all of the matches. Another contribution of the study is that we investigated different parameters of the proposed algorithm and their effect to the performance of the solution.

One of the most important limitations of this algorithm is the estimation of the fitness of individual chromosomes. Since it is based on repeating the simulation many times, as the fidelity of the simulator increases, the running time of the simulator also increases. Therefore, we need to deal with a trade-off between the running time, and the accuracy.

In future work, we plan to develop a better fitness evaluation method and experiment with it in the RoboCup 2D simulator. Our ultimate future plan is to implement and experiment this algorithm in the RoboCup 3D simulator and use it in the RoboCup Standard Platform League.

Acknowledgments. This study was supported by Boğaziçi University Research Fund project 09M105.

References

[1] Eker, B.: Evolutionary Algorithms for Solving DEC-POMDP Problems. PhD thesis, Boğaziçi University (2012)

[2] Bernstein, D.S., Hansen, E.A., Zilberstein, S.: Bounded Policy Iteration for Decentralized POMDPs. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, pp. 1287–1292 (2005)

[3] Balch, T.: Teambots mobile robot simulator (2000)

[4] Meriçli, Ç., Meriçli, T., Levent Akın, H.: A Reward Function Generation Method Using Genetic Algorithms: A Robot Soccer Case Study. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2010, Richland, SC, vol. 1, pp. 1513–1514 (2010); International Foundation for Autonomous Agents and Multiagent Systems

[5] Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The Complexity of Decentralized Control of Markov Decision Processes. Math. Oper. Res. 27, 819–840 (2002)
[6] Wu, F., Chen, X.: Solving Large-Scale and Sparse-Reward DEC-POMDPs with Correlation-MDPs. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007. LNCS (LNAI), vol. 5001, pp. 208–219. Springer, Heidelberg (2008)
[7] Stone, P., Sutton, R.S.: Scaling Reinforcement Learning toward RoboCup Soccer. In: Proc. 18th International Conf. on Machine Learning, pp. 537–544. Morgan Kaufmann, San Francisco (2001)
[8] Stone, P., Sutton, R.S., Singh, S.: Reinforcement Learning for 3 vs. 2 Keepaway. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 249–258. Springer, Heidelberg (2001)
[9] Stone, P., Sutton, R.S., Singh, S.: Reinforcement Learning for 3 vs. 2 Keepaway. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 249–258. Springer, Heidelberg (2001)
[10] Whiteson, S., Kohl, N., Miikkulainen, R., Stone, P.: Evolving Soccer Keepaway Players Through Task Decomposition. Machine Learning 59, 5–30 (2005), 10.1007/s10994-005-0460-9
[11] Stone, P., Kuhlmann, G., Taylor, M.E., Liu, Y.: Keepaway Soccer: From Machine Learning Testbed to Benchmark. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 93–105. Springer, Heidelberg (2006)
[12] Pietro, A.D., While, L., Barone, L.: Learning In RoboCup Keepaway Using Evolutionary Algorithms. In: GECCO 2002, pp. 1065–1072 (2002)
[13] Amato, C., Bernstein, D.S., Zilberstein, S.: Optimal Fixed-Size Controllers for Decentralized POMDPs. In: Proceedings of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains, Hakodate, Japan, pp. 61–71 (2006)
[14] Levent Akın, H.: Evolutionary Computation: A Natural Answer to Artificial Questions. In: Proceedings of ANNAL: Hints from Life to Artificial Intelligence, pp. 41–52. METU, Ankara (1994)
[15] Eker, B., Levent Akın, H.: Using evolution strategies to solve DEC-POMDP problems. Soft Computing-A Fusion of Foundations, Methodologies and Applications 14(1), 35–47 (2010)
[16] Meffert, K., Meseguer, J., Marti, E.D., Meskauskas, A., Vos, J., Rotstan, N.: Jgap: Java genetic algorithms package (2011)
[17] Meriçli, Ç., Levent Akın, H.: A Layered Metric Definition and Evaluation Framework for Multirobot Systems. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 568–579. Springer, Heidelberg (2009)

Towards a Principled Solution to Simulated Robot Soccer

Aijun Bai, Feng Wu, and Xiaoping Chen

Department of Computer Science,
University of Science and Technology of China
{baj,wufeng}@mail.ustc.edu.cn, xpchen@ustc.edu.cn

Abstract. The RoboCup soccer simulation 2D domain is a very large testbed for the research of planning and machine learning. It has competed in the annual world championship tournaments in the past 15 years. However it is still unclear that whether more principled techniques such as decision-theoretic planning take an important role in the success for a RoboCup 2D team. In this paper, we present a novel approach based on MAXQ-OP to automated planning in the RoboCup 2D domain. It combines the benefits of a general hierarchical structure based on MAXQ value function decomposition with the power of heuristic and approximate techniques. The proposed framework provides a principled solution to programming autonomous agents in large stochastic domains. The MAXQ-OP framework has been implemented in our RoboCup 2D team, WrightEagle. The empirical results indicated that the agents developed with this framework and related techniques reached outstanding performances, showing its potential of scalability to very large domains.

Keywords: RoboCup, Soccer Simulation 2D, MAXQ-OP.

1 Introduction

As one of oldest leagues in RoboCup, soccer simulation 2D has achieved great successes and inspired many researchers all over the world to engage themselves in this game each year [5]. Hundreds of research articles based on RoboCup 2D have been published.[1] Comparing to other leagues in RoboCup, the key feature of RoboCup 2D is the abstraction made, which relieves the researchers from having to handle low-level robot problems such as object recognition, communications, and hardware issues. The abstraction enables researchers to focus on high-level functions such as cooperation and learning. The key challenge of RoboCup 2D lies in the fact that it is a fully distributed, multi-agent stochastic domain with continuous state, action and observation space [8].

Stone *et al.* [9] have done a lot of work on applying reinforcement learning methods to RoboCup 2D. Their approaches learn high-level decisions in a keep-away subtask using episodic SMDP Sarsa(λ) with linear tile-coding function

[1] http://www.cs.utexas.edu/~pstone/tmp/sim-league-research.pdf

approximation. More precisely, their robots learn individually when to hold the ball and when to pass it to a teammate. Most recently, they extended their work to a more general task named half field offense [6]. On the same reinforcement learning track, Riedmiller *et al.* [7] have developed several effective techniques for learning mainly low-level skills in RoboCup 2D.

In this paper, we present an alternative approach based on MAXQ-OP [1] to automated planning in the RoboCup 2D domain. It combines the main advantages of online planning and hierarchical decomposition, namely MAXQ. The proposed framework provides a principled solution to programming autonomous agents in large stochastic domains. The key contribution of this paper lies in the overall framework for exploiting the hierarchical structure online and the approximation made for computing the *completion function*. The MAXQ-OP framework has been implemented in our team WrightEagle, which has been participating in annual competitions of RoboCup since 1999 and have got 3 champions and 4 runners-up of RoboCup in recent 7 years.[2] The empirical results indicated that the agents developed with this framework and the related techniques reached outstanding performances, showing its potential of scalability to very large domains.

The remainder of this paper is organized as follows. Section 2 introduces some background knowledge. Section 3 describes the MAXQ-OP framework in detail. Section 4 presents the implementation details in the RoboCup 2D domain, and Section 5 shows the empirical evaluation results. Finally, Section 6 concludes the paper with some discussion of future work.

2 Background

In this section, we briefly introduce the background, namely RoboCup 2D and the MAXQ hierarchical decomposition methods. We assume that readers already have sufficient knowledge on RoboCup 2D. For MAXQ, we only describe some basic concepts but refer [4] for more details.

2.1 RoboCup Soccer Simulation 2D

In RoboCup 2D, a central *server* simulates a 2-dimensional virtual soccer field in real-time. Two teams of fully autonomous agents connect to the server via network sockets to play a soccer game over 6000 steps. A team can have up to 12 clients including 11 players (10 fielders plus 1 goalie) and a coach. Each client interacts independently with the server by 1) receiving a set of observations; 2) making a decision; and 3) sending actions back to the server. Observations for each player only contain noisy and local geometric information such as the distance and angle to other players, ball, and field markings within its view range. Actions are atomic commands such as turning the body or neck to an angle, dashing in a given direction with certain power, kicking the ball to an angle with specified power, or slide tackling the ball.

[2] Team website: `http://www.wrighteagle.org/2d`

2.2 MAXQ Hierarchical Decomposition

Markov decision processes (MDPs) have been proved to be a useful model for planning under uncertainty. In this paper, we concentrate on undiscounted goal-directed MDPs (also known as *stochastic shortest path problems*). It is shown that any MDP can be transformed into an equivalent undiscounted negative goal-directed MDP where the reward for non-goal states is strictly negative [2]. So undiscounted goal-directed MDP is actually a general formulation.

The MAXQ technique decomposes a given MDP M into a set of sub-MDPs arranged over a hierarchical structure, denoted by $\{M_0, M_1, \cdots, M_n\}$. Each sub-MDP is treated as a distinct subtask. Specifically, M_0 is the root subtask which means solving M_0 solves the original MDP M. An *unparameterized* subtask M_i is defined as a tuple $\langle T_i, A_i, \tilde{R}_i \rangle$, where:

- T_i is the *termination predicate* that defines a set of active states S_i, and a set of terminal states G_i for subtask M_i.
- A_i is a set of actions that can be performed to achieve subtask M_i, which can either be primitive actions from M, or refer to other subtasks.
- $\tilde{R}_i$ is the optional *pseudo-reward function* which specifies pseudo-rewards for transitions from active states S_i to terminal states G_i.

It is worth pointing out that if a subtask has task parameters, then different binding of the parameters, may specify different instances of a subtask. Primitive actions are treated as primitive subtasks such that they are always executable, and will terminate immediately after execution.

Given the hierarchical structure, a *hierarchical policy* π is defined as a set of policies for each subtask $\pi = \{\pi_0, \pi_1, \cdots, \pi_n\}$, where π_i is a mapping from active states to actions $\pi_i : S_i \to A_i$. The *projected value function* of policy π for subtask M_i in state s, $V^\pi(i, s)$, is defined as the expected value after following policy π at state s until the subtask M_i terminates at one of its terminal states in G_i. Similarly, $Q^\pi(i, s, a)$ is the expected value by firstly performing action M_a at state s, and then following policy π until the termination of M_i. It is worth noting that $V^\pi(a, s) = R(s, a)$ if M_a is a primitive action $a \in A$.

Dietterich [4] has shown that a *recursively optimal policy* π^* can be found by recursively computing the optimal projected value function as:

$$Q^*(i, s, a) = V^*(a, s) + C^*(i, s, a), \tag{1}$$

where

$$V^*(i, s) = \begin{cases} R(s, i) & \text{if } M_i \text{ is primitive} \\ \max_{a \in A_i} Q^*(i, s, a) & \text{otherwise} \end{cases}, \tag{2}$$

and $C^*(i, s, a)$ is the *completion function* fot optimal policy π^* that estimates the cumulative reward received with the execution of (macro-) action M_a before completing the subtask M_i, as defined below:

$$C^*(i, s, a) = \sum_{s', N} \gamma^N P(s', N | s, a) V^*(i, s'), \tag{3}$$

where $P(s', N|s, a)$ is the probability that subtask M_a at s terminates at state s' after N steps.

3 Online Planning with MAXQ

In this section, we explain in detail how our MAXQ-OP solution works. As mentioned above, MAXQ-OP is a novel online planning approach that incorporates the power of the MAXQ decomposition to efficiently solve large MDPs.

3.1 Overview of MAXQ-OP

In general, online planning interleaves planning with execution and chooses the best action for the current step. Given the MAXQ hierarchy of an MDP, $M = \{M_0, M_1, \cdots, M_n\}$, the main procedure of MAXQ-OP evaluates each subtask by forward search to compute the recursive value functions $V^*(i, s)$ and $Q^*(i, s, a)$ online. This involves a complete search of all paths through the MAXQ hierarchy starting from the root task M_0 and ending with some primitive subtasks at the leaf nodes. After the search process, the best action $a \in A_0$ is chosen for the root task based on the recursive Q function. Meanwhile, the best primitive action $a_p \in A$ that should be performed first is also determined. This action a_p will be executed to the environment, leading to a transition of the system state. Then, the planning procedure starts over to select the best action for the next step.

Algorithm 1. `OnlinePlanning()`

```
Input: an MDP model with its MAXQ hierarchical structure
Output: the accumulated reward $r$ after reaching a goal
1 $r \leftarrow 0$;
2 $s \leftarrow$ GetInitState();
3 while $s \notin G_0$ do
4     $\langle v, a_p \rangle \leftarrow$ EvaluateState($0, s, [0, 0, \cdots, 0]$);
5     $r \leftarrow r +$ ExecuteAction($a_p, s$);
6     $s \leftarrow$ GetNextState();
7 return $r$;
```

As shown in Algorithm 1, state s is initialized by `GetInitState` and the function `GetNextState` returns the next state of the environment after `ExecuteAction` is performed. It executes a primitive action to the environment and returns a reward for running that action. The main process loops over until a goal state in G_0 is reached. Obviously, the key procedure of MAXQ-OP is `EvaluateState`, which evaluates each subtask by depth-first search and returns the best action for the current state. Section 3.2 will explain `EvaluateState` in more detail.

3.2 Task Evaluation over Hierarchy

In order to choose the best action, the agent must compute a Q function for each possible action at the current state s. Typically, this will form a search tree starting from s and ending with the goal states. The search tree is also known as an AND-OR tree where the AND nodes are actions and the OR nodes are states. The root node of such an AND-OR tree represents the current state. The search in the tree is processed in a depth-first fashion until a goal state or a certain pre-determined fixed depth is reached. When it reaches the depth, a heuristic is often used to evaluate the long term value of the state at the leaf node.

Algorithm 2. `EvaluateState(`i, s, d`)`

```
Input: subtask $M_i$, state $s$ and depth array $d$
Output: $\langle V^*(i,s),$ a primitive action $a_p^*\rangle$
1  if $M_i$ is primitive then return $\langle R(s, M_i), M_i\rangle$;
2  else if $s \notin S_i$ and $s \notin G_i$ then return $\langle -\infty, nil\rangle$;
3  else if $s \in G_i$ then return $\langle 0, nil\rangle$;
4  else if $d[i] \geq D[i]$ then return $\langle$HeuristicValue($i, s$)$, nil\rangle$;
5  else
6      $\langle v^*, a_p^*\rangle \leftarrow \langle -\infty, nil\rangle$;
7      for $M_k \in$ Subtasks($M_i$) do
8          if $M_k$ is primitive or $s \notin G_k$ then
9              $\langle v, a_p\rangle \leftarrow$ EvaluateState($k, s, d$);
10             $v \leftarrow v+$ EvaluateCompletion($i, s, k, d$);
11             if $v > v^*$ then
12                 $\langle v^*, a_p^*\rangle \leftarrow \langle v, a_p\rangle$;
13     return $\langle v^*, a_p^*\rangle$;
```

When the task hierarchy is given, it is more difficult to perform such a search procedure since each subtask may contain other subtasks or several primitive actions. As shown in Algorithm 2, the search starts with the root task M_i and the current state s. Then, the node of the current state s is expanded by trying each possible subtask of M_i. This involves a recursive evaluation of the subtasks and the subtask with the highest value is selected. As mentioned in Section 2, the evaluation of a subtask requires the computation of the value function for its children and the completion function. The value function can be computed recursively. Therefore, the key challenge is to calculate the completion function.

Intuitively, the completion function represents the optimal value of fulfilling the task M_i after executing a subtask M_a first. According to Equation 3, the completion function of an optimal policy π^* can be written as:

$$C^*(i, s, a) = \sum_{s', N} \gamma^N P(s', N|s, a) V^{\pi^*}(i, s'), \tag{4}$$

Algorithm 3. `EvaluateCompletion`(i, s, a, d)

```
Input: subtask $M_i$, state $s$, action $M_a$ and depth array $d$
Output: estimated $C^*(i, s, a)$
1 $\tilde{G}_a \leftarrow$ ImportanceSampling($G_a$, $D_a$);
2 $v \leftarrow 0$;
3 for $s' \in \tilde{G}_a$ do
4     $d' \leftarrow d$;
5     $d'[i] \leftarrow d'[i] + 1$;
6     $v \leftarrow v + \frac{1}{|\tilde{G}_a|}$ EvaluateState($i, s', d'$);
7 return $v$;
```

where

$$P(s', N|s, a) = \sum_{\langle s,s_1,\ldots,s_{N-1}\rangle} P(s_1|s, \pi_a^*(s)) \cdot P(s_2|s_1, \pi_a^*(s_1)) \cdots P(s'|s_{N-1}, \pi_a^*(s_{N-1})). \tag{5}$$

More precisely, $\langle s, s_1, \ldots, s_{N-1}\rangle$ is a path from the state s to the terminal state s' by following the optimal policy $\pi_a^* \in \pi^*$. It is worth noticing that π^* is a recursive policy constructed by other subtasks. Obviously, computing the optimal policy π^* is equivalent to solving the entire problem. In principle, we can exhaustively expand the search tree and enumerate all possible state-action sequences starting with s, a and ending with s' to identify the optimal path. Obviously, this may be inapplicable for large domains. In Section 3.3, we will present a more efficient way to approximate the completion function.

Algorithm 2 summarizes the major procedures of evaluating a subtask. Clearly, the recursion will end when: 1) the subtask is a primitive action; 2) the state is a goal state or a state outside the scope of this subtask; or 3) a certain depth is reached, i.e. $d[i] \geq D[i]$ where $d[i]$ is the current forward search depth and $D[i]$ is the maximal depth allowed for subtask M_i. It is worth pointing out, different maximal depths are allowed for each subtask. Higher level subtasks may have smaller maximal depth in practice. If the subtask is a primitive action, the immediate reward will be returned as well as the action itself. If the search reaches a certain depth, it also returns with a heuristic value for the long-term reward. In this case, a *nil* action is also returned, but it will never be chosen by higher level subtasks. If none of the above conditions holds, it will loop over and evaluate all the children of this subtask recursively.

3.3 Completion Function Approximation

To exactly compute the optimal completion function, the agent must know the optimal policy π^* first which is equivalent to solving the entire problem. However, it is intractable to find the optimal policy online due to the time constraint. When applying MAXQ-OP to large problems, approximation should be made to compute the completion function for each subtask. One possible solution is to calculate an approximate policy offline and then to use it for the online computation of the completion function. However, it may be also challenging to find a good approximation of the optimal policy if the domain is very large.

Notice that the term γ^N in Equation 3 is equal to 1 when $\gamma = 1$, which is the default setting of this paper. Given an optimal policy, the subtask will terminate at a certain goal state with the probability of 1 after several steps. To compute the completion function, the only term need to be considered is $P(s', N|s, a)$–a distribution over the terminal states. Given a subtask, it is often possible to directly approximate the distribution disregarding the detail of execution.

Based on these observations, we assume that each subtask M_i will terminate at its terminal states in G_i with a prior distribution of D_i. In principle, D_i can be any probability distribution associated with each subtask. Denoted by $\tilde{G}_a$ a set of sampled states drawn from prior distribution D_a using *importance sampling* [10] techniques, the completion function $C^*(i, s, a)$ can be approximated as:

$$C^*(i, s, a) \approx \frac{1}{|\tilde{G}_a|} \sum_{s' \in \tilde{G}_a} V^*(i, s'). \tag{6}$$

A recursive procedure is proposed to estimate the completion function, as shown in Algorithm 3. In practice, the prior distribution D_a–a key distribution when computing the completion function, can be improved by considering the domain knowledge. Take the robot soccer domain for example. The agent at state s may locate in a certain position of the field. Suppose s' is the goal state of successfully scoring the ball. Then, the agent may have higher probability to reach s' if it directly dribbles the ball to the goal or passes the ball to some teammates who is near the goal, which is specified by the action a in the model.

3.4 Heuristic Search in Action Space

For some domains with large action space, it may be very time-consuming to enumerate all possible actions exhaustively. Hence it is necessary to introduce some heuristic techniques (including prune strategies) to speed up the search process. Intuitively, there is no need to evaluate those actions that are not likely to be better. In MAXQ-OP, this is done by implementing a iterative version of `Subtasks` function which dynamically selects the most promising action to be evaluated next with the tradeoff between exploitation and exploration. Different heuristic techniques can be used for different subtasks, such as A*, hill-climbing, gradient ascent, etc. The discussion of the heuristic techniques is beyond the scope of this paper, and the space lacks for a detailed description of it.

4 Implementation in RoboCup 2D

It is our long-term effort to apply the MAXQ-OP framework to the RoboCup 2D domain. In this section, we present the implementation details of the MAXQ-OP framework in WrightEagle.

4.1 RoboCup 2D as an MDP

In this section, we present the technical details on modeling the RoboCup 2D domain as an MDP. As mentioned, it is a partially-observable multi-agent domain with continuous state and action space. To model it as a fully-observable single-agent MDP, we specify the state and action spaces and the transition and reward functions as follows:

State Space. We treat teammates and opponents as part of the environment and try to estimate the current state with sequences of observations. Then, the state of the 2D domain can be represented as a fixed-length vector, containing state variables that totally cover 23 distinct objects (10 teammates, 11 opponents, the ball, and the agent itself).

Action Space. All primitive actions, like dash, kick, tackle, turn and turn_neck, are originally defined by the 2D domain. They all have continuous parameters, resulting a continuous action space.

Transition Function. Considering the fact that autonomous teammates and opponents make the environment unpredictable, the transition function is not obvious to represent. In our team, the agent assumes that all other players share a same predefined behavior model: they will execute a random kick if the ball is kickable for them, or a random walk otherwise. For primitive actions, the underlying transition model for each atomic command is fully determined by the server as a set of *generative* models.

Reward Function. The underlying reward function has a *sparse* property: the agent usually earns zero rewards for thousands of steps before ball scored or conceded, may causing that the forward search process often terminate without any rewards obtained, and thus can not tell the differences between subtasks. In our team, to emphasize each subtask's characteristic and to guarantee that positive results can be found by the search process, a set of pseudo-reward functions is developed for each subtask.

To estimate the size of the state space, we ignore some secondary variables for simplification (such as heterogeneous parameters and stamina information). Totally 4 variables are needed to represent the ball's state including position (x, y) and velocity (v_x, v_y). In addition with (x, y) and (v_x, v_y), two more variables are used to represent each player's state including body direction d_b, and neck direction d_n. Therefore the full state vector has a dimensionality of 136. All these state variables have continuous values, resulting a high-dimensional continuous state space. If we discretize each state variable into 10^3 uniformly distributed values in its own field of definitions, then we obtain a simplified state space with 10^{408} states, which is extremely larger than domains usually studied in the literature.

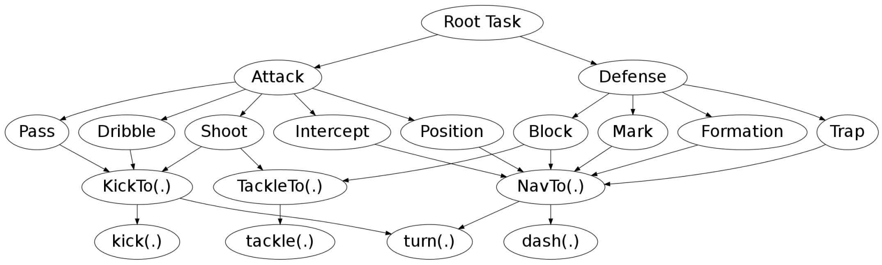


Fig. 1. MAXQ task graph for WrightEagle

4.2 Solution with MAXQ-OP

In this section, we describe how to apply MAXQ-OP to the RoboCup soccer simulation domain. Firstly, a series of subtasks at different levels are defined as the building blocks of constructing the MAXQ hierarchy, listed as follows:

– kick, turn, dash, and tackle: They are low-level parameterized primitive actions originally defined by the soccer server. A reward of -1 is assigned to each primitive action to guarantee that the optimal policy will try to reach a goal as fast as possible.
– KickTo, TackleTo, and NavTo: In the KickTo and TackleTo subtask, the goal is to kick or tackle the ball to a given direction with a specified velocity, while the goal of the NavTo subtask is to move the agent from its current location to a target location.
– Shoot, Dribble, Pass, Position, Intercept, Block, Trap, Mark, and Formation: These subtasks are high-level behaviors in our team where: 1) Shoot is to kick out the ball to score; 2) Dribble is to dribble the ball in an appropriate direction; 3) Pass is to pass the ball to a proper teammate; 4) Position is to maintain the teammate formation for attacking; 5) Intercept is to get the ball as fast as possible; 6) Block is to block the opponent who controls the ball; 7) Trap is to hassle the ball controller and wait to steal the ball; 8) Mark is to mark related opponents; 9) Formation is to maintain formation for defense.
– Attack and Defense: Obviously, the goal of Attack is to attack opponents to score while the goal of Defense is to defense against opponents.
– Root: This is the root task. It firstly evaluate the Attack subtask to see whether it is ready to attack, otherwise it will try the Defense subtask.

The graphical representation of the MAXQ hierarchical structure is shown in Figure 1, where a parenthesis after a subtask's name indicates this subtask will take parameters. It is worth noting that state abstractions are implicitly introduced by this hierarchy. For example in the NavTo subtask, only the agent's own state variables are relevant. It is irrelevant for the KickTo and TackleTo subtasks to consider those state variables describing other players' states. To deal with the large action space, heuristic methods are critical when applying MAXQ-OP. There are many possible candidates depending on the characteristic of subtasks.

For instance, hill-climbing is used when searching over the action space of KickTo for the Pass subtask and A* search is used when searching over the action space of dash and turn for the NavTo subtask.

As mentioned earlier, the method for approximating the completion function is crucial for the performance when implementing MAXQ-OP. In RoboCup 2D, it is more challenging to compute the distribution because: 1) the forward search process is unable to run into an sufficient depth due to the online time constraint; and 2) the future states are difficult to predict due to the uncertainty of the environment, especially the unknown behaviors of the opponent team. To estimate the distribution of reaching a goal, we used a variety techniques for different subtasks based on the domain knowledge. Take the Attack subtask for example. A so-called *impelling speed* is used to approximate the completion probability. It is formally defined as:

$$impelling\_speed(s, s', \alpha) = \frac{dist(s, s', \alpha) + pre\_dist(s', \alpha)}{step(s, s') + pre\_step(s')}, \tag{7}$$

where α is a given direction (called aim-angle), $dist(s, s', \alpha)$ is the ball's running distance in direction α from state s to state s', $step(s, s')$ is the estimated steps from state s to state s', $pre\_dist(s')$ estimates final distance in direction α that the ball can be impelled forward starting from state s', and $pre\_step(s')$ estimates the respective steps. The aim-angle in state s is determined dynamically by $aim\_angle(s)$ function. The value of $impelling\_speed(s, s', aim\_angle(s))$ indicates the fact that the faster the ball is moved in a right direction, the more attack chance there would be. In practice, it makes the team attack more efficient. As a result, it can make a fast score within tens of steps in the beginning of a match. Different definitions of the aim_angle function can produce substantially different attack styles, leading to a very flexible and adaptive strategy, particularly for unfamiliar teams.

5 Empirical Evaluation

To test how the MAXQ-OP framework affects our team's final performance, we compared three different versions of our team, including:

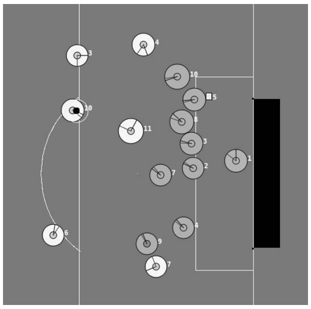


Fig. 2. A selected scene from the final match of RoboCup 2011

– FULL: This is exactly the full version of our team, where a complete MAXQ-OP online planning framework is implemented as the key component.
– RANDOM: This is nearly the same as FULL, except that when the ball is kickable for the agent and the Shoot behavior finds no solution, the Attack behavior randomly chooses a macro-action to perform between Pass and Dribble with uniform probability.
– HAND-CODED: This is similar to RANDOM, but instead of a random selection between Pass and Dribble, a hand-coded strategy is used. With this strategy, if there is no opponent within 3m from the agent, then Dribble is chosen; otherwise, Pass is chosen.

The only difference between FULL, RANDOM and HAND-CODED is the local selection strategy between Pass and Dribble in the Attack behavior. In FULL, this selection is automatically based on the value function of subtasks (i.e. the solutions found by `EvaluateState(Pass, ·, ·)` and `EvaluateState(Dribble, ·, ·)` in the MAXQ-OP framework). Although RANDOM and HAND-CODED have different Pass-Dribble selection strategies, the other subtasks of Attack, including Shoot, Pass, Dribble, and Intercept, as that of FULL, remain the same.

For each version, we use an offline coach (also known as a trainer) to independently run the team against the Helios11 binary (which has participated in RoboCup 2011 and won the second place) for 100 episodes. Each episode begins with a fixed scene (i.e. the full state vector) taken from the final match we have participated in of RoboCup 2011, and ends when: 1) our team scores a goal, denoted by **success**; or 2) the ball's x coordination is smaller than -10, denoted by **failure**; or 3) the episode lasts longer than 200 cycles, denoted by **timeout**. It is worth mentioning that although all of the episode begin with the same scene, none of them is identical due to the uncertainty of the environment.

The selected scene, which is originally located at cycle #3142 of that match, is depicted in Figure 2 where white circles represent our players, gray ones represent opponents, and the small black one represents the ball. We can see that our player 10 was holding the ball at that moment, while 9 opponents (including goalie) were blocking just in front of their goal area. In RoboCup 2011, teammate 10 passed the ball directly to teammate 11. Having got the ball, teammate 11 decided to pass the ball back to teammate 10. When teammate 11 had moved to an appropriate position, the ball was passed again to it. Finally, teammate 11 executed a tackle to shoot at cycle #3158 and scored a goal 5 cycles later.

Table 1 summarizes the test results showing that the FULL version of our team outperforms both RANDOM and HAND-CODED with an increase of the chance of **sucess** by 86.7% and 64.7% respectively. We find that although FULL, RANDOM and HAND-CODED have the same hierarchical structure and subtasks of Attack, the local selection strategy between Pass and Dribble plays a key role in the decision of Attack and affects the final performance substantially. It can be seen from the table that MAXQ-OP based local selection strategy between Pass and Dribble is sufficient for the Attack behavior to achieve a high performance. Recursively, this is also true for other subtasks over the MAXQ hierarchy,

Table 1. Empirical results of WrightEagle in episodic scene test

Version	Episodes	Success	Failure	Timeout
Full	100	28	31	41
Random	100	15	44	41
Hand-coded	100	17	38	45

Table 2. Empirical results of WrightEagle in full game test

Opponent Team	Games	Avg. Goals	Avg. Points	Winning Rate
BrainsStomers08	100	3.09 : 0.82	2.59 : 0.28	$82.0 \pm 7.5\%$
Helios10	100	4.30 : 0.88	2.84 : 0.11	$93.0 \pm 5.0\%$
Helios11	100	3.04 : 1.33	2.33 : 0.52	$72.0 \pm 8.8\%$
Oxsy11	100	4.97 : 1.33	2.79 : 0.16	$91.0 \pm 5.6\%$

such as Defense, Shoot, Pass, etc. To conclude, MAXQ-OP is able to be the key to success of our team in this episodic scene test.

We also tested the Full version of our team in full games against 4 best RoboCup 2D opponent teams, namely BrainsStomers08, Helios10, Helios11 and Oxsy11, where BrainStormers08 and Helios10 were the champion of RoboCup 2008 and RoboCup 2010 respectively. In the experiments, we independently ran our team against the binary codes officially released by them for 100 games on exactly the same hardware. Table 2 summarizes the detailed empirical results with our winning rate, which is defined as $p = n/N$, where n is the number of games we won, and N is the total number of games. It can be seen from the table that our team with the implementation of MAXQ-OP substantially outperforms other tested teams. Specifically, our team had about 82.0%, 93.0%, 72.0% and 91.0% of the chances to win BrainsStomers08, Helios10, Helios11 and Oxsy11 respectively.

While there are multiple factors contributing to the general performance of a RoboCup 2D team, it is our observation that our team benefits greatly from the abstraction we made for the actions and states. The key advantage of MAXQ-OP in our team is to provide a formal framework for conducting the search process over a task hierarchy. Therefore, the team can search for a strategy-level solution automatically online by given the pre-defined task hierarchy. To the best of our knowledge, most of the current RoboCup teams develop their team based on hand-coded rules and behaviors.

6 Conclusions

This paper presents a novel approach to automated planning in the RoboCup 2D domain. It benefits from both the advantage of hierarchical decomposition and the power of heuristics. Barry *et al.* proposed an *offline* algorithm called DetH* [3] to solve large MDPs hierarchically by assuming that the transitions between macro-states are totally deterministic. In contrast, we assume a prior distribution over the terminal states of each subtask, which is more realistic. The MAXQ-OP framework has been implemented in the team WrightEagle. The empirical results indicated that the agents developed with this framework and the related techniques reached outstanding performances, showing its potential of scalability to very large domains. This demonstrates the soundness and stability of MAXQ-OP for solving large MDPs with the pre-defined task hierarchy. In the future, we plan to theoretically analyze MAXQ-OP with different task priors and try to generate these priors automatically.

Acknowledgments. This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundation of China under grant 60745002 and 61175057. The authors thank Haochong Zhang, Guanghui Lu, and Miao Jiang for their contributions to this work. We are also grateful to the anonymous reviewers for their constructive comments and suggestions.

References

1. Bai, A., Wu, F., Chen, X.: Online planning for large MDPs with MAXQ decomposition (extended abstract). In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems, Valencia, Spain (June 2012)
2. Barry, J.: Fast Approximate Hierarchical Solution of MDPs. Ph.D. thesis, Massachusetts Institute of Technology (2009)
3. Barry, J., Kaelbling, L., Lozano-Perez, T.: Deth*: Approximate hierarchical solution of large markov decision processes. In: International Joint Conference on Artificial Intelligence, pp. 1928–1935 (2011)
4. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Machine Learning Research 13(1), 63 (May 1999)
5. Gabel, T., Riedmiller, M.: On progress in roboCup: The simulation league showcase. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS (LNAI), vol. 6556, pp. 36–47. Springer, Heidelberg (2011)
6. Kalyanakrishnan, S., Liu, Y., Stone, P.: Half field offense in roboCup soccer: A multiagent reinforcement learning case study. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 72–85. Springer, Heidelberg (2007)
7. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots 27(1), 55–73 (2009)
8. Stone, P.: Layered learning in multiagent systems: A winning approach to robotic soccer. The MIT press (2000)
9. Stone, P., Sutton, R., Kuhlmann, G.: Reinforcement learning for robocup soccer keepaway. Adaptive Behavior 13(3), 165–188 (2005)
10. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Robust monte carlo localization for mobile robots. Artificial Intelligence 128(1-2), 99–141 (2001)

People Detection in 3*d* Point Clouds Using Local Surface Normals

Frederik Hegger, Nico Hochgeschwender, Gerhard K. Kraetzschmar, and Paul G. Ploeger

Bonn-Rhein-Sieg University of Applied Sciences, Sankt Augustin, Germany
{frederik.hegger,nico.hochgeschwender,gerhard.kraetzschmar, paul.ploeger}@h-brs.de

Abstract. The ability to detect people in domestic and unconstrained environments is crucial for every service robot. The knowledge where people are is required to perform several tasks such as navigation with dynamic obstacle avoidance and human-robot-interaction. In this paper we propose a people detection approach based on 3*d* data provided by a RGB-D camera. We introduce a novel 3*d* feature descriptor based on Local Surface Normals (LSN) which is used to learn a classifier in a supervised machine learning manner. In order to increase the systems flexibility and to detect people even under partial occlusion we introduce a top-down/bottom-up segmentation. We deployed the people detection system on a real-world service robot operating at a reasonable frame rate of $5Hz$. The experimental results show that our approach is able to detect persons in various poses and motions such as sitting, walking, and running.

Keywords: Human-Robot Interaction, People Detection, RGB-D.

1 Introduction

Domestic service robots such as the Care-O-bot 3 [5] and PR2 [3] are deployed more and more in realistic, unconstrained, and unknown environments such as offices and households. In contrast to artificial environments the real-world is populated with humans which are walking, sitting, and running. In order to interact in such environments with humans in a safe manner a service robot must be aware of their positions, movements, and actions. Therefore, a robust people detection system is crucial for every domestic service robot. An appropriate sensor type providing perceptual information about the environment is required to detect people in a robust and reliable manner. Quite recently, a new type of cameras has been become available, namely *RGB-D* cameras such as the *Microsoft Kinect*[1] and *Asus Xtion*[2]. Those cameras provide a 3*d* point cloud and additional RGB values at the same time. The low-cost and the high frequency of

[1] www.xbox.com/kinect
[2] www.asus.com

$30Hz$ makes the *Kinect* very attractive for the robotics community. Hence, the people detection approach described in this paper proposes to use and processes RGB-D data provided by a *Kinect* sensor. Previous contributions in the field of people detection are based on range data provided by laser range finders. In [1] Arras et al. proposed a system which segments a complete scan into smaller clusters, extracting geometric features, and then classifies the cluster in human or non-human. The classifier has been created in a supervised machine learning manner with methods such as AdaBoost and SVM. This principle has been extended in [7] by mounting two additional laser range finders on different height in order to retrieve a more sophisticated view on the scene. Spinello et al. [11] extended this idea by extracting a fixed set of $2d$ vertical scan lines from a full $3d$ point cloud. The detection is performed in each layer separately. The layers are later fused with a probabilistic voting scheme. Other approaches are based on vision as a primary modality. They apply well-known techniques such as implicit shape models [13], haar-like features [14], or histogram of oriented gradients [12] for feature extraction. However, all these approaches operate only in $2d$ space. First approaches operating in $3d$ are described by Satake et al. [9] where template matching (depth templates) is used to detect the upper body of humans. In [2] and [8] the $3d$ point cloud is first reduced to a $2.5d$ map in order to keep the computational effort low. The classification itself is again based on different $2d$ features and a machine learning classifier. The approach which comes close to our approach has been introduced by Spinello and Arras [10]. In a hybrid manner the detection is based on a combination of $3d$ depth and $2d$ image data. Inspired from a histogram of oriented gradients (HOG) detector Spinello proposes a novel histogram of oriented depths (HOD) for the detection in $3d$. Both information (HOG and HOD) are fused which yields in a robust and later GPU-optimized people detection system.

In contrast to [10], our approach uses only the $3d$ point cloud provided by a Microsoft Kinect camera. The data is split into smaller clusters using a layered sub-division of the scene and a top-down/bottom-up segmentation technique. A random forest classifier is used to label the resulting $3d$ clusters either as human or non-human. Inspired from [6], we extended the idea of using local surface normals (LSN) and composed a new feature vector based on a histogram of local surface normals plus additional $2d$ and $3d$ statistical features. An overview of the complete processing pipeline is depicted in Figure 1. The major contribution of our approach is a novel feature descriptor based on local surface normals and the capability to robustly detect persons in various poses/motions, even if they are partially occluded like sitting behind a table or desk.

2 People Detection Using Local Surface Normals

In this section we introduce our $3d$ people detection approach using Local Surface Normals (LSN). The approach consists of four phases as shown in Figure 1, namely *Preprocessing*, *Top-Down Segmentation*, *Classification*, and *Bottom-Up Segmentation*.

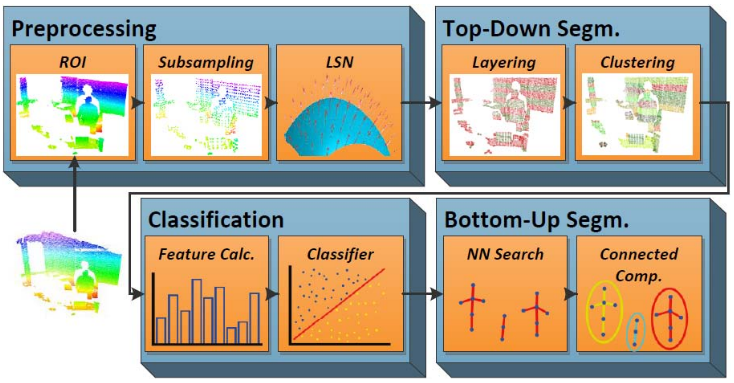


Fig. 1. The processing pipeline is divided into four phases (*blue boxes*). Each phase consists of several sub-components (*orange boxes*) which perform the actual computation of the input data.

2.1 Preprocessing

A single point cloud from the Kinect sensor consists of $\approx$ 300.000 points. In order to keep the overall processing time reasonable we carefully reduced the raw input data in the preprocessing phase.

Region of Interest (ROI). A major disadvantage of the Kinect camera is the increasing depth discretization error for large distances. Beyond $5m$ the depth values are very noisy and shaky. Therefore, the ROI is defined as $0.5m <= depth <= 5.0m$ and $0.0m <= height <= 2.0m$. The height has been choosen because people usually appear in this range. The ROI steps already reduces (depending on the actual scene) the point cloud to $\approx$ 110.000 points in average.

Subsampling. The remaining points provided by the ROI step are further reduced by a subsampling routine to make the point cloud more sparse, i.e. a $3d$ grid with a predefined cell size is overlayed over the full point cloud. The points inside each box are merged to a single new point. An increased cell size will yield to a sparse point cloud. We have used a cell size of $3cm$ x $3cm$ x $3cm$ which still maintains the desired accuracy for the normal estimation and simultaneously reduces the point cloud to $\approx$ 16.000 points in average.

Local Surface Normals (LSN). In the classification phase (see Section 2.3) we propose a feature vector which consists of a histogram of local surface normals. A local surface normal is computed through fitting a plane to the k-nearest neighbors of the target point. A more detailed description of the algorithm can be found in [6]. Before the preprocessed point cloud is forwarded to the segmentation phase, for all remaining points the local surface normals are computed. In case the normals would be calculated after the segmentation, the accuracy of the normal estimation for those points which lie on the border of a cluster would be significant lower. A reasonable part of the neighborhood might already belong to another cluster.

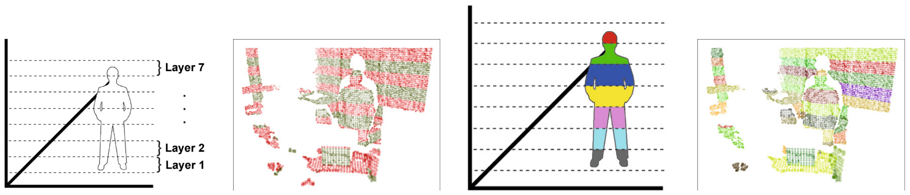
2.2 Top-Down Segmentation

The segmentation of large 3*d* point clouds is a (computational) costly and complex exercise. Segmentation approaches such as region growing or graph-based approaches are known to have a huge computational complexity. Therefore, such approaches are not feasible in robotics where reasonable performance is crucial.

Layering. We propose a basic top-down segmentation technique (see Figure 2). The general idea is decompose the point cloud into a fixed set of different 3*d* height layers and then start to segment each layer separately in smaller clusters. In detail, the layering and segmentation algorithm can be explained as follows: Let $\mathbf{P} = \{p_1, ..., p_N\}$ be a point cloud with $p_i = (x, y, z)$ and N which is equal to the number of points in the point cloud. Then $\mathbf{P}$ is split into a fixed number of 3*d* layers $\mathbf{L} = \{l_1, ..., l_M\}$ with

$$M = \frac{(Z_{max} - Z_{max})}{SH}$$

where Z_{min} and Z_{max} are the minimum and maximum height values of the predefined ROI and SH is the desired slice height. For each layer l_j the minimum and maximum height is calculated. For instance, assuming a predefined slice height of $20cm$ then the first layer l_1 contains only points with $0.0m <= p_i(z) <= 0.2m$. The remaining layers $l_2, ...l_M$ will be established according to this principle. As experimentally validated we consider a slice height of $25cm$ as optimal (see Section 3).



(a) schematic layering (b) layering result (c) schematic segmentation (d) segmentation result

Fig. 2. Images *(a)* and *(b)* show the layering process, where each point cloud is divided into a set of 3*d* layers according to a manually defined slice height. For the layering, we have applied a slice height of $25cm$. Each layer is segmented into clusters using a Euclidean Clustering approach (see Image *(c)* and *(d)*). The different colored points indicate either the different height layers or the segmented 3*d* clusters.

Clustering. The actual segmentation generates for each layer l_j a sequence of small clusters $\mathbf{C} = \{c_1, ..., c_O\}$, where each cluster $c_{j,k}$ contains a subset of points located in l_k. The segmentation applies an Euclidean clustering technique which is less parameterizable. Only a distance threshold $thres_{EuclDist}$ has to be defined which defines whether a target point is added to the cluster or not. Furthermore, $thres_{EuclDist}$ also determines whether there are many small clusters ($thres_{EuclDist} \leftarrow 0$) or only a few large clusters ($thres_{EuclDist} \rightarrow \infty$). As mentioned, we have used a grid-size of $3cm$ for subsampling. According to this dimensions and a certain amount of noise, we set $thres_{EuclDist} = 2 \times grid\_size$ in order to ensure that two persons which stand close to each other are not merged to a single cluster. The proposed fine-grained clustering has the advantage over a clustering *without* prior layering when one object is partially occluded by another object. For instance, if a person is sitting at a table, our approach creates several smaller clusters for both objects. Instead, the pure Euclidean clustering would create a single cluster which consists of a table and the person, because the person is sitting very close to the table or has put the arms on it. Furthermore, the user-defined slice height plays also an important role for the performance of the segmentation. A reasonable small height ends up in really tiny clusters with few local surface normals which are not sufficient for a robust classification. On the other hand, a large slice height creates also large clusters (where two or more objects would get merged to a single cluster) which would alleviate the specific advantage of the proposed segmentation stage.

2.3 Classification of 3d Clusters

The previous segmentation phase produces a list of 3d clusters. In the classification phase we want to assign a label to each cluster (human or non-human). We approached the two-class classification problem with a supervised machine learning technique. We evaluated the performance of three popular machine learner on different datasets recorded in different environments, namely AdaBosst, SVM and Random Forests [4]. The results showed that for all datasets the Random Forest classifier outperforms both other machine learning techniques.

Feature Calculation. As a feature vector for the Random Forest we propose a histogram of local surface normals (HLSN). The use of such a feature vector can be motivated as follows: households and offices contain to a large extend walls, tables, desks, shelfs, and chairs. More precisely, a reasonable part of daily environments consists of horizontal and vertical planes. Whereas the human body has a more cylindrical appearance. With a histogram of LSNs we can express this property to distinguish between human and non-human clusters. We compute a fix-sized histogram over the normals for all points in a cluster which is the input for a feature vector. However, the Random Forests algorithm expects a one dimensional input vector. Therefore, a separate histogram for each normal axis (x, y and z) is established. In addition, the width and the depth of a cluster is added to the feature vector, which helps to decrease the false positive rate.

Classifier. Learning the Random Forest classifier requires a large-set of training samples. As in other fields the collection of positive and negative training samples is a time consuming task, especially when many samples (> 1000) are required and the annotation of each sample has to be done manually. Therefore, we integrated a procedure to capture positive and negative training samples automatically. Negative samples have been collected with a mobile service robot. We established a map of our University building which at least consisted of an office, laboratory, long corridor and an apartment. For each room a navigation goal has been manually annotated. An automatic procedure generated a random order in which the rooms should be visited. The robot started to navigate autonomously through all the environments and simultaneously segmenting each incoming point cloud. Each extracted cluster has been labeled as negative example. During the whole run we ensured that there has been no person in the field-of-view (FOV) of the robot. This process guarantees that the samples are indeed collected in a random manner. The positive samples have been collected with a static mounted Kinect camera. The camera was placed in a laboratory where people are frequently walking and sitting around. We defined a ROI which does not contain any object and consequently provides an empty point cloud. In case the person passed the ROI, the segmentation stage extracted the related clusters and labeled them as positive samples.

2.4 Bottom-Up Segmentation

In the last phase we obtain a sequence of $3d$ clusters which are classified as human. However, those "part-based" detections have to be assembled and associated to one respective person. A graph-based representation based on the cluster's center is created. The advantage is that not the whole data points of a cluster have to be processed which keeps the computational effort low. Each center point is then connected to its two nearest neighbors as long as the Euclidean distance between those points does not exceed a certain threshold. Each cluster has always a maximum height (equal to the predefined slice height) which allows us to derive the threshold, because the center points of two neighboring clusters can only have a maximum distance of $2 \times slice\_height$. When all the points in the queue have been processed the overall graph can be split in its connected components, which builds the actual person detection. Due to false positive detection when classifying the extracted $3d$ clusters, we consider a successful person detection only, if at least three clusters belong to one person (at least $\approx 45cm$ of the persons body must visible).

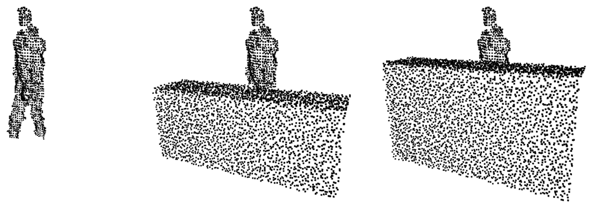
3 Experimental Evaluation

In order to evaluate the proposed people detection system we performed several experiments with different objectives as described below.

3.1 Experiment Objectives

Objective 1. Investigate the impact of the predefined slice height on the classification error.

The segmentation is based on separating the point cloud into several fix-sized layers. The amount of layers depends on the chosen slice height. In this experiment we investigated the impact of the predefined slice height on the resulting classification error. The experiment was executed several times with different slice heights ranging from $10cm$ to $100cm$ (= half of the maximum perceivable height). Every range value below $10cm$ results in very few points which is not sufficient to represent a comprehensive distribution. Thus one requirement for the people detection approach is the ability to detect people even if they are partially occluded. In each experiment the slice height is constantly increased by $5cm$ (when starting at the minimum). A 10-fold cross-validation was applied. In order to evaluate the segmentation behavior against occlusion, synthetic generated occlusion (e.g., a cupboard) was added to the data. The experiment was repeated three times with different amount of occlusion, namely no occlusion, 50%, and 70% (see also Figure 3). Moreover, Gaussian noise was added to the synthetic data in order to achieve approximation to the Kinect data.



(a) No occlusion (b) 50% occlusion (c) 70% occlusion

Fig. 3. Different occlusion levels

Objective 2. Investigate the actual people detection performance.

In order to assess the detection rates under different circumstances we defined two categories, namely poses and motions. For the pose category we evaluated the detection rate for persons sitting on a chair and for persons which where partially occluded (at least 30% of the whole body). For the motion category we evaluated three different natural motions: not moving, random walking, and random running. We executed the experiment with ten subjects in three different environments. In our RoboCup@Home laboratory, a real German living room, and the entrance of our University where people frequently enter and leave the building. The test procedure (or test cases) looked as follows:

1. **Standing pose:** the persons were asked to position themselves in various random positions and usual body postures.
2. **Sitting pose:** the persons were asked to sit down on a chair and position themselves in various random positions and usual sitting postures.
3. **Partially occluded pose:** the persons were asked to stand behind a cupboard of 80 cm height and to move up and down in a natural way.
4. **Not moving motion:** it is identical to the test for standing person and only mentioned for completeness.
5. **Random walking motion:** the test was execute at the entrance of our University. Many people were entering and leaving the building. Even sometimes in small groups.
6. **Random running motion:** the persons were asked to run in a jogging manner through the FOV of the camera in various paths.

For each of the ten persons and the corresponding posture/motion 200 frames have been evaluated. To avoid manual annotation a simplified change detection was applied. Initially the point cloud size (after ROI building) of ten subsequent frame has been averaged and stored. In the evaluation phase the size of the recent acquired point cloud is compared to the stored size. If the difference is above certain threshold, the person has entered the cameras FOV. This simplified evaluation was applied for the test cases 2, 3 and 6. In case of test case 1 and 4, we waited until the person reached a new position and then evaluated each time five frames. For test case 5, each frame had to be manually annotated since the number of persons in the FOV was varying between one and five during the whole test.

Objective 3. How does the people detection system behave in a scenario-like setting.

So far the people detection system has been evaluated stand-alone. However, we are interested in how the system behaves when it is integrated on a real-world domestic service robot. We have integrated the system on our Care-O-bot 3 robot and performed a more scenario-like evaluation, where an autonomous mobile service robot tries to find a predefined number of persons in the environment. The scenario is basically derived from the "Who is who?" test in the RoboCup@Home competition where five people are spread around in the apartment. As an initial knowledge, the robot has a map of the environment and a set of room poses for each part of the apartment (e.g., living room or kitchen). In our test implementation a script first generates random positions in the map for five persons (also defining whether the person should sit or stand). In case the proposed position is blocked (e.g., a wall or table) the person will be assigned to stand/sit next to the generated pose. When all persons are placed at the generated positions, the robot generates a random path through all available room poses. The rest of the experiment consists of executing a drive & search behavior which we have implemented for the RoboCup@Home competition.

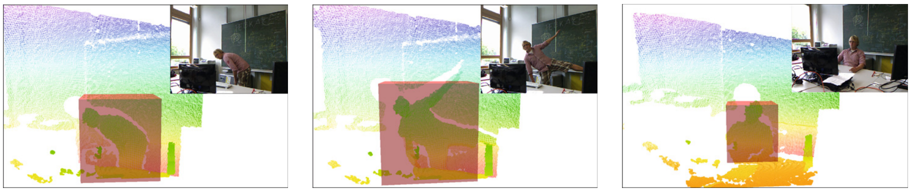
3.2 Experiment Results

Objective 1. Figure 5 depicts the cross-validation error with respect to the actual slice height. In case of no occlusion of the actual person the classification decreases with an increasing slice height. Above $50cm$ the error converges to an error rate of $\approx 15\%$. However, occlusion causes a major increase of the error rate when applied to an increased slice height. The reason is that the segmentation with high slice height creates clusters which might contain parts of the human and part of the object which causes the occlusion. We used the experiment to determine a good (minimized error rate) slice height. Thereby, we calculated the mean curvature for all three error curves and identified the global minima. A slice height of 25cm yielded in the minimum averaged error of 15.49%.

Objective 2. As shown in Table 1 our system shows a quite robust performance at least for standing person. In Figure 4 some detections for person poses are shown. The performance is independent from the actual distance to the person and is only limited by the predefined maximum distance of 5 meters. However, we observed a degrading detection rate when the person is sitting. The detection rate is significant lower, namely 74.94%. This is due to the fact that the training was only performed with standing persons. Therefore, only the head and the upper body can be detected. The horizontal leg parts can not be detected.

Table 1. Detection rates for different human poses and motions

Poses	Detection Rate	Motions	Detection Rate
standing	87.29%	not moving	87.29%
sitting	74.94%	rnd. walk	86.32%
part. occl.	82.35%	rnd. run	86.71%



(a) bending (b) random pose (c) partially occluded + sitting

Fig. 4. Detections for various person poses

In case the Random Forest would have trained also with sitting person, there would be clusters whose normal distribution would be similar to horizontal planes (because the upper leg is parallel aligned). Of course, this would cause a very high false positive rate. However, when a person is sitting, the upper body is still visible and sufficient for a quite robust detection with the model trained only with standing persons. Although, it is significant lower than detecting standing persons. Persons which were partially occluded, e.g. behind a table or a cupboard, can be detected similar robust to standing person, because only a minority of the lower body is occluded. For different motion speeds, only slightly different results could be observed. It does not matter in which speed the person is moving or even standing still, since the detection is done frame by frame. Only the pose configuration is different for the different motions. In general, the experiment showed that people are detected in various pose configurations and speeds with an average detection rate of 84.15%. A short video showing the people detection can be found on: `http://www.youtube.com/watch?v=d004nQE8Qko`.

Objective 3. In total ten runs of the described experiment were executed (see Table 2). In all cases the robot was able to find at least the two standing persons

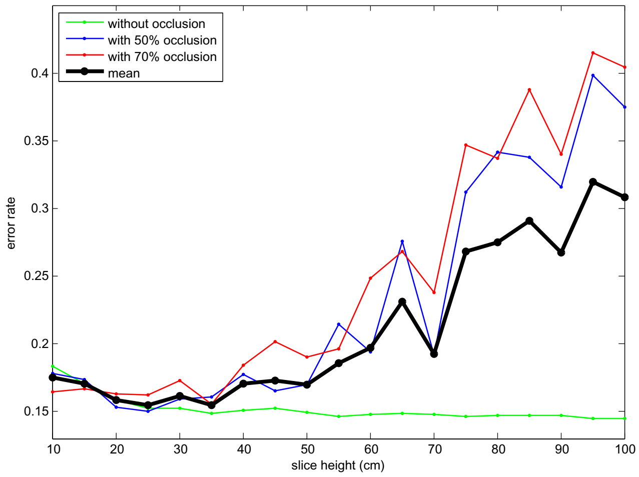


Fig. 5. Error rates of the segmentation in the presence of occlusion

Table 2. Result of 10 executed runs with auto-generated person positions (three standing and two sitting). TP = true positives, FN = false negatives, FP = false positives.

Run	TP standing	TP sitting	FN standing	FN sitting	FP
1	3	2	0	0	2
2	3	1	1	1	2
3	2	2	1	0	1
4	3	2	0	0	2
5	2	1	1	1	2
6	2	2	1	0	1
7	2	1	1	1	1
8	3	2	0	0	2
9	3	1	0	1	2
10	3	1	0	1	1

and always one sitting person. The missed detections where caused by a occlusion through another person or when the person was sitting in an arm chair and only a small part of the shoulder and head was visible. Beside the successful and missing detections, there were quite a lot false positive detections. In each run at least one false positive detection occurred. Due to the fact that a detected person (in this cases a false detection) is approached only once and then stored, the false detections do not effect the overall performance so much. Only the time for approaching the false detection for the first time is gone. However, in other scenarios this effect could result in a worst performance. Nevertheless, the integration of the people detection component into a higher level behavior was able to successfully detect the majority of people in the environment. Standing people could be detected with a rate of 86.67% and sitting person with 75.00% in this experiment. Astonishingly, the detection rates from this experiment almost reflect the results acquired in the experiment for the second objective.

4 Conclusion

We presented an approach to detect the 3*d* position of people in 3*d* point clouds using a feature vector which is composed of a histogram of local surface normals. The preliminary segmentation is based on a top-down/bottom-up technique which supports the detection of partially occluded persons, e.g. standing behind a desk or cupboard. The information gained from the local surface normals enables our system to detect a person in various poses and motions, e.g., sitting on other objects, bended to the front or side, walking fast/slow. With the presented approach we are able to detect even multiple people up to a distance of 5m with a detection rate of 84%. Future improvements will cover a reduction of false positive detections by extending the existing feature set with additional geometrical and statistical features. The proposed approach covered only the detection of people in 3*d*, a 3*d* tracking system would also enhance the overall system performance. We further aim an implementation on GPU, in order to improve the processing performance towards a real-time system. Another step would be the integration of color information into the detection process, which is provided simultaneously with the point cloud data by the Kinect sensor.

Acknowledgement. We gratefully acknowledge the continued support by the b-it Bonn-Aachen International Center for Information Technology.

References

1. Arras, K.O., Mozos, O.M., Burgard, W.: Using Boosted Features for the Detection of People in 2D Range Data. In: Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy, pp. 3402–3407 (2007)
2. Bajracharya, M., Moghaddam, B., Howard, A., Brennan, S., Matthies, L.H.: A Fast Stereo-based System for Detecting and Tracking Pedestrians from a Moving Vehicle. The International Journal of Robotics Research 28(11-12), 1466–1485 (2009)
3. Bohren, J., Rusu, R.B., Jones, E.G., Marder-Eppstein, E., Pantofaru, C., Wise, M., Mosenlechner, L., Meeussen, W., Holzer, S.: Towards autonomous robotic butlers: Lessons learned with the pr2. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2011)
4. Breiman, L.: Random Forests. Machine Learning 45, 5–32 (2001)
5. Graf, B., Reiser, U., Hagele, M., Mauz, K., Klein, P.: Robotic home assistant care-o-bot 3 - product vision and innovation platform. In: Proceedings of the IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO) (2009)

6. Holz, D., Holzer, S., Rusu, R.B., Behnke, S.: Real-time plane segmentation using RGB-D cameras. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 306–317. Springer, Heidelberg (2012)
7. Mozos, O.M., Kurazume, R., Hasegawa, T.: Multi-Layer People Detection using 2D Range Data. In: Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking, Kobe, Japan (2009)
8. Navarro-Serment, L.E., Mertz, C., Hebert, M.: Pedestrian Detection and Tracking Using Three-dimensional LADAR Data. The International Journal of Robotics Research 29(12), 1516–1528 (2010)
9. Satake, J., Miura, J.: Robust Stereo-Based Person Detection and Tracking for a Person Following Robot. In: Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking, Kobe, Japan (2009)
10. Spinello, L., Arras, K.O.: People Detection in RGB-D Data. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), San Francisco, USA (2011)
11. Spinello, L., Arras, K.O., Triebel, R., Siegwart, R., Luber, M., Tipaldi, G.D., Lau, B., Burgard, W., et al.: A Layered Approach to People Detection in 3D Range Data. In: IEEE International Conference on Robotics and Automation, Anchorage, Alaska, vol. 55, pp. 30–38 (2010)
12. Spinello, L., Siegwart, R.: Human Detection using Multimodal and Multidimensional Features. In: Proceedings of the International Conference in Robotics and Automation (ICRA), Pasadena, USA (2008)
13. Spinello, L., Siegwart, R., Triebel, R.: Multimodal People Detection and Tracking in Crowded Scenes. In: Proc. the AAAI Conf. on Artificial Intelligence: Physically Grounded AI Track, Chicago, USA, pp. 1409–1414 (2008)
14. Zivkovic, Z., Kroese, B.: Part based People Detection using 2D Range Data and Images. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, USA, pp. 214–219 (2007)

Simulation Competitions on Domestic Robots

Jianmin Ji, Zhiqiang Sui, Guoqiang Jin, Jiongkun Xie, and Xiaoping Chen*

Multi-Agent Systems Lab, School of Computer Science and Technology
University of Science and Technology of China, 230026, Hefei, China
{jizheng,zqsui,abxeeled,devilxjk}@mail.ustc.edu.cn, xpchen@ustc.edu.cn

Abstract. This paper reports a series of simulation competitions on domestic robots. All of these five competitions were based on a simulation platform focused on evaluating high-level functions of a domestic robot, including task planning and dialogue understanding. The object of holding these competitions is to promote research and development of service robots while avoiding limitations imposed by hardware of real robots. We also analyze the results and performances of participating teams since the competition was first held in 2009, showing that more and more terms are participating and they are performing better and better.

Keywords: RoboCup@Home, domestic robots, simulation platform, task planning, dialogue understanding.

1 Introduction

Researchers from Artificial Intelligence (AI), Robotics and related areas have shown increasing interest in developing intelligent service robots [1,3,8,12]. One of the most promising applications for a service robot is to provide services for untrained and non-technical users at home. Then, as a part of RoboCup, RoboCup@Home league [13] was held to develop service robots for future personal domestic applications and the RoboCup@Home competition is held each year since 2006. In the competition, a number of standard tests are used to evaluate robots' functions and performance in a realistic non-standardized home environment setting. These tests focus on functions which are essential for domestic applications including human-robot interaction, task planning, navigation, mapping, vision, object recognition, object manipulation, system integration and so on. However, due to the limitations of hardware and complexity of robotics techniques like vision, navigation, etc, it is not easy to test the different realizations of high-level cognitive functions of a real robot frequently or to develop a real robot to participate in competitions such as RoboCup@Home. In this paper, we report an effort against these limitations.

Five competitions have been held so far. All of them are based on the same simulation platform, though it has been upgraded several times. The platform

* Corresponding author.

is intended to evaluate the performance of a robot on task planning and dialogue understanding. A typical application scenario of a robot is to extract and understand users' requirements and information from dialogue through natural language interface, then resolve corresponding tasks and compute a plan of motions and sub-tasks to meet these requirements. Clearly, these two functions are indispensable for domestic applications, in addition to robots' hardware and underlying technologies in robotics. The platform simulates the low-level functions of an ordinary domestic robot and the features of common home environments related to the tests, by sending to each competing program a list of testing problems expressed in some verbal languages. The competing programs are required to try to solve all the testing problems, ie, to understand each problem and generate a plan for it within a given time limit. The competing programs are evaluated in terms of the performance of the plans they generate.

The first competition was held on December 2009 with 4 teams, while in 2011 two competitions were held with 12 teams and more challenging testing problems. In this paper, we analyze the results of all five competitions. It shows that more and more teams are participating and they are performing better and better. It also indicates that the platform can be used to compare different approaches for task planning and dialogue understanding of a domestic robot.

The rest of the paper is organized as follows. Section 2 presents the simulation platform. Section 3 and 4 report the results of competitions and compare the different approaches employed by the participating teams. Further discussions and conclusions are given in Section 6 and Section 7, respectively.

2 A Simulation Platform for Task Planning and Dialogue Understanding of Domestic Robots

Task planning and dialogue understanding play essential roles in the development of a domestic robot's high-level functions. In principle, these functions can be realized by various approaches. Then it is extremely interesting to compare these approaches in solving the same problems that involve these functions under the same conditions.

For this purpose, we developed a software platform for testing the relevant high-level functions of competing programs which may be developed by different approaches. The platform simulates the low-level functions of an ordinary domestic robot which could automatically move to a specific place, has an arm with a gripper to manipulate small objects and a plate to handle an object each time. The platform also simulates the features of common home environments related to the tests, including the location of an object, whether an object is portable, etc. Human-robot dialogue is simulated in a simplified way, by sending to each competing program a list of testing problems expressed in some verbal languages.

The competing programs are required to try to solve all the testing problems, ie, to understand each problem and generate a plan for it within a given time limit, 5 seconds. The competing programs are evaluated in terms of the performance of the plans they generate. Obviously, the simulated tests on the software platform are much simpler than what can be done with a real robot. But we get much more experimental data from the competitions on the platform, which in turn make the comparisons between different approaches possible.

Now we show some details.

The Primitive Actions of the Simulated Robot. A set of primitive actions are pre-defined in the platform. They keep fixed for all the testing problems. Following the AI convention, each primitive action is specified by its preconditions and effects. In the original version of the platform, there are five primitive actions, listed below.

– $move(X)$: The robot moves and arrives at location X.
– $pickup(A)$: The robot picks up object A.
– $putdown(A)$: The robot puts down object A.
– $toplate(A)$: The robot puts object A in its plate.
– $fromplate(A)$: The robot picks object A up from its plate.

Note that there is a plate on the robot, so that the robot can carry two objects, one in the plate and the other in its gripper. The definitions of these actions are also specified in PDDL statements[1].

The Testing Problems. Each testing problem is specified by a scenario description and a task description. The scenario description specifies the initial state of the home environment, which simulates the robot's perception since a real robot perceives its environment state through its sensors. The task description consists of one or more goals, constraints, and other additional information which the user tells the robot when he/she requests a specific service.

A scenario description provides the information of the types of the objects appeared in the scenario, their locations and other attributes. It also provides the current state of the robot. A scenario description is stored as a file in the following form. The objects and agents existing in the scenario are assigned a unique positive integer, denoted as *num*. In particular, number 1 represents the robot. For simplicity, number 0 is used to represent "nothing". Different locations are labeled as non-negative integers, denoted as *loc*. And *sort* denotes the type of the object. The properties (*prop*) of an object include the object's type, location, color and size:

$$\begin{aligned} color &:= \text{white} \mid \text{red} \mid \text{green} \mid \text{yellow} \mid \text{blue} \mid \text{black} \\ size &:= \text{big} \mid \text{small} \\ prop &:= sort(num). \mid color(num). \mid size(num). \mid \text{location}(num, loc). \end{aligned}$$

The robot's state includes its location, the state of the plate and the state of its gripper:

$$robot := \text{location}(1, loc). \mid \text{plate}(num). \mid \text{plate}(0). \mid \text{hold}(num). \mid \text{hold}(0).$$

[1] http://www.wrighteagle.org/homesimulation/en/competitions.php

A statement about a scenario description, denoted as *state*, is either a property of an object or a state of the robot: $state := prop \mid robot$.

We assume that human-robot dialogues are spoken in limited segments of natural languages (LSNLs) [5]. A task description specifies a user's task and may consist of three components: goal, constraint, and additional information. They are expressed in a simplified LSNL (actually, a command language). In fact, we set some sub-competitions for a real LSNL, however the results are similar for simplified LSNL. Therefore, we concentrate on the sub-competitions for the command language here.

Formally, a goal is defined as follows.

$$\begin{aligned} goal := {} & \text{give(human, } obj_1) \mid \text{puton}(obj_1,\ obj_2) \mid \text{goto}(obj_1) \mid \\ & \text{putdown}(obj_1) \mid \text{pickup}(obj_1), \end{aligned}$$

where

$$\begin{aligned} adj &:= \text{big} \mid \text{small} \mid \text{white} \mid \text{black} \mid \text{red} \mid \text{green} \mid \text{yellow} \mid \text{blue} \\ obj &:= sort \mid adj\ sort \end{aligned}$$

The additional information *info* is defined as follows, which specifies some supplementary information to the initial state of the problem:

$$info := \text{on}(obj_1,\ obj_2) \mid \text{near}(obj_1,\ obj_2) \mid \text{onplate}(obj_1)$$

A constraint *cons* is defined as:

$$cons := not\, goal \mid not\, info \mid not\, not\, info,$$

which specifies the conditions that must be satisfied during the process of task execution. *not goal* means the action specified in *goal* is forbidden, *not info* means the condition specified in *info* needs to be avoided, and *not not info* means the condition specified in *info* needs to be maintained.

Finally, a task description is defined as a set of *goal*, *cons* and *info*, and a statement $task := goal \mid info \mid cons$.

Scoring Criteria. A competition consists of two stages, each containing 40 testing problems. The competing programs are evaluated according to their total scores for all the testing problems in two stages. In the first stage, every task description only contains goals, while constraints and other additional information are used for the testing problems in the second stage.

The score of a competing program gets from a testing problem depends on the number of goals and constraints that the program accomplishes or maintains, as well as the number of primitive actions in the resulting plan generated by the program for the problem. The concrete criteria are as follows:

– Accomplishment of a goal: A goal is considered to be accomplished, if the final state after performing the plan generated by the competing program meets the goal specification.

– Maintaining a constraint: A constraint is considered to be maintained, if it has been satisfied from the initial state to the final one, in other words, every step of the plan's execution meets the requirement of the constraint.

The scoring system is defined as following:

– 10 marks for completing a goal.
– 5 marks for maintaining one constraint.
– −3 marks for each *move* action.
– −1 mark for each primitive action of the rest.

Therefore, the score for a testing problem is computed as: 10× the number of completed goals + 5× the number of maintained constraints − 3× the number of *move* actions − the number of the rest actions.

Like other RoboCup simulation leagues, we also developed a simulator logplayer to play back robot's actions in the visual simulation environment for a test problem, as shown in Figure 1.

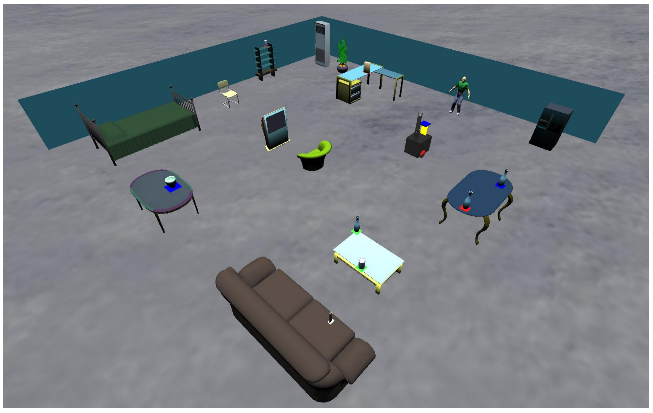


Fig. 1. The Simulator Logplayer for the Simulation Platform

3 Early Competitions

Three competitions based on the testing platform were held in 2009 and 2010[1], respectively. These competitions have similar testing problems. As time goes by, more teams participated and generally they performed better.

5 teams in total participated in the first two competitions on December 2009 and May 2010. All the 80 testing problems are the same in the two competitions. These problems were released after the first competition. All the participants to the second competition knew all the problems from beginning. They also debugged their programs with the problems. In this section, we report the results of the second competition and make comparisons based on the results.

We take three representative competing programs for comparison. They are representatives of the three approaches employed in the competitions and each of them got the highest score in its class. The first one is ours; the competing program is just the high-level part of KeJia robot [4,5], which is implicated via a nonmonotonic logic programming language called Answer Set Programming (ASP) [2]. This represents the nonmonotonic approach (denoted as NM). The second one is realized with search technology (Search). The third one is based on naive problem-solving approach (PS).

NM Approach. As presented in [4,5], the task planning problem in the competition is converted into that of finding an answer set of an ASP program, where the actions of the robot, the scenario descriptions and the task descriptions are represented as ASP rules. Due to the progress on ASP and ASP solvers, as well as the framework problem [11], causality [10], etc, this approach shows many advantages as expected. In particular, there is no difference in handling goals and constraints in this approach, while all the participants adopting other approaches complained about the difficulty of handling constraints. However, efficiency is still the major bottleneck for this approach. In KeJia system, we use iclingo [9] (an incremental ASP solver) to compute answer sets. For a testing problem with 20 portable objects and 14 locations, the system can compute an optimal plan with 12 actions in 5 seconds. More complicated problems may not be solved within the time limit.

Search Approach. The search approach treats a testing problem as a search problem. The competing program is based on a depth-first search algorithm with some pruning strategies. Firstly, the initial state is acquired from the scenario description and the additional information in the task description. Based on the initial state, the algorithm chooses an primitive action to expand, if the succeeding state meets the requirement of the task, then a plan is computed and it would be stored if it is better than the current best partial plan. If a plan is found, there are not any proper actions to expand, or the search steps are longer than the current best plan, then the algorithm will backtrack to the precious state. Due to the very large state space, strong pruning strategies are needed to ensure that the algorithm terminates in a finite time. But these strategies may exclude the optimal plan—this is the price for the efficiency of the program.

PS Approach. This approach requires the programmer to predict the detailed solutions for the possible cases of testing problems. A simple strategy is to code a solution for each goal in the task description. The generated plan can be improved by adjusting the order of goals and choosing proper objects. For example, if there are two goals "give" and "puton", the program can choose the objects which are initially at the same location, then an optimal plan can be achieved by holding these two objects at the same time (pick up one and put another on the plate). It is not easy for this approach to handle constraints in the task description. Instead, the constraints were only employed to rule out some forbidden actions. This approach is efficient, but not flexible or reliable. It cannot guarantee to compute the optimal plan for every problem. Re-programming is needed when the domain of the problem changes.

Results. There are 14 locations, 8 to 21 different portable objects in the testing scenarios. Initially the robot may have some portable object on its plate or in its gripper. We have run the platform for the second competition on a computer with an AMD Athlon(tm) II X4 620 CPU and a 2GB RAM, the logs and the competing programs can be downloaded from the web site[1].

For a single problem in stage 1, there are 2 to 4 different goals in the task description and optimally it will take 5 to 15 actions to accomplish a task. Note that the difficulty of a testing problem depends on whether or not it contains "related goals". Two goals in a problem are called *related*, if interleaving the execution of actions for them can reduce the total cost (an example is given in Section 5). If goals in a problem are not related, then they can be accomplished separately without loss of efficiency. Based on this observation, we list the results with and without related goals, respectively. The results of stage 1 are shown in Table 1.

Table 1. Results of stage 1 for the 2nd competition

competing program	score for problems without related goals (14)	score for problems with related goals (26)	total score (40 problems)
NM approach	261	460	721
search approach	242	343	585
PS approach	274	410	684

The competing program based on the NM approach returns the optimal plans for 38 problems in stage 1, while the competing programs based on the search and PS approaches find out plans, which may not be optimal, for all problems. For problems without related goals, the NM approach program and the PS approach program compute the same results, except for one problem for which NM runs out of time. For problems with related goals, the NM program can always compute the optimal plans if it completes the task within the time limit, while the PS program returns the plans which are closed to the optimal plans.

For a single problem in stage 2, there are 2 to 4 different goals, at most 5 constraints and 3 pieces of additional information. Optimally, a program will take 5 to 13 actions to accomplish a task. If a problem contains constraints, it requires further reasoning. For example, "pickup(red bottle)" is a goal and "not not on(bottle,table)" is a constraint, which means that "there must be a bottle on the table". Suppose that initially the 'red bottle' is the only bottle on the table. Then the robot should first put another bottle on the table to accomplish the task. Therefore, constraints add another kind of difficulty. The results of stage 2 are shown in Table 2.

The NM approach returns the optimal plans for 39 problems in stage 2, while the search and PS approach find out plans for all problems. The results show that the NM approach works better for problems with constraints if it can complete the planning in time (it runs out of time for one problem with constraints). It is

Table 2. Results of stage 2 for the 2nd competition

competing program	score for problems without constraints (9)	score for problems with constraints (31)	total score (40 problems)
NM approach	133	700	833
search approach	112	552	664
PS approach	120	625	745

Table 3. Results of stage 1 for the 3rd competition

competing program	score for problems without related goals (8)	score for problems with related goals (32)	total score (40 problems)
Team A (NM)	156	705	861
Team B (NM)	149	697	846
Team C (PS)	132	654	786
Team D (search)	117	597	714
Team E (PS)	108	542	650
Team F (PS)	131	515	646
Team G (PS)	124	508	632
Team H (PS)	118	464	582
Team I (PS)	20	-77	-57
Team J (PS)	-20	-98	-118
Team K (PS)	-36	114	-150

Table 4. Results of stage 2 for the 3rd competition

competing program	score for problems without constraints (5)	score for problems with constraints (35)	total score (40 problems)
Team A (NM)	87	948	1035
Team B (NM)	69	854	923
Team C (PS)	84	815	899
Team D (search)	74	826	900
Team E (PS)	58	798	856
Team F (PS)	72	739	811
Team G (PS)	76	726	802
Team H (PS)	87	653	740
Team I (PS)	17	266	283

also shown that the competing program by search approach encountered more difficulty in handling constraints.

For most testing problems in both stages, the NM approach program can find out the optimal plans in 5 seconds, but fails for some complicated problems (need more than 12 actions to accomplish). This indicates that the NM program is "religious" and "cautious". The search approach program returns plans for all problems in both stages, but the pruning strategies rule out the optimal plans for most problems. The PS approach program returns plans for all problems.

Although for most problems in stage 1 the results are closed to optimal plans, the gap grows for complicated problems, especially when there are constrains.

Another competition was held on July 2010[1]. The competition uses 80 new testing problems with the similar size of previous problems. There are 11 different teams in the competition. Their results and corresponding approaches are listed in Table 3 and 4. Note that, the results still meet the previous observation.

4 The 4th and 5th Competition

In 2011, two competitions based on the simulation platform were held on May[1] and August[2] respectively with more challenging testing problems. Each of the competitions has 12 teams. Different from previous ones, more primitive actions are allotted to the virtual robot and more variables are considered in these competitions. In particular, a new type of objects named "container" is introduced and four new primitive actions become available to the virtual robot.

- *putin*(A, C): The robot puts object A into container C.
- *takeout*(A, C): The robot takes out object A from container C.
- *open*(C): The robot opens container C.
- *close*(C): The robot closes container C.

Generally, each testing problem involves 30 portable objects and 17 locations, which requires 12 to 23 actions to accomplish the test.

Clearly, testing problems became more challenging. However, most teams still performed well. Despite approaches reported in Section 3, some new approaches are employed in the competitions.

Improved NM Approach. On top of the NM approach, "macro actions" are introduced as consecutive executions of some original actions. When a plan contains a macro action, it means the robot should execute a sequence of actions at the step. Clearly, using macro actions can reduce the number of actions in a plan, thus improve the efficiency of the original approach. However, the plan contained with macro actions may not be an optimal solution. We can remedy the problem through careful definitions of macro actions.

IDA* Search Approach. The approach is based on the Iterative Deepening A* (IDA*) search algorithm, which is a variant of the A* search algorithm using iterative deepening to keep the memory usage lower than in A*. The heuristic is essential for the performance of the approach and some pruning techniques are still required for certain cases.

NM Plus PS Approach. The NM approach can compute an optimal plan taking a long time, while the PS approach can compute a plan in shorter time that is not necessarily optimal. This approach combines the benefits of both approaches. It first provides a skeleton of the solution by the PS approach, then fulfills details by the NM approach. However, the solution may not be optimal.

Improved PS Approach. The approach improves the original PS approach through a much deeper analysis of structures of corresponding testing problems.

[2] `http://www.wrighteagle.org/rco/athome/2011/results.php`

For each testing problem, the approach creates a directed graph based on the initial and goal locations of related objects. Then, a strategy of solving the problem is chosen based on the structure of the graph.

The results of the 5th competition (similar to the results of the 4th competition) are listed in Table 5. It shows that most teams perform well and the Improved NM approach and the IDA* approach perform better than others.

Table 5. Results of the 5th competition

Team Name	score for stage 1 (40 problems)	score for stage 2 (40 problems)
Team A(improved NM)	1060	1880
Team B(IDA*)	1061	1850
Team C(NM+PS)	912	1735
Team D(NM+PS)	1003	1691
Team E(improved PS)	954	1672
Team F(improved PS)	844	1671
Team G(PS)	787	1486
Team H(PS)	816	1448
Team I(PS)	588	-
Team J(PS)	435	-
Team K(PS)	357	-
Team L(PS)	0	-

5 Discussion

Since 2010, the RoboCup@Home competition has added a new suit of tests, named General Purpose Service Robot (GPSR) [6,7]. Different from other tests, GPSR is not incorporated into a story and there is not a predefined set of actions. In the test, the domestic robot is asked to serve user's needs which are specified in English. Note that, the requirements for high-level functions in GPSR are similar to the requirements in simulation competitions reported in this paper.

We believe that, besides underlying robotics techniques, high-level functions are also crucial for future domestic applications of a service robot. In the simulation competitions, the following three issues related to high-level functions are mainly considered.

(1) **Planning for Complicated Tasks** Figure 2 shows an example of a complicated task. Suppose you and your friend are setting in the living room and you ask your robot to fetch two cans of beer from the dining room. This request is a complex task consisting of two related goals, move the first can from the dining room to the living room and move the second from the dining room to the living room. If the robot cannot understand or make use of the relatedness of the goals, it will fetch the cans one by one separately, as shown in Figure 2a. Obviously, it is not necessarily optimal and is typically inefficient. An optimal plan is shown in Figure 2b. This is the way an intelligent robot is expected to do it. In the simulation platform, the optimal plan would get the highest score. Different testing problems in competitions correspond to various complicated tasks in domestic applications.

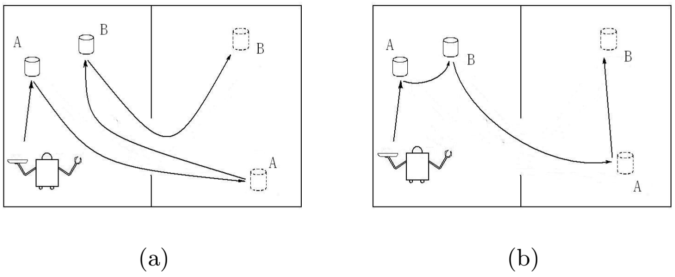


Fig. 2. Related goals

(2) **From Dialogue Understanding to Planning** An important requirement for a intelligent service robot is to extract knowledge and information from human-robot dialogue, and translate them to task planners, with which the task planners can make use of the knowledge and information to solve new problems and the robots can accumulate knowledge and improve performance. In competitions, we use tasks specified in LSNLs or a simplified LSNL to simulate sentences in the human-robot dialogue.

(3) **Efficiency Issues** Robots are required to quickly respond to users' utterances. Then efficiency issues become more acute, dialogue understanding and task planning should be terminated in a short time. In competitions, each program needs to return a result in 5 seconds, which is taken as the length of time that users can tolerate.

From the results of the series of competitions, we can see that most teams perform better and better, especially these teams using the Improved NM approach or the IDA* approach. With the accumulation of the five competitions, we can see that the testing problems become more and more challenging. In the 1st competition, a testing problem may contain 14 different locations and 8 to 21 portable objects, and the problem can be solved less than 15 steps. While in the 4th and 5th competitions, a testing problem involves 17 locations and 30 portable objects, and the problem requires 12 to 23 actions to be solved. On the other hand, the performance of participating teams also become better and better. In the first two competitions, only a few teams performed well. While in the last two competitions, most teams can solve almost all testing problems and the differences of their performances are lessening.

6 Conclusion

In this paper, we report five simulation competitions based on a platform for evaluating high-level function of a domestic robot. These competitions focus on the performance of a robot on task planning and dialogue understanding while avoiding the consideration of robots' hardware. From the results of the series of competitions, we can see that more and better approaches have been developed through the competitions, indicating that the competitions are welcome by researchers and students (graduates and undergraduates) and also helpful for promoting research and education on high-level functions of service robots. In addition, we hope this competition will help draw more and more teams to participate in real robot competitions as real robots become available to more and more people.

In the future, we will extend the simulation competition to consider other high-level functions of domestic robots, including coping with dynamic environments, failure recovery, uncertain information processing, human-robot dialogue during the execution of a current plan, multi-robot scenarios and so on.

Acknowledgments. This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the National Natural Science Foundation of China under grants 60745002 and 61175057. We thank Daniele Nardi, Wei Liu and Fangkai Yang for their help to this work. We are also grateful to the anonymous reviewers for their constructive comments and suggestions.

References

1. Asoh, H., Vlassis, N., Motomura, Y., Asano, F., Hara, I., Hayamizu, S., Ito, K., Kurita, T., Matsui, T., Bunschoten, R., Kröse, B.: Jijo-2: An office robot that communicates and learns. IEEE Intelligent Systems 16(5), 46–55 (2001)
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, New York (2003)
3. Burgard, W., Cremers, A., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: Experiences with an interactive museum tour-guide robot. Artificial Intelligence 114(1-2), 3–55 (1999)
4. Chen, X., Jiang, J., Ji, J., Jin, G., Wang, F.: Integrating nlp with reasoning about actions for autonomous agents communicating with humans. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (IAT 2009), pp. 137–140 (2009)
5. Chen, X., Ji, J., Jiang, J., Jin, G., Wang, F., Xie, J.: Developing high-level cognitive functions for service robots. In: Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), pp. 989–996 (2010)
6. Committee, R.H.T., et al.: Robocup@home: Rules and regulation (2010)
7. Committee, R.H.T., et al.: Robocup@home: Rules and regulation (2011)
8. Ferrein, A., Lakemeyer, G.: Logic-based robot control in highly dynamic domains. Robotics and Autonomous Systems 56(11), 980–991 (2008)
9. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental asp solver. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 190–205. Springer, Heidelberg (2008)
10. Lin, F.: Embracing causality in specifying the indeterminate effects of actions. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI 1996), pp. 670–677 (1996)
11. Reiter, R.: The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In: Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, pp. 359–380 (1991)
12. Tenorth, M., Beetz, M.: KnowRob — knowledge processing for autonomous personal robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), pp. 4261–4266 (2009)
13. Wisspeintner, T., van der Zant, T., Iocchi, L., Schiffer, S.: Robocup@home: Scientific competition and benchmarking for domestic service robots. Interaction Studies 10(3), 392–426 (2009)

Throwing Skill Optimization through Synchronization and Desynchronization of Degree of Freedom

Yuji Kawai[1], Jihoon Park[1], Takato Horii[1], Yuji Oshima[1], Kazuaki Tanaka[1,2], Hiroki Mori[1], Yukie Nagai[1], Takashi Takuma[3], and Minoru Asada[1]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering, Osaka University, Osaka, Japan
[2] CREST, Japan Science and Technology Agency
[3] Dept. of Electrical and Electronic Systems Engineering, Osaka Institute of Technology, Osaka, Japan
robocup@er.ams.eng.osaka-u.ac.jp

Abstract. Humanoid robots have a large number of degrees of freedom (DoFs), therefore motor learning by such robots which explore the optimal parameters of behaviors is one of the most serious issues in humanoid robotics. In contrast, it has been suggested that humans can solve such a problem by synchronizing many body parts in the early stage of learning, and then desynchronizing their movements to optimize a behavior for a task. This is called as "Freeze and Release." We hypothesize that heuristic exploration through synchronization and desynchronization of DoFs accelerates motor learning of humanoid robots. In this paper, we applied this heuristic to a throwing skill learning in soccer. First, all motors related to the skill are actuated in a synchronized manner, thus the robot explores optimal timing of releasing a ball in one-dimensional search space. The DoFs are released gradually, which allows to search for the best timing to actuate the motors of all joints. The real robot experiments showed that the exploration method was fast and practical because the solution in low-dimensional subspace was approximately optimum.

1 Introduction

Skilled behaviors of a humanoid robot are important in the robot soccer domain. Soccer skills such as throwing, kicking, and biped locomotion require coordination of the whole body movements with a large number of degrees of freedom (DoFs). Designing a skilled behavior of a humanoid robot with high DoFs is one of the most serious issues.

There exist many studies on the heuristic exploration approach to solve such problems. Among them, evolutionary computation (e.g., [1,2]) and particle swarm optimization (e.g., [3,4]) enabled the robot to acquire faster gait. Main optimization parameters in these studies have been trajectories of limbs or parameters of Central Pattern Generators. However, the number of iterations including evaluation of performance was very large because of a vast. Generally, real robots are prone to be easily broken, therefore optimization methods with much less trials are desired.

Peter and his colleagues [5–7] have demonstrated that Hill Climbing and Policy Gradient algorithms successfully optimized the parameters for quadruped locomotion and

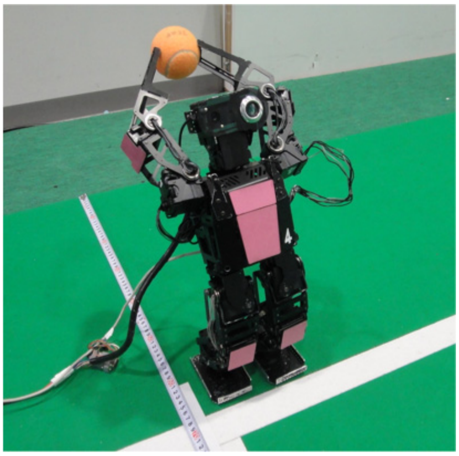


Fig. 1. Throwing for exploration of optimal parameters

kicking a ball. These algorithms converge to solution more rapidly than evolutionary computation and particle swarm optimization. However, the complexity of a robot's body still intrinsically causes a large number of iterations.

We take an idea from the progression of skills in humans who have their high-dimensional motor space. Bernstein [8] (see also [9, 10]) suggested freezing and releasing of DoFs in skill acquisition. In the early stage of learning of a motor skill, some DoFs are reduced (frozen). The DoFs are then released (freed) gradually as the skill progresses. These stages of motor learning enable to reduce the search space dimensionality. Yamamoto and Fujinami [11] also found a common organization of acquisition of a periodic skill: synchronization and desynchronization. They compared clay kneading movements for pottery of experienced subjects with those of the experts. While the experienced subjects tend to synchronize their body parts, slight phase differences between body parts are observed in the experts' movements. Their group [12] found the similar process for proficiency of samba dance. A possible interpretation of the synchronization of movements in the early learning for the skill is that smaller number of parameters of the movements simplify the optimization by reducing the dimensionality.

We introduce this idea of synchronization and desynchronization to optimization methods, and then apply the method to progress of soccer throwing skill. Of importance in the acquisition of skilled throwing is the timing when to release the ball. A robot searches for the best timing to actuate each joint based on timing of releasing a ball through practice as shown in Fig. 1. All joints related to the throwing skill are synchronized initially. That is, the robot optimizes roughly the timing of releasing a ball in one-dimensional space. The joints are then gradually released, which allows the robot to search more optimal parameters. As a result, the robot acquires skilled throwing even with a small number of trials.

This paper is organized as follows: In section 2, we explain heuristic exploration using synchronization and desynchronization of DoFs. Throwing parametrization of a

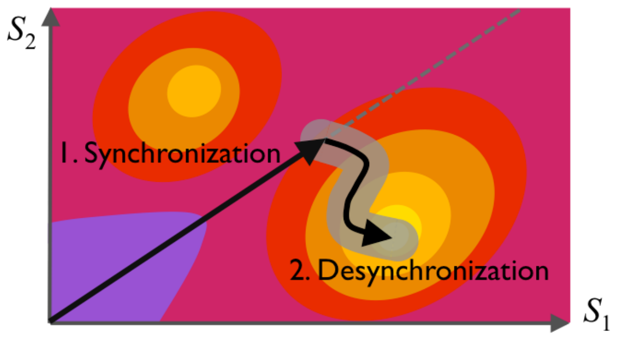


Fig. 2. An example of exploration by Hill Climbing algorithm through synchronization and desynchronization. In case of a two-dimensional objective function, the two parameters, S_1 and S_2, are synchronized at first. These parameters are desynchronized after optimization in one-dimensional space. The glay area means the neighborhood of a local solution in two-dimensional space. The constraint on the search space enables faster exploration.

humanoid robot and the experimental setting are described in sections 3 and 4, respectively. Section 5 then demonstrates that the proposed optimization method results in quicker exploration of optimal parameters. In section 6, the results are given and future issues are discussed, and in section 7, we conclude our research.

2 Heuristic Exploration through Synchronization and Desynchronization

2.1 Synchronization and Desynchronization

Fig. 2 illustrates an example of exploration through synchronization and desynchronization of parameters. Here, we assume a two-dimensional search space, that is, the only two parameters are S_1 and S_2. There are two stages of optimization: synchronization and desynchronization.

Synchronization. The search space is restricted to synchronization of all parameters, namely, $S_1 = S_2$ $(= S_3 = \cdots = S_n)$. An initial value is selected in one-dimensional search space (on the dashed line in Fig. 2). An optimal parameter is then explored on this line.

Desynchronization. The restriction is gradually lifted after finishing the optimization in the previous search space. The search space is hence extended to one of other dimensions. Initial values are the best one in the previous stage. A solution of this algorithm is an optimal set of parameters when all parameters are explored.

The following is a procedure:

1. Synchronization process (above): one-dimensional search by freezing.
2. $i = 1$, and repeat the following until all dimensions are explored.

2-1 Release one dimension ($i = i + 1$) and apply an optimization method starting from the optimal solution in the previous stage (before releasing) as the initial value in i-dimensional search space.
2-2 Find the optimal one in this space. If $i = n$ (the full dimension), then the optimal one is globally optimal. Else, go to 2-1

Therefore, the dimension of the search space is gradually increasing while the search area is expected to decreasing owing to starting the optimization from reasonable initial value.

2.2 Optimization Method

Every time one dimension is released, an optimization method is applied in the search space. We use a Hill Climbing algorithm and a modified particle swarm optimization (PSO). These algorithms are widely applied to parameter optimization problems (see [3–5, 7]). In the both algorithms, the initial value in next search space is solution in previous one.

Hill Climbing. A Hill Climbing algorithm is one of the simplest optimization methods. It is well-known that this algorithm explores a solution quickly. An initial value is selected and evaluated in the search space. All neighbors of the initial one are evaluated, and the highest-scoring parameter among the neighbors is selected. The selected value is the next center, and then repeat the evaluation and the selection until no higher scores can be found.

Modified Particle Swarm Optimization (PSO). PSO [13] is a probabilistic optimization method just as genetic algorithms. Initially, a swarm of N particles is generated in the D-dimensional search space. Here, we introduce an initial value to this algorithm so that the optimization can inherit the best parameters in the previous search space. Although the existing PSO gives randomly the positions of initial particles, we give the initial positions according to normal distribution, where its mean and variance are the initial value and v, respectively. These particles are assigned a position $\mathbf{x_i}$ and a velocity $\mathbf{v_i}$ ($1 \leq i \leq N$), which are both D-dimensional vectors. Each particle is evaluated by the performance. At each iteration, the velocity of each particle is updated depending on two values: the personal best position $\mathbf{p}best_i$ ($1 \leq i \leq N$) and the global best position $\mathbf{g}best$. $\mathbf{p}best_i$ is the best position that each particle has ever evaluated. $\mathbf{g}best$ is the best position that all particles have evaluated. Each velocity $\mathbf{v}_i^t$ at iteration t is updated by:

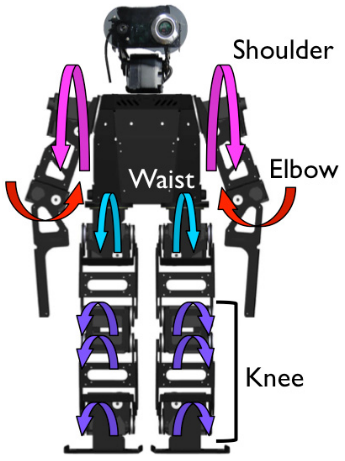
$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_p r_p^t \times (\mathbf{p}best_i^t - \mathbf{x}_i^t) + c_g r_g^t \times (\mathbf{g}best^t - \mathbf{x}_i^t), \tag{1}$$

where, w, c_p and c_g are weights. r_p and r_g are normal random numbers between 0 and 1. We restrict the range of velocity between $-\mathbf{v}^{max}$ and $\mathbf{v}^{max}$, which is determined by:

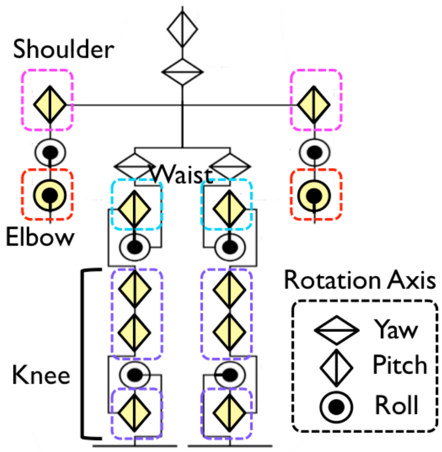
$$\mathbf{v}^{max} = k \times \mathbf{x}^{max}, \tag{2}$$

where, $\mathbf{x}^{max}$ is the range of exploration in each dimension, and $0.1 \leq k \leq 1$. The next positions of particles $\mathbf{x}_i^{t+1}$ are calculated by:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}. \tag{3}$$



(a) Front view of VisiON 4G and its essential DoFs for throwing.



(b) DoF configuration of VisiON 4G. The yellow joints are used in this experiment.

Fig. 3. The number of the substantial DoFs used in the current experiment is 4: the pitch shoulder, the roll elbow, the pitch waist and the pitch knee. we assume symmetry of the motors. The DoF of the knee consists of 6 motors. The elbow affects the holding and releasing the ball.

We judge the end of the exploration when **g***best* does not change during *n* iterations in the current experiment.

3 Throwing Parametrization

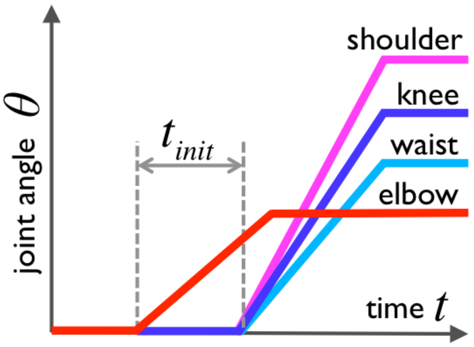
A robot searches the optimized combination of the start timing of each joint to throw the ball as far as possible. The VisiON 4G robot was used for this experiment (see Fig. 1), a commercial humanoid robot manufactured by Vstone Co.,Ltd. Fig. 3 depicts the robot's DoF configuration. The robot has 22 DoFs and each joint is actuated by a VS-SV410 servomotor.

We, however, selected essential 4 DoFs for throwing:

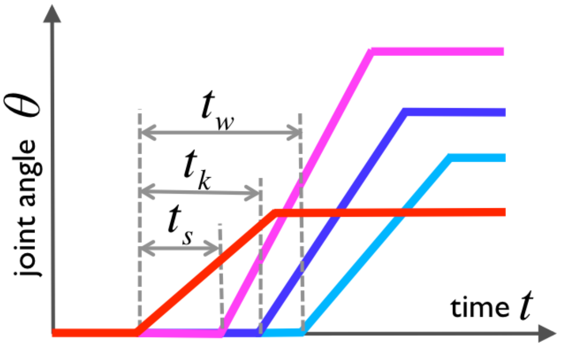
- Pitch shoulder: throwing the ball overhead.
- Roll elbow: holding and releasing the ball.
- Pitch waist: achieving more force by the reaction.
- Knee: stretching the both knees, which consisting of 6 motors.

Unfortunately, VisiON 4G does not have the DoF of pitch elbow, which is required for human throwing.

Fig. 4 shows the definition of the parameters. We did not use velocities or positions of individual DoFs but the timing of movements of three DoFs (shoulder, knee, and waist) as the parameters. Here, DoF of the elbow is a base of the timing because it is important for a skilled throwing to optimize the timing of releasing the ball. We defined the timing of the start of movements of the shoulder, knee and waist based on the



(a) Synchronization of DoFs in the first stage



(b) Desynchronization of DoFs in the latter stage

Fig. 4. The number of parameters increases gradually in the learning. In the early stage of learning (a), the DoFs of the shoulder, the knee and the waist are synchronized. The robot explores the optimal t_{init}, namely timing of releasing of the ball in the one-dimensional space. In the last stage of learning (b), the timing of the start of the each DoF, t_s, t_w and t_k, is optimized.

elbow's timing as t_s, t_w and t_k, respectively. The robot learns the optimal $\mathbf{t} = (t_s, t_w, t_k)$ through practice. Initially, the 3 DoFs are synchronized, i.e., $t_s = t_w = t_k = t_{init}$, and then the robot optimizes t_{init} (see Fig. 4(a)). Secondly, one DoF, the shoulder, the waist or the knee, is differentiated from other DoFs. If the DoF of the shoulder is selected here, the parameters are t_s and $t_w = t_k$. In the last stage of learning, all of DoFs are desynchronized. Thus the robot searches optimal $\mathbf{t}$ in the three-dimensional space (see Fig. 4(b)).

4 Experimental Setting

In order to validate the proposed optimization method, we conduct experiments using a real robot. The robot explores optimal combinations of t_s, t_w, and t_k.

The performance is evaluated by the distance between a robot's toe and a ball fall point. The throwing distance is measured by a visual inspection through video

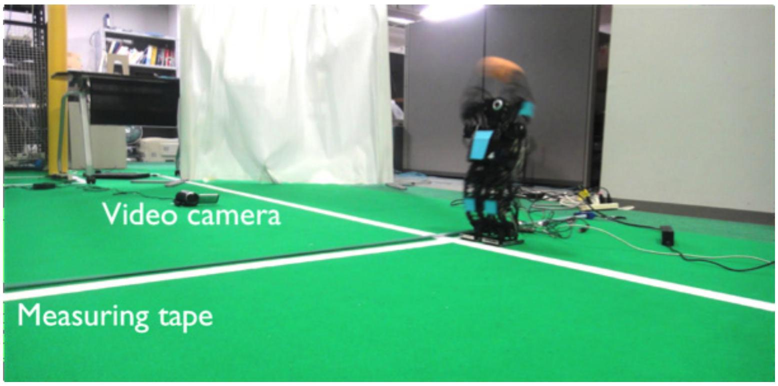


Fig. 5. The experimental environment to optimize the parameters for throwing. We recorded the distance between a robot's toe and a ball fall point.

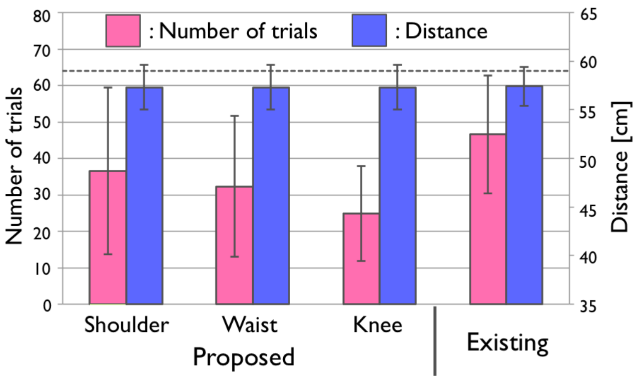
recording with a measuring tape as shown in Fig. 5. We evaluate the distance of throwing regardless of the posture after throwing (keep standing or not).

The robot starts its motion from the same initial pose as shown in Fig. 1 in each trial. We give the ball to the robot so that the robot can hold the ball with both hands. It takes 10 steps to execute throwing motion, where 1 step is 1/30 sec. The range of exploration is set to [-5, +2] based on start timing of the elbow. The robot rests for 5 minutes every time after 10 trials to prevent overheat of the motors.

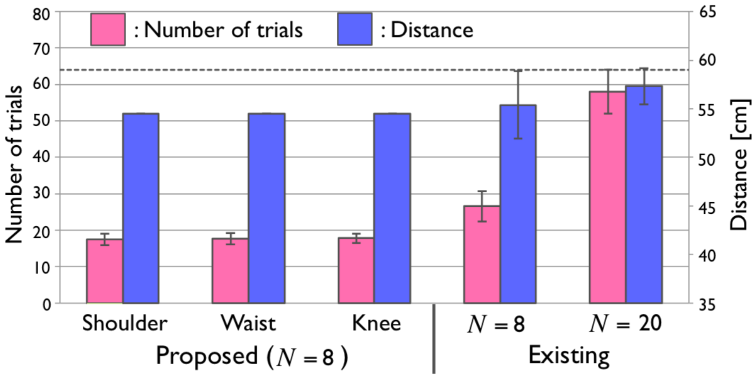
Optimization experiments are conducted off-line. We evaluate optimization methods using dataset obtained by exhaustive search in advance. Two trials of the experiment are performed, each of which consists of 512 different timings. The objective function is given the mean of two trials. We tested four optimization algorithms: Hill Climbing and PSO through synchronization and desynchronization, and existing Hill Climbing and PSO. We then compare the number of the evaluations and achieved optimal performance.

In the Hill Climbing algorithm, one iteration needs 26 evaluations in three-dimensional search space. We, however, does not count the evaluations of the parameters where the robot once searched. The variables in the PSO are empirically determined: 8 particles are initially positioned according to a normal distribution, whose variance is set to 3. We set $w = c_p = c_g = 0.5$ in Eq. (1) and $k = 0.25$ in Eq. (2). The optimization is finished when the $\mathbf{g}_{best}$ does not change for 3 iterations.

An initial parameter is given as an integer between -5 and 2 (i.e., 8 patterns). Each optimization method is conducted 8 times for all initial parameters. The proposed PSO is ran 10 times with each initial parameter setting because PSO includes randomness. In the existing PSO, randomly-selected initial parameters are given, and then we test it 80 times.



(a) Hill Climbing algorithm



(b) PSO

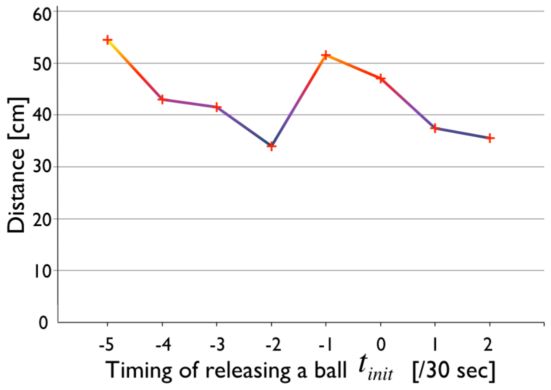
Fig. 6. The results of (a) Hill Climbing algorithm and (b) PSO. The blue and the pink bars indicate the number of trials and throwing performance, respectively. The proposed methods desynchronized DoF of shoulder (left), waist (middle), or knee (right) from other DoFs in the second stage. *N* is the number of particles. The dashed line denotes the global optimum (59cm). Each errer bar indicates the standard deviation.

5 Result

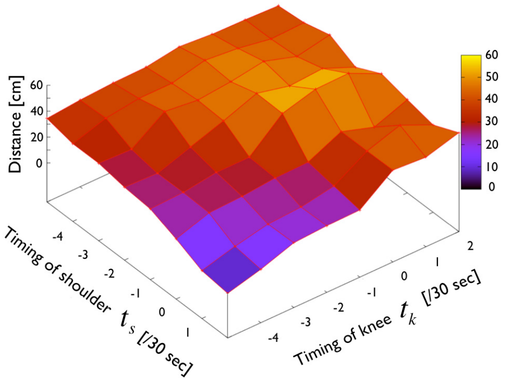
5.1 Number of Trials and Throwing Performance

Fig. 6 shows the results of optimization methods: (a) Hill Climbing algorithm and (b) PSO. The pink bars denote average of the number of trials in each optimization method. Less trials mean faster exploration, which relieves the robot of load. The blue bars denote average of flying distance of a ball, i.e., throwing performance. There are three results in the proposed optimization through synchronization and desynchronization: shoulder (left), waist (middle) or knee (right) was differentiated from other DoFs in the second stage.

It is noted that the results of both proposed methods show less trials than the existing methods. We can find that all results of Hill Climbing show high throwing



(a) One-dimensional search space



(b) Two-dimensional search space

Fig. 7. Objective function. Proposed algorithm optimizes t_{init} in the one-dimensional space (a) in the first stage of exploration. In two-dimensional search space with $t_w = -1$ (b), the global optimal parameter $\mathbf{t}^{opt}$ is (-1, -1, 1) which results in a distance travelled of 59cm.

performance (see Fig.6(a)), which are equivalent to global optimum: the dashed lines (59cm) in Fig. 6. The results of proposed PSO through synchronization are less variance and nearly the same performance as existing PSO with $N = 8$. Therefore, the proposed method can reduce the number of trials while maintaining the high performance.

The number of trials in PSO, compared to the results of the proposed Hill Climbing, is much less. However, the throwing performance of PSO with $N = 8$ is worse than global optimum. More particles (e.g., $N = 20$) are required for the same level of throwing performance as Hill Clibming. There esists a tradeoff between performance and number of particles in PSO.

5.2 Objective Function

In Fig. 7, we show the objective function obtained by exhaustive search to discuss above result. Fig. 7(a) illustrates one-dimensional objective function, where the optimal t_{init} is explored in the first stage. There are two local maxima: $t_{init} = -1$ and -5. The global optimum $\mathbf{t}^{opt}$ is (t_s, t_w, t_k) = (-1, -1, 1) as shown in Fig. 7(b). Thus, the result of optimization in one-dimensional space should be -1 so that the robot can find the $\mathbf{t}^{opt}$ finally. In Hill Climbing, the optimal t_{init} is -1 if initial value is more than -3. On the other hand, a particle swarm found -5 as global best and then all particles move toward -5 in PSO. This is why the optimization by PSO with synchronization of DoFs was worse than by Hill Climbing (see Fig. 6).

The synchronization of the DoFs of the shoulder and the waist simplifies to reach $\mathbf{t}^{opt}$ because the $\mathbf{t}^{opt}$ is $t_s = t_w = -1$. Thus, the Hill Climbing through synchronizing the DoFs of t_s and t_w in the second stage results in the least number of trials as shown the knee's pink bar in Fig. 6(a). The adequate order of releasing the DoFs may be task-dependent.

6 Discussion

6.1 Necessity of Optimization in Real World

It is hard for a robot to acquire a skilled throwing. There exists a gap between the real and the virtual world even if we use a realistic simulator or make a dynamical mathematical model of a robot. One of the differences originates from the environmental complexity. The robot's body, for example, interacts with the ball during throwing. The ball deforms slightly and the robot undergoes reaction forces. This interaction seems to influence the performance. Most simulators, however, cannot address detailed touch calculations. The inherent delay of motors from motor commands is also a crucial problem. Many athletic behaviors such as throwing are instantaneous movements. The throwing took only 1/3 sec in this experiment. Thus the motor's slight delay makes a difference of performance. After all, it is necessary for acquisition of skilled behavior to optimize the parameters in high-dimensional space using a real robot.

6.2 Synchronization and Desynchronization in Human Skilled Behaviors

We demonstrated that optimization of the robot's throwing skill was accelerated by synchronization and desynchronization of the DoFs. The humanoid robot, consequently, could acquire the skilled throwing with less trials (see Fig. 6). The optimal throwing had asynchrony with small differences between DoFs' timing. This asynchrony of DoFs is also observed in human throwing. In the throwing by an expert the timing to maximum velocities of body parts does not always correspond to the timing of releasing of a ball [14]. In addition, skillful cyclical movements such as clay kneading [11] and samba dance [12] have the slight phase differences between body parts. These studies [11, 12] also showed that there is a process from synchronization to desynchronization of the body parts in acquisition of these periodic skills. The results reported here may suggest that the process in human motor learning has a role of reduction of the motor dimensionality and then accelerates optimization of the movement. Our study, however, does not explain how human optimizes their skills through the process. The desynchronization in human skilled behaviors may result from dynamic interaction between body parts with compliance and environment. More detailed modeling of human motor learning is necessary to expand our approach.

6.3 Possibility of Application to other Skills

The proposed optimization method were evaluated in throwing task as a case study in this paper. The optimal parameters for throwing in current experiment were $(t_s, t_w, t_k) = (-1, -1, 1)$, which implies that desynchronization with small differences was important for skilled throwing. Athletic behaviors are instantaneous movements, which can be regarded as synchronization of body parts. From a micro perspective, however, a little desynchronization of movements of each body part is required for skilled athletic behaviors (e.g. [14]). In other words, the timing optimized in synchronization of the DoFs is close to the global optimum. Thus, the local maximum reached in the first stage of exploration could be a reasonable initial estimate even if the space dimensionality increases.

We can apply the proposed method to other athletic behaviors if the tasks' optimal parameters exist around synchronized parameters. A slight differentiation of the timing of leg's DoFs may be important in high-kicking (kicking the ball as high as possible), which has been an official technical challenge in the RoboCup soccer humanoid league since 2012. We will attempt to optimize these soccer skills by applying the proposed method. In addition, velocity of body parts is also important for skilled behavior. We will address the skill acquisition with more parameters such as velocity or acceleration.

7 Conclusion

In this paper, we presented a practical optimization method through synchronization and desynchronization of a robot's body parts. All of the DoFs related to the skill were synchronized in the first stage of learning. Thus, the robot optimized the timing of the start of releasing the ball in one-dimensional space. The DoFs were then desynchronized one by one, which enabled the robot to explore the optimal timing of the start of each joint's movement. The reduction of the search space dimensionality, consequently, could decrease the number of trials. Our experiments showed that the optimization through synchronization of the DoFs resulted in as high performance as the result of optimizing without synchronization even if less trials were used.

This optimization method may be leveraged when acquiring quick movements such as throwing, kicking and so on. Instantaneous athletic skills can be synchronized behaviors. Thus the optimization of synchronized DoFs might be more plausible, i.e., not just a local solution. The robot can reach quickly a valid solution because of usage of the best solution in the previous stage.

Acknowledgment. The authors gratefully acknowledge the contribution of the team members of the JEAP, which is an acronym for JST ERATO Asada Project, for all materials, and also Associate Prof. Tomomichi Sugihara and a former student, Hirotaka Kimura (Currently, the City of Nagoya, Japan) at Osaka University for their suggestion about heuristic exploration.

References

1. Hornby, G.S., Fujita, M., Takamura, S., Yamamoto, T., Hanagata, O.: Autonomous evolution of gaits with the sony quadruped robot. In: Proc. of the Genetic and Evolutionary Computation Conference, vol. 2, pp. 1297–1304 (1999)
2. Daoxiong, G., Jie, Y., Guoyu, Z.: A review of gait optimization based on evolutionary computation. Applied Computational Intelligence and Soft Computing (2010)
3. Rong, C., Wang, Q., Huang, Y., Xie, G., Wang, L.: Autonomous evolution of high-speed quadruped gaits using particle swarm optimization. In: Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C. (eds.) RoboCup 2008. LNCS, vol. 5399, pp. 259–270. Springer, Heidelberg (2009)
4. Shafii, N., Aslani, S., Nezami, O.M., Shiry, S.: Evolution of biped walking using truncated fourier series and particle swarm optimization. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 344–354. Springer, Heidelberg (2010)
5. Kohl, N., Stone, P.: Machine learning for fast quadrupedal locomotion. In: Proc. of the 19th National Conf. on Artificial Intelligence, pp. 611–616 (2004)
6. Saggar, M., D'Silva, T., Kohl, N., Stone, P.: Autonomous learning of stable quadruped locomotion. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 98–109. Springer, Heidelberg (2007)
7. Hausknecht, M., Stone, P.: Learning powerful kicks on the aibo ERS-7: The quest for a striker. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS, vol. 6556, pp. 254–265. Springer, Heidelberg (2010)
8. Bernstein, N.A.: The co-ordination and regulation of movements. Pergamon Press (1967)
9. Newell, K.M., Vaillancourt, D.E.: Dimensional change in motor learning. Human Movement Science 20(4-5), 695–715 (2001)
10. Vereijken, B., van Emmerik, R.E.A., Whiting, H.T.A., Newell, K.M.: Free(z)ing degrees of freedom in skill acquisition. Journal of Motor Behavior 24(1), 133–142 (1992)
11. Yamamoto, T., Fujinami, T.: Hierarchical organization of the coordinative structure of the skill of clay kneading. Human Movement Science 27(5), 812–822 (2008)
12. Matsumura, K., Yamamoto, T., Fujinami, T.: A study of samba dance using acceleration sensors. In: Proc. of the 8th Motor Control and Human Skill Conference, pp. 5–4 (2007)
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
14. Reilly, T.: Science and Soccer, Routledge (1995)

Positioning to Win: A Dynamic Role Assignment and Formation Positioning System

Patrick MacAlpine, Francisco Barrera, and Peter Stone

Department of Computer Science, The University of Texas at Austin
{patmac,tank225,pstone}@cs.utexas.edu

Abstract. This paper presents a dynamic role assignment and formation positioning system used by the 2011 RoboCup 3D simulation league champion UT Austin Villa. This positioning system was a key component in allowing the team to win all 24 games it played at the competition during which the team scored 136 goals and conceded none. The positioning system was designed to allow for decentralized coordination among physically realistic simulated humanoid soccer playing robots in the partially observable, non-deterministic, noisy, dynamic, and limited communication setting of the RoboCup 3D simulation league simulator. Although the positioning system is discussed in the context of the RoboCup 3D simulation environment, it is not domain specific and can readily be employed in other RoboCup leagues as it generalizes well to many realistic and real-world multiagent systems.

1 Introduction

Coordinated movement among autonomous mobile robots is an important research area with many applications such as search and rescue [1] and warehouse operations [2]. The RoboCup 3D simulation competition provides an excellent testbed for this line of research as it requires coordination among autonomous agents in a physically realistic environment that is partially observable, non-deterministic, noisy, and dynamic. While low level skills such as walking and kicking are vitally important for having a successful soccer playing agent, the agents must work together as a team in order to maximize their game performance.

One often thinks of the soccer teamwork challenge as being about where the player with the ball should pass or dribble, but at least as important is where the agents position themselves when they *do not* have the ball [3]. Positioning the players in a formation requires the agents to coordinate with each other and determine where each agent should position itself on the field. While there has been considerable research done in the 2D soccer simulation domain (for example by Stone et al. [4] and Reis et al. [5]), relatively little outside of [6] has been published on this topic in the more physically realistic 3D soccer simulation environment. [6], as well as related work in the RoboCup middle size league (MSL) [7], rank positions on the field in order of importance and then iteratively assign the closest available agent to the most important currently unassigned position until every agent is mapped to a target location. The work presented in this paper differs from the mentioned previous work in the 2D and 3D simulation and MSL RoboCup domains as it takes into account real-world concerns and movement dynamics such as the need for avoiding collisions of robots.

In UT Austin Villa's positioning system players' positions are determined in three steps. First, a full team formation is computed (Section 3); second, each player computes the best assignment of players to role positions in this formation according to its own view of the world (Section 4); and third, a coordination mechanism is used to choose among all players' suggestions (Section 4.4). In this paper, we use the terms (player) position and (player) role interchangeably.

The remainder of the paper is organized as follows. Section 2 provides a description of the RoboCup 3D simulation domain. The formation used by UT Austin Villa is given in Section 3. Section 4 explains how role positions are dynamically assigned to players. Collision avoidance is discussed in Section 5. An evaluation of the different parts of the positioning system is given in Section 6, and Section 7 summarizes.

2 Domain Description

The RoboCup 3D simulation environment is based on SimSpark,[1] a generic physical multiagent system simulator. SimSpark uses the Open Dynamics Engine[2] (ODE) library for its realistic simulation of rigid body dynamics with collision detection and friction. ODE also provides support for the modeling of advanced motorized hinge joints used in the humanoid agents.

The robot agents in the simulation are homogeneous and are modeled after the Aldebaran Nao robot,[3] which has a height of about 57 cm, and a mass of 4.5 kg. The agents interact with the simulator by sending torque commands and receiving perceptual information. Each robot has 22 degrees of freedom: six in each leg, four in each arm, and two in the neck. In order to monitor and control its hinge joints, an agent is equipped with joint perceptors and effectors. Joint perceptors provide the agent with noise-free angular measurements every simulation cycle (20 ms), while joint effectors allow the agent to specify the torque and direction in which to move a joint. Although there is no intentional noise in actuation, there is slight actuation noise that results from approximations in the physics engine and the need to constrain computations to be performed in real-time. Visual information about the environment is given to an agent every third simulation cycle (60 ms) through noisy measurements of the distance and angle to objects within a restricted vision cone (120°). Agents are also outfitted with noisy accelerometer and gyroscope perceptors, as well as force resistance perceptors on the sole of each foot. Additionally, agents can communicate with each other every other simulation cycle (40 ms) by sending messages limited to 20 bytes.

[1] `http://simspark.sourceforge.net/`
[2] `http://www.ode.org/`
[3] `http://www.aldebaran-robotics.com/eng/`

3 Formation

This section presents the formation used by UT Austin Villa during the 2011 RoboCup competition. The formation itself is not a main contribution of this paper, but serves to set up the role assignment function discussed in Section 4 for which a precomputed formation is required.

In general, the team formation is determined by the ball position on the field. As an example, Figure 1 depicts the different role positions of the formation and their relative offsets when the ball is at the center of the field. The formation can be broken up into two separate groups, an offensive and a defensive group. Within the offensive group, the role positions on the field are determined by adding a specific offset to the ball's coordinates. The *onBall* role, assigned to the player closest to the ball, is always based on where the ball is and is therefore never given an offset. On either side of the ball are two forward roles, *forwardRight* and *forwardLeft*. Directly behind the ball is a *stopper* role as well as two additional roles, *wingLeft* and *wingRight*, located behind and to either side of the ball. When the ball is near the edge of the field some of the roles' offsets from the ball are adjusted so as to prevent them from moving outside the field of play.

Within the defensive group there are two roles, *backLeft* and *backRight*. To determine their positions on the field a line is calculated between the center of the team's own goal and the ball. Both backs are placed along this line at specific offsets from the end line. The goalie positions itself independently of its teammates in order to always be in the best position to dive and stop a shot on goal. If the goalie assumes the *onBall* role, however, a third role is included within the defensive group, the *goalieReplacement* role. A field player assigned to the *goalieReplacement* role is told to stand in front of the center of the goal.

During the course of a game there are occasional stoppages in play for events such as kickoffs, goal kicks, corner kicks, and kick-ins. When one of these events occur UT Austin Villa adjusts its team formation and behavior to assume situational set plays which are detailed in a technical report [8].

Kicking and passing have yet to be incorporated into the team's formation. Instead the *onBall* role always dribbles the ball toward the opponent's goal.

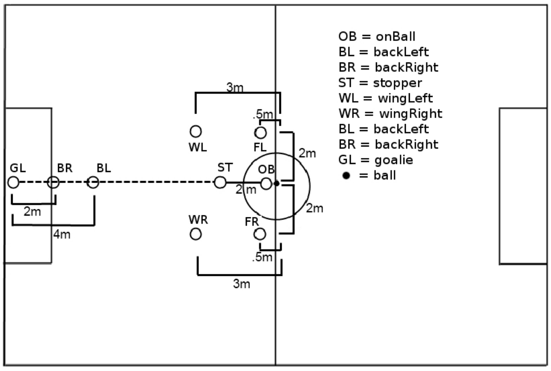


Fig. 1. Formation role positions

4 Assignment of Agents to Role Positions

Given a desired team formation, we need to map players to roles (target positions on the field). A naïve mapping having each player permanently mapped to one of the roles performs poorly due to the dynamic nature of the game. With such static roles an agent assigned to a defensive role may end up out of position and, without being able to switch roles with a teammate in a better position to defend, allow for the opponent to have a clear path to the goal. In this section, we present a dynamic role assignment algorithm. A role assignment algorithm can be thought of as implementing a role assignment *function*, which takes as input the state of the world, and outputs a one-to-one mapping of players to roles. We start by defining three properties that a role assignment function must satisfy (Section 4.1). We then construct a role assignment function that satisfies these properties (Section 4.2). Finally, we present a dynamic programming algorithm implementing this function (Section 4.3).

4.1 Desired Properties of a Valid Role Assignment Function

Before listing desired properties of a role assignment function we make a couple of assumptions. The first of these is that no two agents and no two role positions occupy the same position on the field. Secondly we assume that all agents move toward fixed role positions along a straight line at the same constant speed. While this assumption is not always completely accurate, the omnidirectional walk used by the agent, and described in [9], gives a fair approximation of constant speed movement along a straight line.

We call a role assignment function *valid* if it satisfies three properties:

1. *Minimizing longest distance* - it minimizes the maximum distance from a player to target, with respect to all possible mappings.
2. *Avoiding collisions* - agents do not collide with each other as they move to their assigned positions.
3. *Dynamically consistent* - a role assignment function f is dynamically consistent if, given a *fixed* set of target positions, if f outputs a mapping m of players to targets at time T, and the players are moving toward these targets, f would output m for every time $t > T$.

The first two properties are related to the output of the role assignment function, namely the mapping between players and positions. We would like such a mapping to minimize the time until all players have reached their target positions because quickly doing so is important for strategy execution. As we assume all players move at the same speed, we start by requiring a mapping to minimize the maximum distance any player needs to travel. However, paths to positions might cross each other, therefore we additionally require a mapping to guarantee that when following it, there are no collisions. The third property guarantees that once a role assignment function f outputs a mapping, f is committed to it as long as there is no change in the target positions. This guarantee is necessary as otherwise agents might unduly thrash between roles thus impeding progress. In the following section we construct a valid role assignment function.

4.2 Constructing a Valid Role Assignment Function

Let M be the set of all one-to-one mappings between players and roles. If the number of players is n, then there are $n!$ possible such mappings. Given a state of the world, specifically n player positions and n target positions, let the *cost* of a mapping m be the n-tuple of distances from each player to its target, sorted in decreasing order. We can then sort all the $n!$ possible mappings based on their costs, where comparing two costs is done lexicographically. Sorted costs of mappings from agents to role positions for a small example are shown in Figure 2.

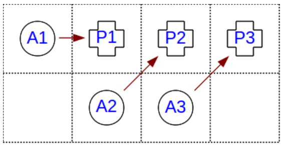


Fig. 2. Lowest lexicographical cost (shown with arrows) to highest cost ordering of mappings from agents (A1,A2,A3) to role positions (P1,P2,P3). Each row represents the cost of a single mapping.

1:	$\sqrt{2}$ (A2→P2),	$\sqrt{2}$ (A3→P3),	1 (A1→P1)
2:	2 (A1→P2),	$\sqrt{2}$ (A3→P3),	1 (A2→P1)
3:	$\sqrt{5}$ (A2→P3),	1 (A1→P1),	1 (A3→P2)
4:	$\sqrt{5}$ (A2→P3),	2 (A1→P2),	$\sqrt{2}$ (A3→P1)
5:	3 (A1→P3),	1 (A2→P1),	1 (A3→P2)
6:	3 (A1→P3),	$\sqrt{2}$ (A2→P2),	$\sqrt{2}$ (A3→P1)

Denote the role assignment function that always outputs the mapping with the lexicographically smallest cost as f_v. Here we provide an informal proof sketch that f_v is a valid role assignment; we provide a longer, more thorough derivation in a technical report [8].

Theorem 1. *f_v is a valid role assignment function.*

It is trivial to see that f_v minimizes the longest distance traveled by any agent (Property 1) as the lexicographical ordering of distance tuples sorted in descending order ensures this. If two agents in a mapping are to collide (Property 2) it can be shown, through the triangle inequality, that f_v will find a lower cost mapping as switching the two agents' targets reduces the maximum distance either must travel. Finally, as we assume all agents move toward their targets at the same constant rate, the distance between any agent and target will not decrease any faster than the distance between an agent and the target it is assigned to. This observation serves to preserve the lowest cost lexicographical ordering of the chosen mapping by f_v across all timesteps thereby providing dynamic consistency (Property 3). Section 4.3 presents an algorithm that implements f_v.

4.3 Dynamic Programming Algorithm for Role Assignment

In UT Austin Villa's basic formation, presented in Section 3, there are nine different roles for each of the nine agents on the field. The goalie always fills the *goalie* role and the *onBall* role is assigned to the player closest to the ball. The other seven roles must be mapped to the agents by f_v. Additionally, when the goalie is closest to the ball, the goalie takes on both the *goalie* and *onBall* roles causing us to create an extra *goalieReplacement* role positioned right in front of the team's goal. When this occurs the size of the mapping increases to eight agents mapped to eight roles. As the total number of mapping permutations is $n!$, this creates the possibility of needing to evaluate 8! different mappings.

Clearly f_v could be implemented using a brute force method to compare all possible mappings. This implementation would require creating up to $8! = 40,320$ mappings, then computing the cost of each of the mappings, and finally sorting them lexicographically to choose the smallest one. However, as our agent acts in real time, and f_v needs to be computed during a decision cycle (20 ms), a brute force method is too computationally expensive. Therefore, we present a dynamic programming implementation shown in Algorithm 1 that is able to compute f_v within the time constraints imposed by the decision cycle's length.

Algorithm 1. Dynamic programming implementation

```
 1: HashMap $bestRoleMap = \varnothing$
 2: $Agents = \{a_1, ..., a_n\}$
 3: $Positions = \{p_1, ..., p_n\}$
 4: for $k = 1$ to $n$ do
 5:     for each $a$ in $Agents$ do
 6:         $S = \binom{n-1}{k-1}$ sets of $k-1$ agents from $Agents - \{a\}$
 7:         for each $s$ in $S$ do
 8:             Mapping $m_0 = bestRoleMap[s]$
 9:             Mapping $m = (a \rightarrow p_k) \cup m_o$
10:             $bestRoleMap[\{a\} \cup s] = mincost(m, bestRoleMap[\{a\} \cup s])$
11: return $bestRoleMap[Agents]$
```

Theorem 2. *Let A and P be sets of n agents and positions respectively. Denote the mapping $m := f_v(A, P)$. Let m_0 be a subset of m that maps a subset of agents $A_0 \subset A$ to a subset of positions $P_0 \subset P$. Then m_0 is also the mapping returned by $f_v(A_0, P_0)$.*

A key recursive property of f_v that allows us to exploit dynamic programming is expressed in Theorem 2. This property stems from the fact that if within any subset of a mapping a lower cost mapping is found, then the cost of the complete mapping can be reduced by augmenting the complete mapping with that of the subset's lower cost mapping. The savings from using dynamic programming comes from only evaluating mappings whose subset mappings are returned by f_v. This is accomplished in Algorithm 1 by iteratively building up optimal mappings for position sets from $\{p_1\}$ to $\{p_1, ..., p_n\}$, and using optimal mappings of $k-1$ agents to positions $\{p_1, ..., p_{k-1}\}$ (line 8) as a base when constructing each new mapping of k agents to positions $\{p_1, ..., p_k\}$ (line 9), before saving the lowest cost mapping for the current set of k agents to positions $\{p_1, ..., p_k\}$ (line 10).

An example of the mapping combinations evaluated in finding the optimal mapping for three agents through the dynamic programming approach of Algorithm 1 can be seen in Table 1. In this example we begin by computing the distance of each agent to our first role position. Next we compute the cost of all possible mappings of agents to both the first and second role positions and save off the lowest cost mapping of every pair of agents to the the first two positions. We then proceed by sequentially assigning every agent to the third position and compute the lowest cost mapping of all agents mapped to all three positions. As all subsets of an optimal (lowest cost) mapping will themselves be optimal, we need only evaluate mappings to all three positions which include the previously calculated optimal mapping agent combinations for the first two positions.

Table 1. All mappings evaluated during dynamic programming using Algorithm 1 when computing an optimal mapping of agents A1, A2, and A3 to positions P1, P2, and P3. Each column contains the mappings evaluated for the set of positions listed at the top of the column.

{P1}	{P2,P1}	{P3,P2,P1}
A1→P1	A1→P2, f_v(A2→P1)	A1→P3, f_v({A2,A3}→{P1,P2})
A2→P1	A1→P2, f_v(A3→P1)	A2→P3, f_v({A1,A3}→{P1,P2})
A3→P1	A2→P2, f_v(A1→P1)	A3→P3, f_v({A1,A2}→{P1,P2})
	A2→P2, f_v(A3→P1)	
	A3→P2, f_v(A1→P1)	
	A3→P2, f_v(A2→P1)	

Recall that during the kth iteration of the dynamic programming process to find a mapping for n agents, where k is the current number of positions that agents are being mapped to, each agent is sequentially assigned to the kth position and then all possible subsets of the other $n-1$ agents are assigned to positions 1 to $k-1$ based on computed optimal mappings to the first $k-1$ positions from the previous iteration of the algorithm. These assignments result in a total of $\binom{n-1}{k-1}$ agent subset mapping combinations to be evaluated for mappings of each agent assigned to the kth position. The total number of mappings computed for each of the n agents across all n iterations of dynamic programming is thus equivalent to the sum of the $n-1$ binomial coefficients. That is,

$$\sum_{k=1}^{n}\binom{n-1}{k-1}=\sum_{k=0}^{n-1}\binom{n-1}{k}=2^{n-1}$$

Therefore the total number of mappings that must be evaluated using our dynamic programming approach is $n2^{n-1}$. For $n = 8$ we thus only have to evaluate 1024 mappings which takes about 3.3 ms for each agent to compute compared to upwards of 50 ms using a brute force approach to evaluate all possible mappings.[4]

4.4 Voting Coordination System

In order for agents on a team to assume correct positions on the field they all must coordinate and agree on which mapping of agents to roles to use. If every agent had perfect information of the locations of the ball and its teammates this would not be a problem as each could independently calculate the optimal mapping to use. Agents do not have perfect information, however, and are limited to

[4] As measured on an Intel Core 2 Duo CPU E8500 @3.16GHz.

noisy measurements of the distance and angle to objects within a restricted vision cone (120°). Fortunately agents can share information with each other every other simulation cycle (40 ms). The bandwidth of this communication channel is very limited, however, as only one agent may send a message at a time and messages are limited to 20 bytes.

We utilize the agents' limited communication bandwidth in order to coordinate role mappings as follows. Each agent is given a rotating time slice to communicate information, as in [4], which is based on the uniform number of an agent. When it is an agent's turn to send a message it broadcasts to its teammates its current position, the position of the ball, and also what it believes the optimal mapping should be. By sending its own position and the position of the ball, the agent provides necessary information for computing the optimal mapping to those of its teammates for which these objects are outside of their view cones. Sharing the optimal mapping of agents to role positions enables synchronization between the agents, as follows.

First note that just using the last mapping received is dangerous, as it is possible for an agent to report inconsistent mappings due to its noisy view of the world. This can easily occur when an agent falls over and accumulates error in its own localization. Additionally, messages from the server are occasionally dropped or received at different times by the agents preventing accurate synchronization. To help account for inconsistent information, a sliding window of received mappings from the last n time-slots is kept by each agent where n is the total number of agents on a team. Each of these kept messages represents a single vote by each of the agents as to which mapping to use. The mapping chosen is the one with the most votes or, in the case of a tie, the mapping tied for the most votes with the most recent vote cast for it. By using a voting system, the agents on a team are able to synchronize the mapping of agents to role positions in the presence of occasional dropped messages or an agent reporting erroneous data. As a test of the voting system the number of cycles all nine agents shared a synchronized mapping of agents to roles was measured during 5 minutes of gameplay (15,000 cycles). The agents were synchronized 100% of the time when using the voting system compared to only 36% of the time when not using it.

5 Collision Avoidance

Although the positioning system discussed in Section 4 is designed to avoid assigning agents to positions that might cause them to collide, external factors outside of the system's control, such as falls and the movement of the opposing team's agents, still result in occasional collisions. To minimize the potential for these collisions the agents employ an active collision avoidance system. When an obstacle, such as a teammate, is detected in an agent's path the agent will attempt to adjust its path to its target in order to maneuver around the obstacle. This adjustment is accomplished by defining two thresholds around obstacles: a *proximity* threshold at 1.25 meters and a *collision* threshold at .5 meters from an obstacle. If an agent enters the *proximity* threshold of an obstacle it will adjust its course to be tangent to the obstacle thereby choosing to circle around to the right or left of said obstacle depending on which direction will move the agent closer to its desired target. Should the agent get so close as to enter the *collision* proximity of an obstacle it must take decisive action to prevent an otherwise imminent collision from occurring. In this case the agent combines the corrective movement brought about by being in the *proximity* threshold with an additional movement vector directly away from the obstacle. Figure 3 illustrates the adjusted movement of an agent when attempting to avoid a collision.

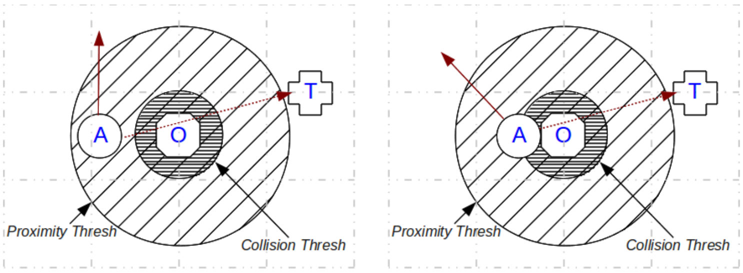


Fig. 3. Collision avoidance examples where agent A is traveling to target T but wants to avoid colliding with obstacle O. The left diagram shows how the agent's path is adjusted if it enters the *proximity* threshold of the obstacle while the right diagram depicts the agent's movement when entering the *collision* threshold. The dotted arrow is the agent's desired path while the solid arrow is the corrected path to avoid a collision.

6 Formation Evaluation

To test how our formation and role positioning system[5] affects the team's performance we created a number of teams to play against by modifying the base positioning system and formation of UT Austin Villa.

UT Austin Villa. Base agent using the dynamic role positioning system described in Section 4 and formation in Section 3.
NoCollAvoid. No collision avoidance.
AllBall. No formations and every agent except for the goalie goes to the ball.
NoTeamwork. Similar to AllBall except that collision avoidance is also turned off.
NoCommunication. Agents do not communicate with each other.
Static. Each role is statically assigned to an agent based on its uniform number.
Defensive. Defensive formation in which only two agents are in the offensive group.
Offensive. Offensive formation in which all agents except for the goalie are positioned in a close symmetric formation behind the ball.
Boxes. Field is divided into fixed boxes and each agent is dynamically assigned to a home position in one of the boxes. Similar to system used in [4].
NearestStopper. The *stopper* role position is mapped to nearest agent.

[5] Video demonstrating our positioning system can be found online at `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2011/html/positioning.html`

PathCost. Agents add in the cost of needing to walk around known obstacles (using collision avoidance from Section 5), such as the ball and agent assuming the *onBall* role, when computing distances of agents to role positions.

PositiveCombo. Combination of *Offensive*, *PathCost*, and *NearestStopper* attributes.

Table 2. Full game results, averaged over 100 games. Each row corresponds to an agent with varying formation and positioning systems as described in Section 6. Entries show the goal difference (row − column) from 10 minute games versus our base agent, using the dynamic role positioning system described in Section 4 and formation in Section 3, as well as the Apollo3D and CIT3D agents from the 2011 RoboCup China Open. Values in parentheses are the standard error.

	UTAustinVilla	Apollo3D	CIT3D
PositiveCombo	0.33 (.07)	2.16 (.11)	4.09 (.12)
Offensive	0.21 (.09)	1.80 (.12)	3.89 (.12)
AllBall	0.09 (.08)	1.69 (.13)	3.56 (.13)
PathCost	0.07 (.07)	1.27 (.11)	3.25 (.11)
NearestStopper	0.01 (.07)	1.26 (.11)	3.21 (.11)
UTAustinVilla	—	1.05 (.12)	3.10 (.12)
Defensive	-0.05 (.05)	0.42 (.10)	1.71 (.11)
Static	-0.19 (.07)	0.81 (.13)	2.87 (.11)
NoCollAvoid	-0.21 (.08)	0.82 (.12)	2.84 (.12)
NoCommunication	-0.30 (.06)	0.41 (.11)	1.94 (.10)
NoTeamwork	-1.10 (.11)	0.33 (.15)	2.43 (.12)
Boxes	-1.38 (.11)	-0.82 (.13)	1.52 (.11)

Results of UT Austin Villa playing against these modified versions of itself are shown in Table 2. The UT Austin Villa agent is the same agent used in the 2011 competition, except for a bug fix,[6] and so the data shown does not directly match with earlier released data in [9]. Also shown in Table 2 are results of the modified agents playing against the champion (Apollo3D) and runner-up (CIT3D) of the 2011 RoboCup China Open. These agents were chosen as reference points as they are two of the best teams available with CIT3D and Apollo3D taking second and third place respectively at the main RoboCup 2011 competition. The China Open occurred after the main RoboCup event during which time both teams improved (Apollo3D went from losing by an average of 1.83 to 1.05 goals and CIT3D went from losing by 3.75 to 3.1 goals on average when playing 100 games against our base agent).

Several conclusions can be made from the game data in Table 2. The first of these is that it is really important to be aggressive and always have agents near the ball. This finding is shown in the strong performance of the *Offensive* agent. In contrast to an offensive formation, we see that a very defensive formation used by the *Defensive* agent hurts performance likely because, as the saying goes, the best defense is a good offense. The poor performance of the *Boxes* agent, in which the positions on the field are somewhat static and not calculated as relative offsets to the ball, underscores the importance of being around the ball and adjusting positions on the field based on the current state of the game. The likely reason for the success of offensive and aggressive formations grouped

[6] A bug in collision avoidance present in the 2011 competition agent where it always moved in the direction away from the ball to avoid collisions was fixed.

close to the ball is because few teams in the league have managed to successfully implement advanced passing strategies, and thus most teams primarily rely on dribbling the ball. Should a team develop good passing skills then a spread out formation might become useful.

The *NearestStopper* agent was created after noticing that the *stopper* role is a very important position on the field so as to always have an agent right behind the ball to prevent breakaways and block kicks toward the goal. Ensuring that the *stopper* role is filled as quickly as possible improved performance slightly. This result is another example of added aggression improving game performance.

Another factor in team performance that shows up in the data from Table 2 is the importance of collision avoidance. Interestingly the *AllBall* agent did almost as well as the *Offensive* agent even though it does not have a set formation. While this result might come as a bit of surprise, collision avoidance causes the *AllBall* agent to form a clumped up mass around the ball which is somewhat similar to that of the *Offensive* agent's formation. For the strategy of all the agents running to the ball to work well it is imperative to have good collision avoidance. This conclusion is evident from the poor performance of the *NoTeamwork* agent where collision avoidance is turned off with everyone running to the ball, as well as from a result in [9] where the *AllBall* agent lost to the base agent by an average of .43 goals when both agents had a bug in their collision avoidance systems. Turning off collision avoidance, but still using formations, hurts performance as seen in the results of the *NoCollAvoid* agent. Additionally the *PathCost* agent showed an improvement in gameplay by factoring in known obstacles that need to be avoided when computing the distance required to walk to each target.

Another noteworthy observation from the data in Table 2 is that dynamically assigning roles is better than statically fixing them. This finding is clear in the degradation in performance of the *Static* agent. It is important that the agents are synchronized in their decision as to which mapping of agents to roles to use, however, as is noticeable by the dip in performance of the *NoCommunication* agent which does not use the voting system presented in Section 4.4 to synchronize mappings. The best performing agent, that being the *PositiveCombo* agent, demonstrates that the most successful agent is one which employs an aggressive formation coupled with synchronized dynamic role switching, path planning, and good collision avoidance. While not shown in Table 2, the *PositiveCombo* agent beat the *AllBall* agent (which only employs collision avoidance and does not use formations or positioning) by an average of .31 goals across 100 games with a standard error of .09. This resulted in a record of 43 wins, 20 losses, and 37 ties for the *PositiveCombo* agent against the *AllBall* agent.

7 Summary and Discussion

We have presented a dynamic role assignment and formation positioning system for use with autonomous mobile robots in the RoboCup 3D simulation domain — a physically realistic environment that is partially observable, non-deterministic, noisy, and dynamic. This positioning system was a key component in UT Austin Villa[7] winning the 2011 RoboCup 3D simulation league competition.

For future work we hope to add passing to our strategy and then develop formations for passing, possibly through the use of machine learning. Additionally we intend to look into ways to compute f_v more efficiently as well as explore other potential functions for mapping agents to role positions.

Acknowledgments. This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. Thanks especially to UT Austin Villa 2011 team members Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan, Michael Quinlan, Nick Collins, Adrian Lopez-Mobilia, Art Richards, Nicolae Ştiurcă, and Victor Vu. LARG research is supported in part by grants from the National Science Foundation (IIS-0917122), ONR (N00014-09-1-0658), and the Federal Highway Administration (DTFH61-07-H-00030). Patrick MacAlpine is supported by a NDSEG fellowship.

References

1. Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., Shimada, S.: Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In: Proc. of 1999 IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), vol. 6, pp. 739–743 (1999)
2. Wurman, P.R., D'Andrea, R., Mountz, M.: Coordinating hundreds of cooperative, autonomous vehicles in warehouses. AI Magazine 29, 9–20 (2008)
3. Kalyanakrishnan, S., Stone, P.: Learning complementary multiagent behaviors: A case study. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 153–165. Springer, Heidelberg (2010)
4. Stone, P., Veloso, M.: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence 110, 241–273 (1999)
5. Reis, L.P., Lau, N., Oliveira, E.C.: Situation based strategic positioning for coordinating a team of homogeneous agents. In: Hannebauer, M., Wendler, J., Pagello, E. (eds.) Reactivity and Deliberation in MAS. LNCS (LNAI), vol. 2103, pp. 175–197. Springer, Heidelberg (2001)
6. Chen, W., Chen, T.: Multi-robot dynamic role assignment based on path cost. In: 2011 Chinese Control and Decision Conference (CCDC), pp. 3721–3724 (2011)
7. Lau, N., Lopes, L., Corrente, G., Filipe, N.: Multi-robot team coordination through roles, positionings and coordinated procedures. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), pp. 5841–5848 (2009)
8. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011 3D Simulation Team report. Technical Report AI11-10, The Univ. of Texas at Austin, Dept. of Computer Science, AI Laboratory (2011)
9. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012) (2012)

[7] More information about the UT Austin Villa team, as well as video highlights from the 2011 competition, can be found at the team's website:
`http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/`

Evacuation Simulation with Guidance for Anti-disaster Planning

Masaru Okaya and Tomoichi Takahashi

Meijo University Shiogamaguchi 1-501, Tenpaku, Nagoya, Japan, 468-8502
m0930007@ccalumni.meijo-u.ac.jp, ttaka@ccmfs.meijo-u.ac.jp
http://sakura.meijo-u.ac.jp/ttakaHP/Rescue_index.html

Abstract. Crowd evacuation simulations are useful tools for analyzing and assessing the safety of building occupants. Agent-based simulations provide a platform for computing individual as well as and collective behaviors in crowds. During an evacuation, it is well known that trained leaders or evacuation guidance play a key role in saving human lives. In this paper, we propose an evacuation simulation system where agents are guided by evacuation orders from authorities. The simulations captured typical behaviors observed during crowd evacuation. For example, the total evacuation time was reduced when most of the agents followed the guidance, although the evacuation times of individual agents were different. When a specific agent is involved in the movement of other agents to a different destination, the evacuation takes a longer amount of time. The simulation appears to depict real-life situations well, which shows that simulations can be a useful tool to estimate evacuation situations prior to emergency evacuation drills.

Keywords: Evacuation, Guidance, BDI model, Disaster prevention planning.

1 Introduction

In the aftermath of Hurricane Katrina and the September 11 attacks, evacuation simulations have been explored for their potential in decreasing the amount of damage resulting from disasters, and in particular, saving human lives. There are different types of evacuation behaviors, and several factors exist that might influence the amount of damage and degree of injury incurred. The Great East Japan Earthquake that occurred on March 11, 2011, along with the resulting tsunami, caused serious damage and injury. During this disaster, teachers guided their students to specific locations that they thought were safe. During the evacuations, some teachers were told that their destination was not safe, and therefore, they guided their students to another location. However, in some instances, they did not have enough time to reach their new destination.

Evacuation guidance has an important influence on evacuation behavior. Guidance from well-trained leaders can facilitate efficient evacuation [1]. The evacuation might suddenly change when evacuees receive different information from

Fig. 1. Types of disasters that can result in change in evacuation behaviors. In the case of the WTC attacks on 9.11, many of the occupants escaped from the buildings. In the case of the Great East Japan Earthquake and tsunami, many people moved to a higher spot.

beliefs that they have, for example, by seeing that other evacuee groups move to different refugees or by reading exit signs that indicate other directions. Evacuees must then decide whether they continue their actions or trust the new information and change the actions. In the above example of the Great East Japan Earthquake, the teachers changed their destination when they heard that tsunami was coming.

In this paper, we propose an agent-based evacuation simulation system that guidance information is announced to agents. The guidance is implemented as communication between authorities and communication among civilians. The remainder of this paper is organized as follows. Related works are introduced in Section 2. Section 3 describes the architecture of the evacuation system, which comprises the belief-desire-intention (BDI) model that represents the mental status of the agents, and crowd behavior models in which evacuation information is considered. The simulation scenarios and results are discussed in Sections 4 and 5. Finally, a summary is provided in Section 6.

2 Related Works

The purposes of an evacuation simulation are to assess the evacuation time and provide important information for improving an evacuation. To assess the evacuation time, a detailed analysis of the behavior during an evacuation is required. National Institute of Standards and Technology (NIST) organized evacuation situation of WTC disaster through interviews and questionnaires. They simulate the evacuation situation of WTC with EXODUS, EXIT89, Simulex and ELVAC. Table 1 shows the issues discussed in the NIST report and comparison to actual works [2]. These issues can be categorized according to the agent level.

2.1 Individual Agent

At this level, only the agent's own properties affect their actions.

Table 1. Issues of Evacuation Simulation in NIST report

Agent level	Issues	EXODUS	EXIT89	Simulex	ELVAC
Individual	Individual travel speed	✓	✓	✓	✓
	Physical limitation	✓			
Interactive	Psychological elements				
	Communication among evacuees				
Social	Evacuation delay	*	*	*	
	Group formation				
	Evacuation guidance				
	Information seeking				

*some of the issue are taken into consideration.

Individual Travel Speed Model: It is well known that congestion of human flows occurs at emergencies. For example, when they evacuate though a narrow space, rescue teams rushing to a building may collide against people who are evacuating from the building, and at staircase landings where people from the upper and lower floors merge together. Helbing et al. proposed a particle model that can simulate these types of situations [3].

Physical Limitation: Various types of obstacles can be encountered in disaster situations, such as debris, smoke, heat, and water. These obstacles pose a threat to safety and prevent a smooth evacuation. The chosen evacuation destination and route can also affect the behaviors of evacuees. In addition, some people may stop to rest during evacuation.

2.2 Interactive Agent

At this level, their surroundings and their state of mind can affect the actions of evacuees. They may also communicate and share information. Agent-based simulation (ABSs) provide a platform for computing individual and collective behaviors that occur in crowds [4]

Psychological Elements: Some people who do not begin evacuating immediately after emergencies occur may evacuate when they see others heading for refuge or loud noises at the disaster sites can make them anxious. The psychological status and agent knowledge on emergencies affect the choice of actions [5]

Communication Among Evacuees: Psychological factors can also influence the behaviors of evacuees, including their walking speed or communication with other victims. One such communication is when a person urges others in the area to evacuate.

2.3 Social Agent

The social agent is related to behaviors related to a social context or common sense of their community.

Evacuation Delay: An evacuation delay occurs when evacuees perform a number of activities before they start evacuations. These activities include gathering personal belongings, milling with other occupants, seeking additional information, and calling family members or friends. These activities may delay the start of their evacuation.

Group Formation: Guidance from well-trained leaders allows an evacuation to flow smoothly [6]. Schools drill their students to follow the instructions of their teachers and evacuate together. At the time of a disaster, people may evacuate under various scenarios, and various factors in these scenarios can result in people forming or breaking away from a group.

Evacuation Guidance: During the WTC disaster, announcements affected the evacuation behaviors of the building occupants. Proper announcements save lives, whereas incorrect announcements can increase the amount of damage resulting from a disaster. The behaviors of occupants will be changed how well information is gathered to a rescue headquarter and how well guidance is announced.

Information Seeking: People unfamiliar with the building will want to know how they can exit. They will look for iconic warning signs, exchange information with people nearby, or follow other persons who appear to be evacuating. The sensor data change the metal state, and sometime make them anxious. The perception abilities or behavior patterns of evacuees change according to their psychological states.

Recently, human relationship among agents has been taken into consideration in MAS [7] [8]. Evacuation guidance that changes the behavior of agents is strongly linked to evacuation efficiency. These behaviors are not considered enough in existing researches. In this paper, we focus on the effect of guidance on evacuation. We assume that an evacuation simulation should be used for assessing the effectiveness of evacuation guidance.

2.4 Significance of Evacuation Guidance

Methods used for receiving evacuation guidance include broadcasts, voice guidance, and electric signs. Each method of communication has a different effect. Evacuation guidance is important for following reasons. An evacuation simulator should have the ability to take these into consideration.

Evacuation Guidance for Visitors: At a large event site, most of participants are less familiar with the place than occupants. Guidance such as evacuation routes should be properly provided to them.

Recognition of Danger: In WTC disaster, most of occupants start to evacuate after gathering personal belongings. It means that they have not noticed the immediate crisis of the disaster. Making the danger clear changes their psychological status, and they recognize need of immediate evacuation.

Evacuation Guidance for Efficient Evacuation: Phased evacuation, under certain circumstances, moves occupants most at risk to a place of relative safety much more quickly and with less total impact upon building tenants than full building evacuation. The phased evacuation had been carried out during the WTC disaster.

Evacuation Guidance According to the Situations: In most cases, the occupants of a building know the location of the evacuation site and the escape route. Evacuation guidance is important when the situations change or something unexpected happens, such as an evacuation route being rendered impassable by rubble. The evacuees might receive differing or conflicting guidance. It can be assumed that an authority knows the appropriate evacuation routes more than a civilian during a disaster situation. Evacuees naturally prefer to act on information heard directly from an authority rather than on information from messages displayed on bulletin boards. They then have to act either on the new information or on the existing guidance. Furthermore, there are many different types of evacuation signage used.

3 Evacuation Guidance and Behavior Models

3.1 Language Model and Loss of Data in Communication

It is assumed that evacuation guidance will be spread among evacuees. The evacuees might tell and ask others some information. The evacuation message contains information regarding to the evacuation destination and an appropriate evacuation route. They may be secondhand information.

Some information broadcast over a loudspeaker might not spread to all evacuees by the noises of surroundings or the damaged announcement system. Disasters can disable the emergency communication systems in buildings. When an evacuee hears only a portion of the evacuation guidance, the evacuee might misunderstand some of the contents. Rumors also belong to this type of communication. Some civilians might therefore prefer to trust only information from an authority figure. Others will trust their neighbors or heed messages sent from their families.

3.2 BDI Model Representing Psychological Status

The evacuation guidance whether it is complete one or partial one, they change their psychological status. The status of agents affects the behavior of their evacuations and it can be categorized as “awareness of danger”, “strong awareness of danger”, or a “normal state”. The degree of awareness of danger differs among different people. These differences influence their behaviors, such as gathering their personal belongings or immediately fleeing the area. Belief-Desire- Intention (BDI) model is adapted to represent such behaviors.

Belief: An awareness of danger is represented as Belief in the BDI model. For instance, the belief of an evacuee will be generated when he/she senses danger or hears evacuation instructions. In the case of an earthquake, all agents share the belief that a large shaking occurred. A belief in the “awareness of danger" or "strong awareness of danger" will be generated as a response to the mindset of an agent. Some agents who do not feel danger might do so when they hear evacuation instructions.

Desire: Most people are in the middle of an activity when a disaster occurs. They may have the desire to finish the activity. Of course, they may have desire to shirk away from the risk. The agent thus has to choose a desire when they have multiple options.

Intention: Most people are doing an activity, which they will finish in some minutes. An agent might have intention to evacuate.

4 Evacuation Scenarios and Simulations

4.1 Prototype System and Agent Behavior Model

Figure 2 shows the architecture of our system. The agents in the left part send their own properties to the crowd simulator at the start time and to their targets during each sense-reason-action cycle. The target is the position according to their intentions which is selected by their BDI models. The crowd simulator calculates the movements of the agents using an equation. The micro simulation step of the crowd simulation, $\Delta\tau(\approx 0.1s)$, is finer than the step of sense-reason-action cycle, $\Delta t(\approx 1s)$. The results of the micro-simulation are returned to every agent along with the agent's own position and the positions of other visible agents.

RoboCup Rescue Simulation v.1 (RCRS) was used as the platform of our system [9]. The RCRS was used to comprehensively simulate agent behavior during a simulated disaster environment, and supports two types of agents: a civilian agent and an authority agent.

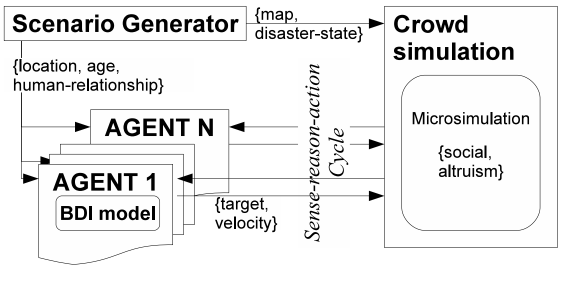


Fig. 2. Architecture of BDI-based crowd evacuation system

4.2 Communication

Message Containing Guidance from an Authority. An authority provides evacuation guidance, including information on an evacuation destination and an appropriate route to that location. Communication language is based on Agent Communication Language (ACL). The messages of evacuation guidance consist of the target person and an evacuation route. Table 2 shows evacuation instructions in which an authority guides evacuees at 1F to R1 by way of A1 and A2. The left column corresponds to the message that consists of target area and evacuation route information. The right is a message without route information and corresponds to a situation in which agents hear part of guidance.

Table 2. Evacuation guidance

complete message	message with loss of data
`(inform`	`(inform`
`  :sender Authority`	`  :sender Authority`
`  :receiver Anonymous`	`  :receiver Anonymous`
`  :time 20110311-100000`	`  :time 20110311-100000`
`  :content`	`  :content`
`    (evacuation-guidance`	`    (evacuation-guidance`
`      :target-area 1F`	`      :target-area 1F`
`      :move A1-A2-R1`	`    )`
`    )`	`)`
`)`	

4.3 Implementation of Communication

Voice and radio were implemented as communication methods in the RCRS. Voice communication is audible to anyone near the sender. During voice communication, the distance up to which the sender can be heard is 30 m. Radio communication is accessible to any person with a radio tuned to the same channel as the sender, allowing them to hear the message. We added a communication protocol with evacuation guidance messages through voice communication.

5 Simulation Scenarios and Results

We simulated three scenarios including evacuation guidance. Situations in which the agents hear a portion of evacuation guidance was simulated.

5.1 Simulation Scenarios

Figure 3 shows a building at our university. 400 people are evacuated from the building, which has 2 stairwells and 2 exits. Table 4 shows the three scenarios. Differences of scenarios are with/without evacuation guidance, agent types, with/without loss of communication. Without the evacuation guidance, the entire agent normally goes out of the front entrance because they do not know the emergency exit. Authority agent announces evacuation guidance after 5 minutes

Table 3. Evacuation guidance: contents are different for each floors

Stair	Content of evacuation guidance
1F	exit
2F	emergency stair [2F-1F] - emergency exit
3F	stair [3F-1F] - exit
4F	emergency stair [4F-1F] - emergency exit

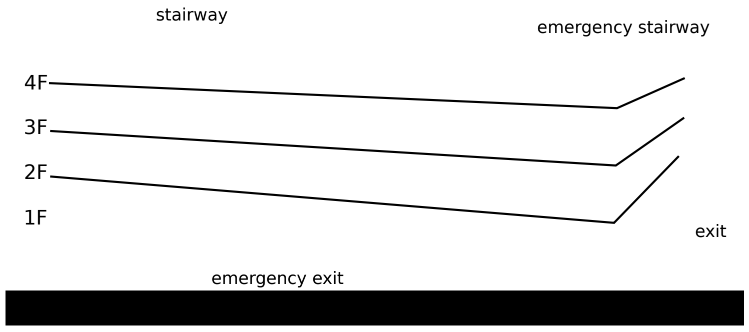


Fig. 3. Simulation map

later with building broadcasting. Contents of the guidance differ according to floors. Agents who are on the first and third floor use front stairway and front entrance, agents who are on the second and fourth floor use emergency stairway and emergency exit(Table3).

Three types of agent were implemented.

A (instant evacuation) This agent feels anxious after feeling a large shaking.
B (evacuation after tasks) This agent does not feel anxious after sensing a large shaking. This agent evacuates after a certain activity. This agent feels anxious when hearing the evacuation guidance.
C (emergent evacuation) This agent does not feel anxious after sensing a large shaking. This agent does not evacuate after a certain activity. This agent feels anxious when hearing the evacuation guidance.

Table 4. Simulation scenarios

Scenario	Guidance	Agent type	Communication
1	✓	B	no loss
		B	no loss
2	✓	A+B+C	no loss
		A+B+C	no loss
3	✓	A+B+C	loss
		A+B+C	loss

Cases of loss of communication have been simulated. The rate of loss in the guidance messages was decided according to reports of the Great East Japan Earthquake [10]. 82 % percent of agents who hear the guidance will hear the announcement of the guidance, and 82 % percent of the agent listen to the evacuation route information in the guidance and recognize the danger. So 56 % of agents start to evacuate.

5.2 Simulation Results

Figure 4 shows the simulation result of Scenario 1. Totally evacuation time in case of scenario with guidance is shorter than that of scenario without guidance. Furthermore, in case of evacuation with guidance, it takes 1600[s] for all agents who used emergency exit, while it takes 900[s] for all agents who used front entrance. It means that more efficient guidance can be considered.

Figure 5 shows comparison of simulation results of Scenario 1, 2 and 3. In case of evacuation without the evacuation guidance of Scenario 2 and 3, some agents who did not recognize the danger did not evacuate. In a case of Scenario 3, agents who came out of the front entrance are more than the others. It is because that agent who did not hear the guidance decided his/her intention by themselves. As a result of that, it took them more time evacuate than the others who heard the guidance.

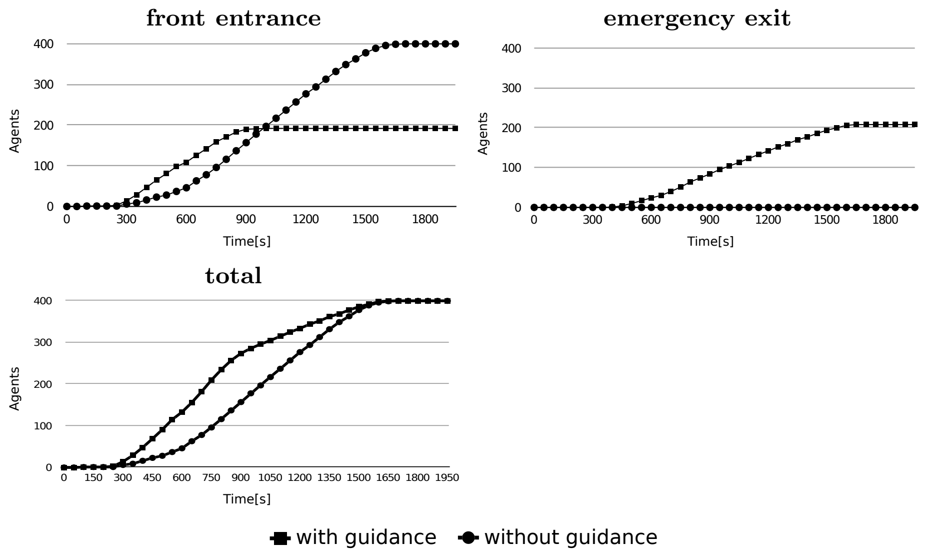


Fig. 4. Simulation result of Scenario 1. Agents who came out of each of the exit. And total agents who exit the building.

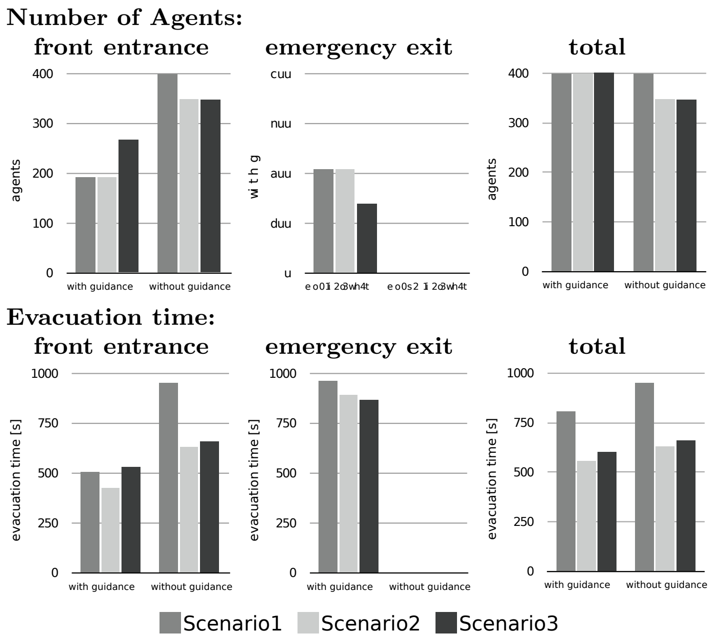


Fig. 5. Simulation result of Scenario 1, 2 and 3

6 Summary

The analysis of building evacuation has recently received an increasing amount of attention as people are keen to assess the safety of occupants. Agent-based simulation systems, such as RCRS, not only provide a platform for computing individual and collective behaviors in crowds but also their communication model supports the announcement of evacuation guidance to agents. The guidance affect the behaviors of agents, especially the delay in evacuations are closely to human lives.

In our system, the announcement of guidance is implemented as communication to agents. And the messages are modeled as a form of ACL. Agents who hear the guidance partially are modeled as they receive missing messages. When agents do not hear clearly the guidance, they behave different from ones who hear the entire message. As a result, our system can simulate the behavior of agents who do not follow evacuation guidance. We also use BDI model to represent the psychological status of agents. In our simulation system, the changes of BDI states that are caused by sensor data affect their evacuation behavior at emergencies. This makes it possible to simulate the behavior of evacuation with guidance information.

These results demonstrate that our simulator have the ability to take these scenarios which contains evacuation guidance into consideration and reconstruct these situations.

References

1. Pelechano, N.I.B.N.: Modeling crowd and trained leader behavior during building evacuation. IEEE Computer Graphics and Applications 26(6), 80–86 (2006)
2. Kuligowski, E.D.: Review of 28 egress models. In: NIST SP 1032; Workshop on Building Occupant Movement During Fire Emergencies (2005)
3. Kaup, D.J., Lakoba, T.I., Finkeistein, N.M.: Modifications of the helbing-molnar-farkas-vicsek social force model for pedestrian evolution. Simulation 81(5), 339–352 (2005)
4. Thalmann, D., Musse, S.R.: Crowd Simulation. Springer (2007)
5. Pan, X.: Computational modeling of human and social behaviors for emergency egress analysis. Ph.D. dissertation, Stanford (2006), `http://eil.stanford.edu/xpan/`
6. Pelechano, N., Allbeck, J.M., Badler, N.I.: Controlling individual agents in high-density crowd simulation. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA 2007, pp. 99–108. Eurographics Association, Switzerland (2007), `http://portal.acm.org/citation.cfm?id=1272690.1272705`
7. Okaya, M., Takahashil, T.: Bdi agent model based evacuation simulation. In: AAMAS Demo (2011)
8. Tsai, J., Tambe, M.: Escapes - evacuation simulation with children, authorities, parents, emotions, and social comparison. In: AAMAS (2011)
9. Cameron Skinner, S.R.: The robocup rescue simulation platform. In: Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010) (2010)
10. Government of Japan, A report on the great east japan earthquake, `http://www.bousai.go.jp/jishin/chubou/higashinihon/7/index.html` (in Japanese)

Motion Capture and Contemporary Optimization Algorithms for Robust and Stable Motions on Simulated Biped Robots

Andreas Seekircher, Justin Stoecker, Saminda Abeyruwan, and Ubbo Visser

University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146 USA
{aseek,justin,saminda,visser}@cs.miami.edu

Abstract. Biped soccer robots have shown drastic improvements in motion skills over the past few years. Still, a lot of work needs to be done with the RoboCup Federation's vision of 2050 in mind. One goal is creating a workflow for quickly generating reliable motions, preferably with inexpensive and accessible hardware. Our hypothesis is that using Microsoft's Kinect sensor in combination with a modern optimization algorithm can achieve this objective. We produced four complex and inherently unstable motions and then applied three contemporary optimization algorithms (CMA-ES, xNES, PSO) to make the motions robust; we performed 900 experiments with these motions on a 3D simulated Nao robot with full physics. In this paper we describe the motion mapping technique, compare the optimization algorithms, and discuss various basis functions and their impact on the learning performance. Our conclusion is that there is a straightforward process to achieve complex and stable motions in a short period of time.

1 Introduction and Related Work

Generating motions on a humanoid robot that operates under the constraints of physics is a time-consuming process; attempts to create even simple motions by manually adjusting parameters are tedious and often end in failure [16]. Our idea is to use motion capture to record and map human motions to a humanoid robot. The immediate problem with this approach is that humans and robots do not share motor capabilities, range of motion, dimensions, mass, and other physical attributes. For the purposes of our experiments, we assume the dimensions and body part masses of the human and robot are roughly equivalent; our focus is on the range of motion. The goal of the motion processing stage is to map from human motion space to a specific robot's motion space.

Several systems exist that enable effective human motion tracking. Perhaps the most familiar of these systems is marker-based optical motion capture: a user typically wears a suit with several reflective markers that are recorded by several overhead cameras, and the positions are triangulated. An example of motion mapping using an optical marker system with the Nao robot is demonstrated in [15]. One major downside to these systems is that they require large labs with expensive equipment and software. Our motion capture experiments were performed with the Microsoft Kinect sensor[1]; while not as accurate as more expensive platforms, we believe the Kinect provides a sufficient level of detail and is easily accessible to researchers without the funds or space for a dedicated motion capture lab.

Even with a system that provides low-noise tracking, a significant challenge remains in stabilizing the robot when motors are adjusted; mapping of human to robot joints, particularly in the legs, will often result in the robot falling over. Kim et al. [9] produce stable whole-body motions from motion capture by imitating a zero moment point (ZMP) trajectory of a simplified human model and dynamically adjusting the pelvis for balance. Amor et al. [1] mention the use of evolutionary algorithms to adjust the features of a mapped motion until the result is stable. Grimes et al. [5] learn a nonparametric model of forward dynamics from constrained exploration to infer actions and full-body imitation. Many other authors do not attempt to solve the balancing problem and focus entirely on mapping the upper body. After acquiring motions either manually or with other methods, the mapped robot motions need further optimization in order to achieve maximum performance and/or robustness [10].

The Covariance Matrix Adaption Evolution Strategy (CMA-ES) algorithm [6,8] is one of the most widely used algorithms for parameter optimization. The family of Natural Evolution Strategies (NES) [17] algorithms are an alternative to CMA-ES in order to perform real-valued black box function optimization. For medium size dimensions, with highly correlated parameters, the exponential NES (xNES) [3] empirically shows significant performance compatible with CMA-ES. Particle swarm optimizations [7] are simple and yet effective algorithms for optimizing a wide range of functions. We use particle swarm optimization (PSO) for both our biped walking engine [11] and as an alternative method that can be applied to our motions.

2 Human Motion Capture

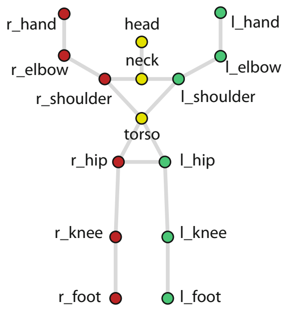
The Kinect itself does not generate motion capture (MoCap) information, but it provides color and depth images (RGB-D) that can be used to track a user's body. Microsoft's Kinect SDK[2] and an open source alternative, OpenNI [13], both implement skeletal tracking algorithms. OpenNI is a framework that provides an interface to a variety of *natural interaction* (NI) devices, such as vision or audio sensors, that record motion and sound for the purpose of human-computer interaction. Rather than directly providing implementations for all imaginable sensors, both low-level and high-level features of the OpenNI API are enabled by middleware packages. We chose to use OpenNI over the Kinect SDK as it can be used with non-Windows operating systems and provides access to existing and future NI devices, such as the Xtion Pro[3]. The PrimeSense NITE [14] middleware enables skeleton tracking for the Kinect sensor.

[1] `http://www.microsoft.com/en-us/kinectforwindows/`

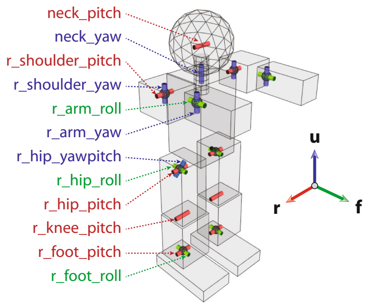
[2] `http://www.microsoft.com/en-us/kinectforwindows/develop/beta.aspx`

[3] `http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO/`

When OpenNI is configured to provide user tracking, data is presented as a skeleton model that approximates the motions of a human user. This skeleton model is defined by fifteen joints, each containing a position in sensor space; these joints are shown in Fig. 1a. To map the OpenNI user skeleton to the simulated Nao model, we first calculate two local coordinate systems for the user skeleton in terms of the vectors $\mathbf{f}$ (forward), $\mathbf{r}$ (right), and $\mathbf{u}$ (up); one coordinate system has the upper torso as the origin, and the second has the lower torso as the origin. For the upper body, which is used to calculate the arm angles: $\mathbf{f} = (\mathsf{l\_shoulder} - \mathsf{torso}) \times (\mathsf{r\_shoulder} - \mathsf{torso})$, $\mathbf{r} = \mathsf{r\_shoulder} - \mathsf{neck}$, and $\mathbf{u} = \mathbf{r} \times \mathbf{f}$. For the lower body orientation, which is used to calculate the leg angles: $\mathbf{f} = (\mathsf{r\_hip} - \mathsf{torso}) \times (\mathsf{l\_hip} - \mathsf{torso})$, $\mathbf{r} = \mathsf{r\_hip} - \mathsf{l\_hip}$, and $\mathbf{u} = \mathbf{r} \times \mathbf{f}$. Finally, the $\mathbf{f}, \mathbf{r}, \mathbf{u}$ vectors for both the upper and lower body are normalized to unit length.



(a)



(b)

Fig. 1. In (a), the OpenNI user skeleton model in the calibration pose. In (b), the physical Nao model with all joints at 0 degrees rotation. Hinge joints in the Nao are represented as cylinders through the respective rotation axis.

Once the skeleton coordinate systems are established, Euler angles are computed for the joints in the Nao model. Our mapping approach uses the vectors between skeleton joint positions to calculate the Nao joint angles; this approach can be extended to any robot model consisting of revolute joints. Inverse kinematics could be used as an alternative approach to determine joint angles, although it introduces a degree of unpredictability: the trajectory of intermediate joints in a kinematic chain are not guaranteed to follow the motion of the human. Furthermore, our goal is not to position the end effectors of the robot, but instead to ensure the relative angles of body parts are correct; a 90° bend in the human's elbow should result in a 90° bend in the robot's elbow. For these reasons, a direct calculation of the joint angles is the most appropriate. Unfortunately, the Nao's head and foot angles must be ignored, as the skeleton does not provide enough information to determine their orientations (see Fig. 1).

Each Nao arm has four joints that apply rotation in the following order: shoulder pitch (θ_s), shoulder yaw (ψ_s), arm roll (φ_a), arm yaw (ψ_a). Fig. 2 illustrates the calculation of these angles for the right arm. Using the joints of

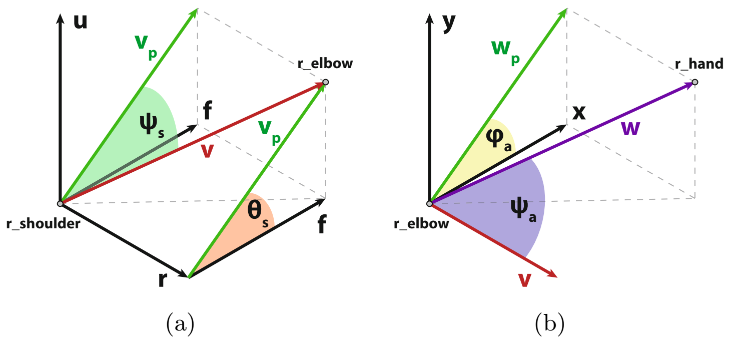


Fig. 2. Using OpenNI skeleton joint vectors calculate to Euler angles for the Nao joints θ_s = r_shoulder_pitch, ψ_s = r_shoulder_yaw, φ_a = r_arm_roll, and ψ_a = r_arm_yaw

the OpenNI skeleton, the vector from r_shoulder to r_elbow, $\mathbf{v}$, is projected onto the plane spanned by $\mathbf{f}$ and $\mathbf{u}$ (upper body) to get $\mathbf{v_p} = \mathbf{f}(\mathbf{f} \cdot \mathbf{v}) + \mathbf{u}(\mathbf{u} \cdot \mathbf{v})$. The shoulder pitch $\theta_s = \angle(\mathbf{f}, \mathbf{v_p})$, where the notation $\angle(\mathbf{a}, \mathbf{b})$ means the angle between vectors $\mathbf{a}$ and $\mathbf{b}$. The shoulder yaw $\psi_s = \angle(\mathbf{v}, \mathbf{v_p})$. After the shoulder joint angles are calculated, the arm joints are found using the same process. If the arm has no roll, it rotates (yaw) in the plane with $\mathbf{y} = \mathbf{v} \times \mathbf{v_p}$ as the normal, and $\mathbf{v}$ and $\mathbf{x} = \mathbf{y} \times \mathbf{v}$ as basis vectors; when roll is introduced, this plane is rotated around the $\mathbf{v}$ vector. The amount of roll can be found by projecting $\mathbf{w}$, the vector from r_elbow to the r_hand, onto the plane spanned by $\mathbf{x}$ and $\mathbf{y}$ to get $\mathbf{w_p} = \mathbf{x}(\mathbf{x} \cdot \mathbf{w}) + \mathbf{y}(\mathbf{y} \cdot \mathbf{w})$. The roll $\varphi_a = \angle(\mathbf{x}, \mathbf{w_p})$, and the yaw $\psi_a = \angle(\mathbf{v}, \mathbf{w})$.

For the legs, we observe that the hip, knee, and foot joints form a plane in space. The hip_yawpitch and hip_roll angles establish the orientation of this plane, and the hip_pitch and knee_pitch angles rotate the leg within this plane. The current thigh vector (knee - hip) and tibia vector (foot - knee) can be used to determine all angles for the leg. We initialize a lookup table to store the hip_yawpitch and hip_roll angles as well as the thigh vector: forward kinematics is used to iterate over possible combinations of these angles, and the normal vector of the leg plane is used as the key. To retrieve the hip_yawpitch and hip_roll angles during mapping, the current leg normal (the cross product of the thigh and tibia vectors from the skeleton) is compared with normals in the lookup table. The knee_pitch is simply the angle between the thigh and tibia vectors. Finally, the hip_pitch angle is calculated as the angle between the thigh vector stored in the lookup table and the current thigh vector.

3 Motion Optimization

The MoCap framework provides a set of traces for each motion. These motions have variable durations, and are inherently noisy. The MoCap framework captures traces only for most of the angles, but any unknown angles (head and foot) default to zero. The direct replay of the captured motions (synchronized to

50 Hz) causes the agent to fall, since the mapping of the human motion to the robot does not consider physics or the masses and capabilities of the robot. The sequences of joint angles provided by the direct mapping have to be adjusted to obtain a stable motion. This is the main problem that we are addressing in this section. Given some motions as input, we extended our framework to (1) construct models of the motions; (2) initialize model parameters using maximum likelihood and least squares; (3) optimize prior parameters to follow the original motions; and (4) find joint angles that can be replayed by the robot without falling.

3.1 Models and Initialization

A motion consists of a sequence of target angles for each joint. These sequences consist of 50 angles per second for the joint control. Instead of adjusting these angles directly, we create models for the movements of the joints. By approximating the given joint angles with functions, only a relatively small set of parameters has to be optimized to achieve the correct motion on the robot.

We use linear combination of fixed nonlinear functions of the input variable to build the motion model. The input variable, $\mathbf{x} = \{x_1, \ldots, x_M\}$, is the the number of frames in a motion. We use a model of the form $y(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=0}^{N-1} \theta_j \phi_j(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}} \mathit{\Phi}(\mathbf{x})$, where $\phi_j(\mathbf{x})$ are the basis functions with $\phi_0(\mathbf{x}) = 1$, $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{N-1})^{\mathrm{T}}$, and $\mathit{\Phi} = (\phi_0, \ldots, \phi_{N-1})^{\mathrm{T}}$. There are N total number of parameters in the model. With the choice of suitable basis functions, we model arbitrary nonlinearities in the input traces. Basis functions take many forms, and we use polynomial basis functions of the form $\phi_j(x) = x^j$, and sigmoidal basis functions of the form $\phi_j(x) = \sigma(\frac{x-\mu_j}{s})$, where $\sigma(a)$ is the logistic sigmoidal function defined by $\sigma(a) = \frac{1}{1+\exp(-a)}$, μ_j fixes the location of the basis functions in the input space, and s represents the spacial scale. Polynomial basis functions are global functions of the input, which cause changes in one region to affect all the other regions. On the other hand, sigmoidal basis functions are local, and a small change to input only affect some of the nearby basis functions. The application of global and local basis function can have a significant influence on the optimization. The target variable, $\mathbf{t} = (t_1, \ldots, t_M)^{\mathrm{T}}$, of the motion is the desired angle of a given trace. Each motion contains K traces ($K = 22$ in our experiments), and each trace of the motion is fitted using the linear basis function model. We minimize the objective function $\sum_{i=1}^{M}(t_i - \boldsymbol{\theta}^{\mathrm{T}} \mathit{\Phi}(\mathbf{x}))^2$ using ES/PSO algorithms to find the maximum likelihood parameters.

3.2 Model Optimization

The evaluation of the models with the initial parameters provides approximately close enough traces to the original motions. A replay of a motion with the respective model initially fails to capture the desired outcome of the original motion. The joints are moved according to the input motion, however depending on specific robot model (e.g., the masses of body parts) it is necessary to change the motion slightly.

The initial optimization of the model parameters is used as a seed for the optimization of the motion for stability on the robot. This task is an optimization problem with two conflicting objectives. Following exactly the joint angles provided by the MoCap does not guarantee that the outcome is the correct motion. For instance, for a kick motion the robot could fall back and kick into the air, which might follow exactly the given joint angles. This can be avoided by including the captured torso orientation of the human for every time step in the motion capture data. The torso orientation of the simulated robot is provided by the simulator as ground truth. The difference between the captured angles and the robots torso orientation is one component of the fitness function. At the same time the changes in the angles have to be small to make sure that the final result is close to the captured motion, e.g., a kick motion should not be stabilized by removing the actual kick from the motion. The differences between the angles provided by the MoCap and the joint angles of the robot are the second component of the fitness function.

We use ES/PSO algorithms again to optimize the model parameters until the desired motion is learned. In this phase, we optimize $N \times L$, where $L < K$, parameters directly. We use the sum of the torso errors and the joint errors over all frames of the motion as the fitness to perform real valued black box function optimization. We have decided to optimize only the traces of the agent's legs. There are twelve such traces for each motion, and we directly optimize $N \times 12$ parameters. e.g., if we commit to a polynomial model with eight parameters, we optimize 96 parameters.

4 Experimental Setup

The experiments in section 5 have been conducted using SimSpark (based on Spark[12]), the simulator of the RoboCup 3D Soccer Simulation League. The simulated robot is a humanoid robot that is similar to the Nao [4]. The robot is equipped with 21 degrees of freedom, and it receives sensor information every cycle (50 Hz) from the server.

We use four different motions in the experiments in which the robot (1) lifts the right leg for a few seconds (motion `leg`); (2) performs a simple kick motion (motion `kick`); (3) leans forward and balances on one leg while stretching the other leg back (motion `balance`); and (4) leans the torso to the side (motion `side`). The joint motions are modeled using two different functions: polynomials and linear weighted sigmoidal basis functions. These models are initialized by minimizing the least squared error to the input angles. Using the initial parameters as a seed, the twelve leg joints are optimized by CMA-ES, xNES, and PSO using the fitness function based on the joint and the torso error explained in 3.2.

In [16], twelve parameters were learned with a population size of 30; since we optimize up to 96 parameters, we use a higher population size of 50 for all optimization methods. CMA-ES and xNES start with the parameters from the initialization of the models as the mean and a standard deviation of 0.06, which produces a suitable amount of exploration in the beginning of the optimization.

The samples for PSO are initialized using the same mean and standard deviation. Our PSO implementation uses the parameters proposed in [2].

For each evaluation of parameters, the robot is initialized to the pose in the first frame of the motion. While the motion is executed using the current parameters, all torso and joint errors are added; these errors are used as the fitness of the tested parameters. For each parameter set, the motion is executed only once since the environment is simulated and we can expect less noise than with a physical robot.

Learning the transition between different motions is beyond the scope of this paper. The evaluation of a parameter set starts with a short phase during which the robot moves all joints to the angles at frame 0 of the motion. The robot is also moved to an initial torso position using a trainer interface of the simulator. This way, the motion is always started from the same initial situation. At the end of the motion, the robot keeps the joints at the angles from the last frame for half a second. During this time, the torso and joint errors for the evaluation are still accumulated; this prevents learning of motions that are unstable in the end and would make the robot fall immediately after the motion is done. Learning transitions between motions is planned as future work.

5 Experiments and Results

For the model of the first experiment, we used polynomial basis functions with five parameters. Fig. 3a shows the joint angles of the balance motion, and Fig. 3b shows the traces of two of the joints. The optimized motion slightly adjusts the initial values of the parameters to obtain stable motion, and the combination of all joint traces together needs to be stable. However, the experiments show that often only very small changes in the joint motion provide a stable and complete motion. In Fig. 3b the polynomial seems to be able to sufficiently approximate the motion of the joint.

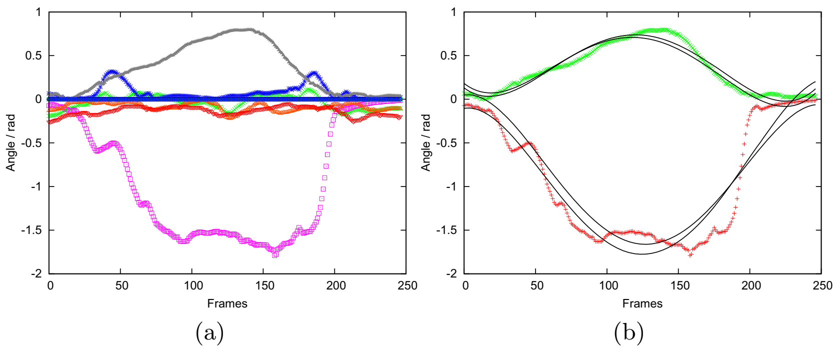


Fig. 3. (a) The leg joint angles of the balance motion. (b) The motions of two leg joints (l_knee_pitch and r_hip_pitch) during the balance motion with the corresponding models using the initial parameters and the adjusted models of the stabilized motion.

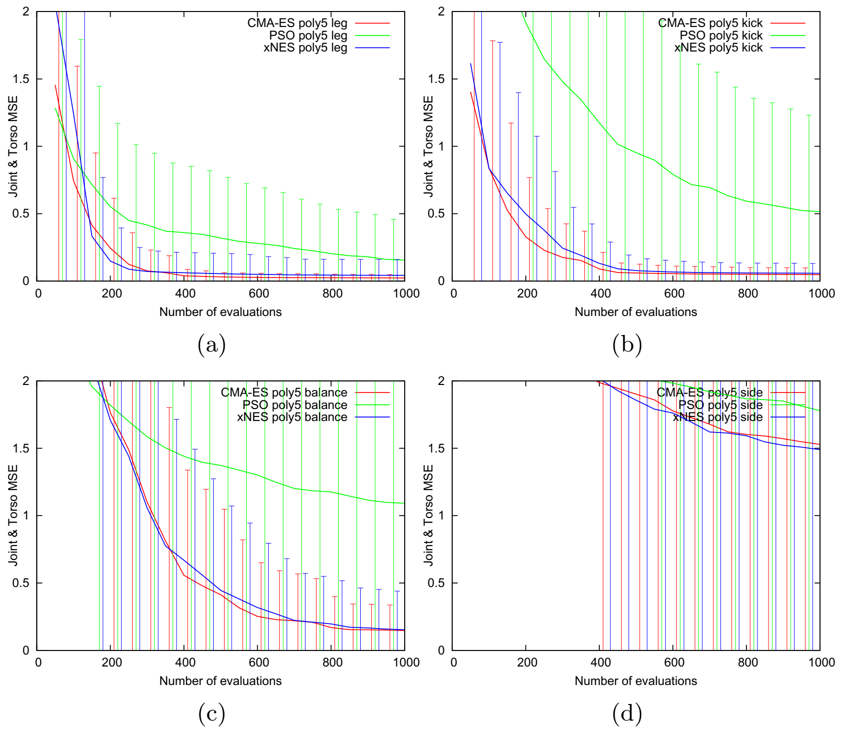


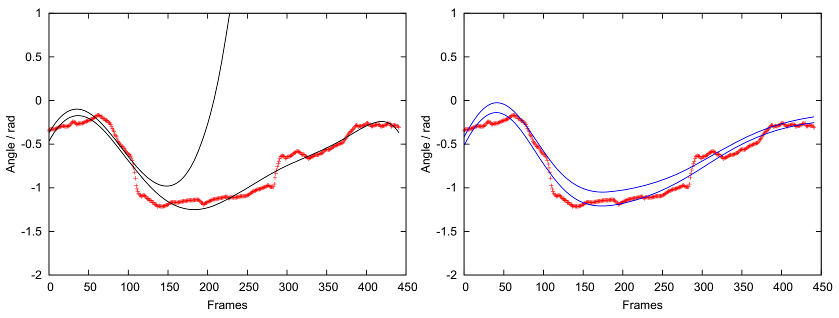
Fig. 4. The learning curves for different motions using polynomials of degree 4. The motions for the twelve leg joints were optimized using CMA-ES, PSO and xNES. The error is the minimum total error (joints MSE + torso angle MSE) averaged over 30 runs. The error bars represent the standard deviation over these 30 runs.

We used the same polynomial approximation to learn four different motions, and Fig. 4 shows the average learning curves. In these experiments, both CMA-ES and xNES quickly learn solutions, with CMA-ES finding a solution with a slightly smaller variance. Both algorithms perform better than PSO. It is possible that the results of PSO could be improved by tuning some internal parameters; CMA-ES and xNES do not require this.

There is a swift learning curve for leg motion. For the kick motion, PSO yields a very high variance, which indicates that the found motion is often unstable. For the balance motion in Fig. 4c, CMA-ES and xNES find good solutions, but the learning time increases. While the polynomials work for these three motions, the algorithms could not find a stable motion for the side balance motion in Fig. 4d. In fact, the results for the other motions are also often unsatisfactory. Polynomials cause these motions to be very smooth. Although the errors are often small and the motion is stable, there is a noticeable lack of detail, and the high-degree polynomials introduce numerical instabilities. In most motions, polynomials cause some joints to move unexpectedly towards the end.

Table 1. The errors after learning for 5 hours simulated time using polynomials with 5 parameters as models (average over 30 runs)

Motion	Optimization	evals	min.err.	avg.err.	stddev	torso err.	joint err.	success
leg	CMA-ES	2989	0.009	0.016	0.004	0.006	0.009	83%
leg	PSO	2987	0.006	0.058	0.154	0.049	0.009	70%
leg	xNES	2990	0.013	0.017	0.003	0.007	0.010	86%
kick	CMA-ES	4926	0.018	0.027	0.005	0.011	0.016	60%
kick	PSO	4924	0.015	0.135	0.210	0.106	0.029	46%
kick	xNES	4931	0.023	0.037	0.047	0.020	0.017	60%
balance	CMA-ES	2970	0.051	0.096	0.076	0.046	0.050	60%
balance	PSO	2971	0.072	0.807	0.403	0.720	0.088	13%
balance	xNES	2971	0.047	0.089	0.153	0.050	0.039	60%
side	CMA-ES	1816	0.432	1.351	0.466	1.203	0.149	0%
side	PSO	1816	0.474	1.634	0.370	1.505	0.129	0%
side	xNES	1816	0.401	1.229	0.490	1.092	0.137	0%



(a) (b)

Fig. 5. An example joint motion (l_knee_pitch) of the side balance and the initial and learned model using CMA-ES and (a) a polynomial with 8 parameters or (b) sigmoidal basis functions with a sum of 8 parameters

Table 1 shows the error values of the experiments after five hours of simulated time. The success rates are created by manually evaluating how many result motions are stable and close enough to the original motion. A longer learning time improves the success rates. However, another reason for lower success rates, despite small average errors, is the noise in the fitness values. There is a chance that an unstable motion gets a small error once and never works again. Averaging the fitness over several runs could lower this noise and improve the learning, but each evaluation would need much more time.

As an attempt to improve the learning, we ran the optimization of the side balance motion again using polynomials, but increased the number of parameters to eight. It still did not find a solution. Increasing the number of degrees only creates more instabilities. Fig. 5a shows that a reason for the unsatisfactory performance is the global influence of parameters on the function. Changing the first part of the motion can create completely wrong angles for the remaining motion.

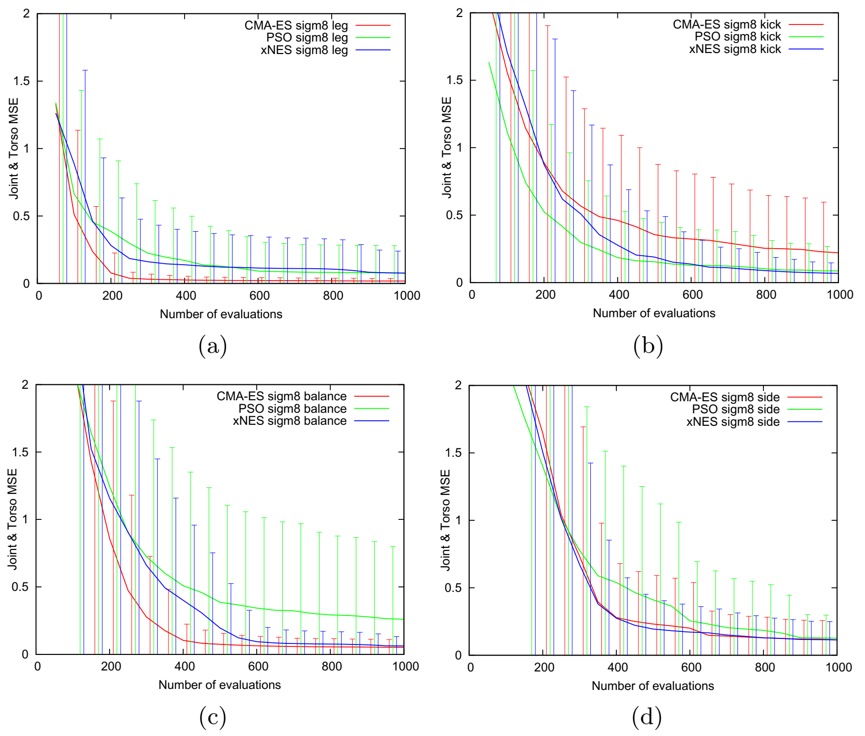


Fig. 6. Using sigmoidal basis functions instead of polynomials improves the results of the learning. The learning curves of the same experiments as in Fig. 4 are shown (averages over 30 runs).

Table 2. Results of the optimization using sigmoidal basis functions

Motion	Optimization	evals	min.err.	avg.err.	stddev	torso err.	joint err.	success
leg	CMA-ES	1781	0.011	0.016	0.004	0.007	0.008	93%
leg	PSO	1782	0.005	0.051	0.149	0.044	0.007	76%
leg	xNES	1782	0.012	0.042	0.086	0.027	0.014	70%
kick	CMA-ES	2935	0.029	0.150	0.206	0.124	0.027	53%
kick	PSO	2935	0.012	0.038	0.068	0.029	0.009	43%
kick	xNES	2940	0.037	0.047	0.007	0.025	0.023	40%
balance	CMA-ES	2970	0.029	0.042	0.016	0.018	0.024	93%
balance	PSO	2971	0.027	0.119	0.256	0.092	0.027	73%
balance	xNES	2971	0.030	0.041	0.009	0.020	0.020	76%
side	CMA-ES	1816	0.041	0.099	0.060	0.073	0.026	70%
side	PSO	1816	0.024	0.093	0.085	0.064	0.029	60%
side	xNES	1817	0.038	0.102	0.062	0.078	0.024	40%

Since the side motion could not be learned at all and the other motions suffered from the high generalization and numerical instabilities of the polynomials, we ran the same experiments again with the linear weighted sigmoidal basis functions as the model. Fig. 5b shows that the local basis functions improve the optimization. All four motions can be adjusted to be stable on the robot using this model (Fig. 6). Although the number of parameters that are optimized has been increased from 60 to 96 (twelve joints, five or eight parameters per model), the optimization needs less time to find solutions for the balance motion and also works for the side balance. The average joint and torso errors are listed in Table 2. For the balance motion the improved models yield significantly smaller joint angle errors compared to the polynomials.

6 Conclusions and Future Work

Our hypothesis that robust and human-like motions for a biped soccer robot can be quickly generated using inexpensive motion capture techniques with optimization is verified. The Kinect sensor provides enough information to map complex and initially unstable poses, such as the balancing motions. Applying modern optimization algorithms to the mapped motions will lead to robust and stable motions on the robot; that said, we do not make the claim that our approach will hold for *all* possible motions. Nevertheless, our results are very promising, and it shows that we have found a process that can be used to produce a number of motions needed for humanoid soccer robots. We ran our parameter optimization on a desktop CPU with four cores and 2.27 GHz. After running experiments with four motions, three algorithms, two model functions with different number of parameters and 30 runs each, we can say that the motions were mostly stable after 30 min (in average). Making use of parallel processing on a cluster of computers can bring the entire processing time down to 10-30 min per motion, including the motion capture.

We have observed that CMA-ES and xNES are similar in performance. Also, the PSO algorithm shows partly smaller errors with a larger variance and it takes longer to learn. However, a smaller error does not necessarily mean a better motion. Sometimes parameters that created a small error once produce motions that result in a falling robot. The influence of running a motion multiple times for each evaluation and averaging the error needs to be evaluated in the future. Nevertheless, the optimization using models with sigmoidal basis functions yields good results for all three algorithms. Optimizing only the leg motions is sufficient for the used robot model. Further experiments have shown that including the arms yields a higher error and variance without finding better solutions.

On the motion capture end, we would like to compare our motions mapped using the Kinect with motions captured from a Vicon optical marker system. Some motions are particularly tricky to perform on the Kinect, such as standing after a fall or kicking, as the skeletal tracking algorithms can be confused when the user rotates or has an body part that is at least partially occluded. Our motion capture pipeline is written in a modular fashion to accommodate changes to the input sensor, the mapping mechanism, and the robot model; we believe this framework can be used to experiment with at least a few other robot models, both simulated and physical, and provide a convenient platform for quickly creating motions on heterogeneous humanoid robots.

References

1. Amor, H., Berger, E., Vogt, D., Jung, B.: Towards Responsive Humanoids: Learning Interaction Models for Humanoid Robots. In: International Conference on Machine Learning, pp. 1–4 (2011)
2. Carlisle, A., Dozier, G.: An off-the-shelf PSO. In: Proceedings of the Workshop on Particle Swarm Optimization, pp. 1–6. Purdue School of Engineering and Technology, IUPUI, Indianapolis (2001)
3. Glasmachers, T., Schaul, T., Schmidhuber, J.: A natural evolution strategy for multi-objective optimization. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 627–636. Springer, Heidelberg (2010)
4. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: The nao humanoid: a combination of performance and affordability. CoRR abs/0807.3223 (2008)
5. Grimes, D.B., Rashid, D.R., Rao, R.P.N.: Learning nonparametric models for probabilistic imitation. In: Advances in Neural Information Processing Systems (NIPS). MIT Press (2006)
6. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks 1995, vol. 4, pp. 1942–1948. IEEE (1995)
8. Kern, S., Müller, S., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms–a comparative review. Natural Computing 3(1), 77–112 (2004)
9. Kim, S., Kim, C., You, B., Oh, S.: Stable whole-body motion generation for humanoid robots to imitate human motions. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, pp. 2518–2524. IEEE (2009)
10. MacAlpine, P., Urieli, D., Barrett, S., Kalyanakrishnan, S., Barrera, F., Lopez-Mobilia, A., Ştiurcă, N., Vu, V., Stone, P.: UT Austin Villa 2011: A champion agent in the RoboCup 3D soccer simulation competition. In: Proc. of 11th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012) (June 2012)
11. Niehaus, C., Röfer, T., Laue, T.: Gait Optimization on a Humanoid Robot using Particle Swarm Optimization (2007)
12. Obst, O., Rollmann, M., Rollmann, M.: Spark - a generic simulator for physical multi-agent simulations. In: Computer Systems Science and Engineering (2004)
13. OpenNI organization: OpenNI User Guide (November 2010), `http://www.openni.org/documentation` (last viewed February 26, 2012)
14. PrimeSense Inc.: Prime Sensor NITE 1.3 Algorithms notes (2010), `http://www.primesense.com` (last viewed February 26, 2012)
15. Setapen, A., Quinlan, M., Stone, P.: Marionet: Motion acquisition for robots through iterative online evaluative training (extended abstract). In: The Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS). International Foundation for Autonomous Agents and Multiagent Systems (May 2010)
16. Urieli, D., MacAlpine, P., Kalyanakrishnan, S., Bentor, Y., Stone, P.: On optimizing interdependent skills: A case study in simulated 3d humanoid robot soccer. In: Tumer, K., Yolum, P., Sonenberg, L., Stone, P. (eds.) Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011), vol. 2, pp. 769–776. IFAAMAS (May 2011)
17. Wierstra, D., Schaul, T., Peters, J., Schmidhuber, J.: Natural evolution strategies. In: Proceedings of the Congress on Evolutionary Computation (CEC 2008), Hongkong. IEEE Press (2008)

A CASE Tool for Robot Behavior Development

Angeliki Topalidou-Kyniazopoulou[1], Nikolaos I. Spanoudakis[2], and Michail G. Lagoudakis[1]

[1] Department of ECE, Technical University of Crete, 73100, Chania, Greece
atop87@gmail.com, lagoudakis@intelligence.tuc.gr

[2] Department of Sciences, Technical University of Crete, 73100, Chania, Greece
nikos@science.tuc.gr

Abstract. The development of high-level behavior for autonomous robots is a time-consuming task even for experts. This paper presents a Computer-Aided Software Engineering (CASE) tool, named Kouretes Statechart Editor (KSE), which enables the developer to easily specify a desired robot behavior as a statechart model utilizing a variety of base robot functionalities (vision, localization, locomotion, motion skills, communication). A statechart is a compact platform-independent formal model used widely in software engineering for designing software systems. KSE adopts the Agent Systems Engineering Methodology (ASEME) model-driven approach. Thus, KSE guides the developer through a series of design steps within a graphical environment that leads to automatic source code generation. We use KSE for developing the behavior of the Nao humanoid robots of our team Kouretes competing in the Standard Platform League of the RoboCup competition.

1 Introduction

Computer-Aided Software Engineering (CASE) tools improve productivity and quality in software development [1]. However, they are not widely used for robot behavior development, even in domains, such as RoboCup, where robot behavior needs to be frequently modified. It is quite common for a RoboCup team to find itself in a place where the code realizing the behavior of its robots needs to be urgently modified, for example during half-time because of some unexpected opponent strategy. The time constraints and the programmers' stress in such situations consist a recipe for failure. CASE tools can be really helpful in this context, as they offer ways to make behavior development and modification quicker and less error-prone.

Recent advances in Agent Oriented Software Engineering (AOSE), Model-Driven Engineering (MDE), and Domain Specific Languages (DSLs) allowed us to define a novel model-driven process for developing collaborative robot behaviors [2]. This process, however, lacked the assistance of a CASE tool that would allow the graphical editing of the behavior models. The work presented in this paper aims to fill this gap by proposing the Kouretes Statechart Editor (KSE) CASE tool, which enables the developer to easily specify a desired robot behavior as a statechart model utilizing a variety of base robot functionalities (vision, localization, locomotion, motion skills, communication). A statechart [3] is a compact platform-independent formal model used widely in software engineering for designing software systems. KSE adopts the Agent Systems Engineering Methodology (ASEME) model-driven approach [4] and assists the developer from the analysis phase to the design and code generation phases. More specifically, KSE supports (a) the automatic generation of the initial abstract statechart model using compact liveness formulas, (b) the graphical editing of the statechart model and the addition of the required transition expressions, and (c) the automatic source code generation for compilation and execution on the robot. KSE has been developed using the Eclipse Modeling Project[1] technologies and has been integrated with our Monas software architecture [5] and our Narukom communication framework [6], which provide the base functionalities. KSE is used for developing the behavior of the Aldebaran Nao humanoid robots of our team Kouretes competing in the RoboCup Standard Platform League (SPL).

In the rest of the paper, after examining the background technologies in Section 2, we present our ASEME-based robot behavior development method in Section 3 and the main features of KSE, including design and implementation choices, in Section 4. Subsequently, we present the results of a first empirical evaluation in Section 5. Finally, we discuss our findings and related work in Section 6 before concluding with Section 7.

2 Background

ASEME [4] supports a modular agent design approach and introduces the concepts of intra- and inter- agent control. The former defines the agent's behavior by coordinating the different modules that implement its own capabilities, while the latter defines the protocols that govern the coordination of the society of the agents. ASEME applies a Model-Driven Engineering (MDE) approach to multi-agent systems development, so that the models of a previous development phase can be transformed to models of the next phase. The transition from one phase to another is assisted by automatic model transformation leading from requirements to computer programs. The ASEME platform-independent model, which is the output of the design phase, is a statechart, and is referred to as the Intra-Agent Control (IAC) model. ASEME uses the models of the Agent Modeling Language (AMOLA) [7]. The AMOLA metamodels have been formally defined using the Eclipse Modeling Framework (EMF) of the Eclipse Modeling Project. Eclipse technology has been employed for developing model transformations and graphical editing tools for both models and processes.

Our communication framework, Narukom [6], is based on the publish/subscribe messaging pattern [8] and supports multiple ways of communication, including point-to-point and multicast connections. The information that needs to be communicated between nodes (agents or activities) is formed as messages, tagged with appropriate topics, and relayed through a message queue for delivery. Three types of messages are supported: (i) *state*, which remain in the blackboard

[1] The Eclipse Modeling Project provides a unified set of modeling frameworks, tooling, and standards implementations: `www.eclipse.org/modeling`

until they are replaced by a newer message of the same type, (ii) *signal*, which are consumed at the first read, and (iii) *data*, which are time-stamped to indicate the precise time the values they carry were acquired. We used Google Protocol Buffers[2] to facilitate the serialization of data and the structural definition of the messages. Additionally, the blackboard paradigm [9] is utilized to provide efficient access to shared information stored locally at each node and is extended to support history queries and a mechanism that controls the information updates.

Our software architecture, Monas [5], provides an abstraction layer from the Nao robot and allows the synthesis of complex robot software as XML-specified Monas modules and/or statechart modules. Monas modules focus on specific functionalities (vision, motion, etc.) and each one of them is executed independently at any desired frequency completing a series of activities at each execution. Statechart modules are executed using a generic multi-threaded statechart engine [5], which is built on top of existing open-source projects, provides the required concurrency, and meets the real-time requirements of the activities on each robot. The base functionalities utilized by a statechart can be implemented as Monas modules and include the following: *Sensors*, for collecting and filtering measurements from the robot sensors; *RobotController*, for handling external signals on the game state; *MotionController*, for managing and executing robot locomotion and special actions; *Vision*, for obtaining visual object observations; *Localization*, for estimating the position of the robot and the ball in the field; and *HeadHandler*, for managing the movements of the robot head (camera).

3 ASEME-based Behavior Development

ASEME suggests a strict hierarchical decomposition of the desired robot behavior into smaller activities until a level of provided base activities is met. Each design step is supported by a formal model. These models are automatically transformed when moving from one step of the design process to the next. Briefly, the process begins with the specification of a set of liveness formulas (analysis phase) which are converted to an initial statechart model; the statechart is subsequently enriched (design phase) and is converted to source code.

Liveness formulas describe and connect the activities included in the desired behavior in a formal way, similar to regular expressions. Each formula is a rule decomposing one activity (shown on the left side) into a number of interconnected smaller activities (shown on the right side). Activities on the right side of a formula are connected using the Gaia operators [10]. Specifically, `A.B` means that activity `B` is executed after activity `A`, `A*` means that `A` is executed zero or more times, `A+` means that `A` is executed one or more times, `A~` means that `A` is executed indefinitely (it resumes as soon as it finishes), `A|B` means that either `A` or `B` is executed exclusively, `A||B` means that `A` and `B` are executed concurrently, and `[A]` means that `A` is optional. Using liveness formulas, the developer can hierarchically decompose the desired behavior into specific activities until

[2] Google's language- and platform- independent, extensible mechanism for serializing structured data: `http://code.google.com/apis/protocolbuffers`

the existing base activities are reached, however without specifying the precise conditions under which individual activities are chosen.

The statecharts [3] are formal models that describe complex processes and control structures using directed graphs with nodes (states) and edges (transitions). Six types of nodes or states are allowed: *start*–states, indicating the entry of execution in a complex state, *end*–states, indicating the exit of execution from a complex state, *or*–states, indicating complex states with mutually exclusive sub-states (only one sub-state is executed at each time), *and*–states, indicating complex states with sub-states of type *or* which are executed concurrently, *basic*–states, indicating the execution of base activities, and *condition*–states, providing the ability to make conditional transitions. The state at the highest level (the only one without a parent) is called the *root*. Each transition from one state (source) to another (target) is characterized by an expression whose syntax follows the pattern e[c]/a, where e is the event triggering the transition, c is the condition that needs to be satisfied for the transition to take place when e occurs, and a is an action executed when the transition is taken. All the elements of an expression are optional. If the expression lacks an event, the condition is evaluated when the source state finishes execution.

The liveness2IAC tool [11] is used to transform the liveness formulas to statecharts and the IAC2Monas tool [5] is used to transform the statechart automatically to C++ source code adhering to the Monas architecture. With the use of the latter, the platform-independent model (statechart) is transformed to a platform-specific model (source code), which is subsequently cross-compiled to produce the executable for the robot.

4 The Kouretes Statechart Editor CASE Tool

KSE is a CASE tool designed to support all steps of ASEME-based behavior development through an intuitive graphical interface. In particular, liveness formulas are given in plain text and are automatically converted to an initial statechart model, where the designer can graphically add the appropriate transition expressions. The syntax of transition expressions is formally specified by an EBNF grammar [12]. Each statechart can be associated with a source code repository containing the base activities; in our case, a repository of Monas activities. KSE also allows the creation of statecharts from scratch (without liveness formulas) and graphical editing and modification of any existing statechart. To ensure that the designer will not produce an invalid statechart with respect to Harel's statechart language [3] and the EBNF grammar, KSE offers a validation procedure which identifies mistakes in the statechart and warns the user. The final statechart is automatically converted to source code which is integrated with the associated source code repository and is cross-compiled for execution.

4.1 KSE Example

We provide a simple example to demonstrate KSE and the behavior development process. Consider a very simple behavior, whereby a robot listening to

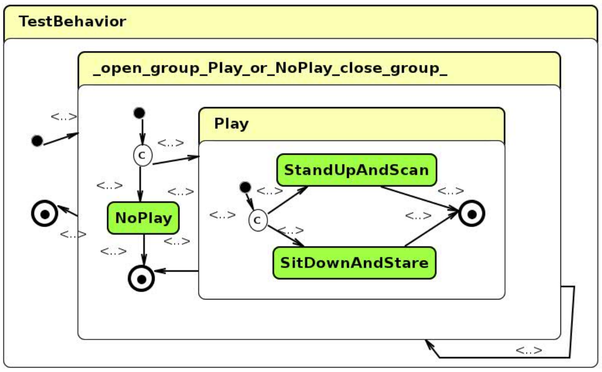


Fig. 1. KSE example: the generated statechart model from the liveness formula

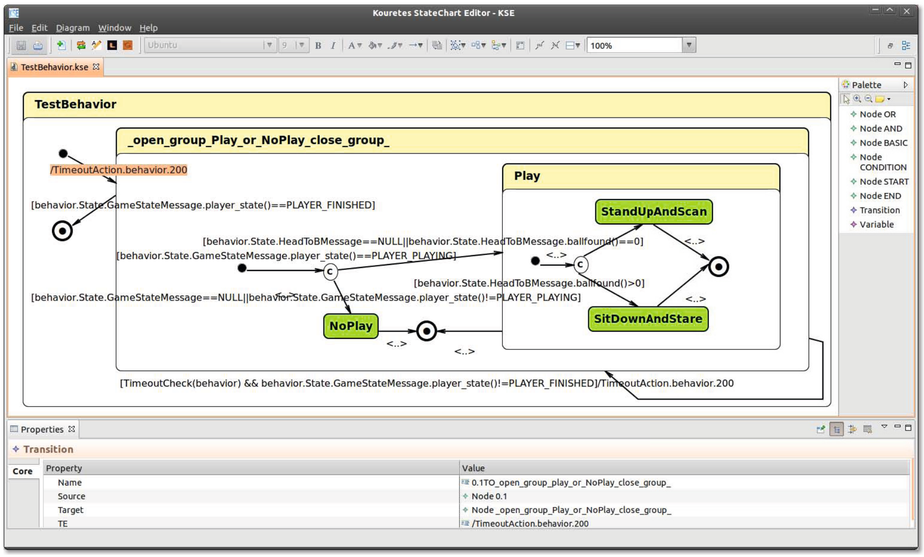
SPL's game controller executes the following actions, whenever it enters the `PLAYER_PLAYING` state: *sit down when you see the ball and track it, stand up and scan for the ball when you lose it*. In any other game state, it does nothing.

The first step in creating a behavior with KSE is to describe the behavior with liveness formulas. The two liveness formulas for this simple behavior are:

```
TestBehavior = (Play | NoPlay)+
Play = SitDownAndStare | StandUpAndScan
```

The first formula indicates that our behavior (`TestBehavior`) will choose one or more times between `Play` and `NoPlay` exclusively. `NoPlay` is a base activity, which handles game states different than `PLAYER_PLAYING`, and is not analyzed further. `Play` is refined in the second formula, which simply states that `Play` will have to choose one of the two base activities, `SitDownAndStare` or `StandUpAndScan`, exclusively. `SitDownAndStare` commands the robot to sit down and stare at the (visible) ball, whereas `StandUpAndScan` commands the robot to stand up and scan for a ball. Note that this decomposition specifies *what* activities are included in the desired behavior, but gives no information on *when* execution will switch from one activity to another.

As soon as the formula is provided to KSE, the initial statechart model is generated and the user has to associate it to a source code repository, which provides the base functionalities and in which the code of the new statechart is going to be integrated. At this point, the user can initialize the graphical representation (Figure 1) of the automatically created statechart model in order to edit the transition expressions or the activities of the *basic*-states. Note that each activity in the liveness formula has become a node/state in the statechart. The yellow-color-labeled rounded rectangles indicate *or*-states, the green-colored rounded rectangles *basic*-states, and the blue-color-labeled rounded rectangles *and*-states (none in this example). A node with a circled `c` is a *condition*-state, whereas solid black nodes correspond to *start*-states and circled black nodes to *end*-states. The model hierarchy is preserved in the graphical node enclosures.


Kouretes StateChart Editor - KSE
File Edit Diagram Window Help
TestBehavior.kse
TestBehavior
_open_group_Play_or_NoPlay_close_group_
/TimeoutAction.behavior.200
[behavior.State.GameStateMessage.player_state()==PLAYER_FINISHED]
Play
StandUpAndScan
[behavior.State.HeadToBMessage==NULL||behavior.State.HeadToBMessage.ballfound()==0]
[behavior.State.GameStateMessage.player_state()==PLAYER_PLAYING]
<..>
[behavior.State.HeadToBMessage.ballfound()>0]
[behavior.State.GameStateMessage==NULL||behavior.State.GameStateMessage.player_state()!=PLAYER_PLAYING]
<..>
NoPlay
SitDownAndStare
<..>
<..>
[TimeoutCheck(behavior) && behavior.State.GameStateMessage.player_state()!=PLAYER_FINISHED]/TimeoutAction.behavior.200
Palette
Node OR
Node AND
Node BASIC
Node CONDITION
Node START
Node END
Transition
Variable
Properties
Transition
Core

Property	Value
Name	0.1TO_open_group_play_or_NoPlay_close_group_
Source	Node 0.1
Target	Node _open_group_Play_or_NoPlay_close_group_
TE	/TimeoutAction.behavior.200

Fig. 2. KSE example: the complete statechart model with all the transition expressions

An empty transition expression (inidicated by `<..>`) implies that no event is required for triggering, the condition is evaluated to `true`, and no action is executed. Such a transition is used to indicate default execution paths, when all other non-empty transition conditions evaluate to `false`. To ensure proper execution of the statechart, the user must define the appropriate transition expressions. In our example, we have to provide six transition expressions: when to continue with `(Play | NoPlay)`, when to leave `(Play | NoPlay)`, when to choose `Play`, when to choose `NoPlay`, when to choose `SitDownAndStare`, and when to choose `StandUpAndScan`. These conditions take into account information delivered by incoming messages from existing Monas modules indicating the game state and whether the ball is visible or not. The complete statechart with all transition expressions is shown in Figure 2. It is worth noting the loop transition on `(Play | NoPlay)`, which executes a timeout action (200 msec) so that the transition to the target state can only take place at a certain frequency enforced by the `TimeoutCheck` function in the condition. The same action is taken when this state is entered the first time (transition from its *start*-state).

At this point, the user can validate the statechart model and generate the source code. Classes are generated for the model, for the activity of each *basic*-state, and for each transition. Additionally, if the activity of a *basic*-state is not already provided, a class template will be generated in which the user must define the corresponding functionality using conventional C++ code. The user can edit the source code of the activity corresponding to a *basic*-state directly within KSE. In this example, the user just needs to define the activities of the three *basic*-states: `NoPlay`, `SitDownAndStare`, and `StandUpAndScan`. A sample of the auto-generated code is shown in Figures 3 and 4. In our example, the size

```
#include "TestBehavior.h"
#include "transitionHeaders.h"
using namespace statechart_engine;
namespace {StatechartRegistrar<TestBehavior >::Type temp("TestBehavior");}
TestBehavior::TestBehavior(Narukom* com) {
  _statechart = new Statechart ( "Node_TestBehavior", com );
  Statechart* Node_0 = _statechart;
  _states.push_back( Node_0 );

  StartState* Node_0_1 = new StartState ( "Node_0_1", Node_0 ); //Name:0.1
  _states.push_back( Node_0_1 );
...
```

Fig. 3. KSE example: an extract of the generated code for the `TestBehavior` statechart

```
#include "architecture/statechartEngine/ICondition.h"
#include "messages/AllMessagesHeader.h"
#include "tools/BehaviorConst.h"
class TrCond_TestBehavior0_20_2:public statechart_engine::ICondition {
public:
  void UserInit(){_blk->updateSubscription("behavior",msgentry::SUBSCRIBE_ON_TOPIC);}
  bool Eval() {
    boost::shared_ptr<const GameStateMessage> var_621149599 = _blk->readState<
        GameStateMessage> ("behavior");
    boost::shared_ptr<const TimeoutMsg> msg= blk->readState<TimeoutMsg>("behavior");
    return ( (msg.get()!=0 && msg->wakeup()!="" && boost::posix_time::from_iso_string
        (msg->wakeup())<boost::posix_time::microsec_clock::local_time()) && (
        var_621149599.get()!=0 && var_621149599->player_state()!=PLAYER_FINISHED) );
  }
};
#include "architecture/statechartEngine/IAction.h"
#include "architecture/statechartEngine/TimoutAciton.h"
class TrAction_TestBehavior0_20_2:public statechart_engine::TimeoutAction {
  public:TrAction_TestBehavior0_20_2():statechart_engine::TimeoutAction("behavior"
      ,200){;}
};
```

Fig. 4. KSE example: the generated code for the loop transition on `(Play | NoPlay)`

of the total auto-generated code is 35.5 KB and with the user-defined activities for the three *basic*-states it increases to 50.9 KB. Therefore, about 70% of the code for this simple behavior has been automatically generated.

4.2 KSE Design and Implementation

To design and implement KSE, we chose the eclipse platform and the technologies offered by the Eclipse Modeling Project, in particular the Eclipse Modeling Framework (EMF), the Graphical Modeling Framework (GMF), and the Xpand language. This choice was apparent, mainly because of the fact that the ASEME tools already used these technologies, but also because other modern CASE tools, such as Yakindu[3], are also based on them.

For creating the liveness formulas, the designer uses a simple text editor. The liveness2IAC transformation tool transforms the liveness formulas to a statechart instance based on the transformation templates for Gaia operators [11]. This text-to-model transformation uses the formal definition of liveness formulas and the `Statechart` metamodel defined in EMF ecore format, shown graphically in Figure 5. According to the `Statechart` metamodel, a *Model* consists of *Node*, *Transition*, and *Variable* instances. A node represents a state in the statechart and has a *name* (providing a description of the state), a *label* (indicating its unique position in the hierarchy), an *activity* (hosting a path to the source code implementing the functionality executed when the state is active), and a *type*

[3] A free toolkit for model-driven development of embedded systems: `www.yakindu.org`

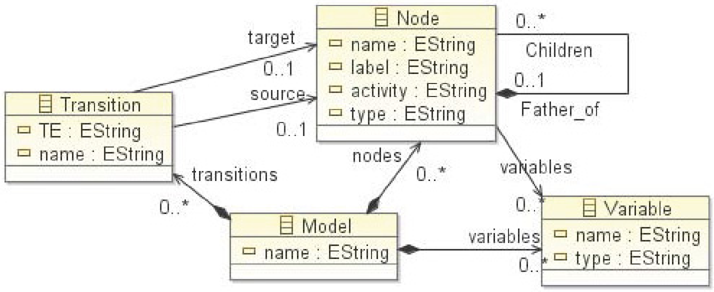


Fig. 5. The `Statechart` metamodel in EMF ecore format

(indicating the type of the state: *or*-state, *and*-state, etc.). Nodes aggregate their *Children* (sub-nodes) and reference their *Father* (parent state). Variables can be defined by the designer and have a *name* and any desired data *type*. Transitions have a *name*, one *source* node, one *target* node, and a transition expression *TE*. Using the EMF representation of the ecore metamodel as Java classes, we used the Java language to write a recursive algorithm that builds the correct statechart by parsing the input liveness formulas.

For editing the statechart we used the GMF technology that allows to associate the ecore metamodel elements with graphical components and dynamically create the graphical model. GMF provides tools for the programmer to define validation rules for the model, if any, and the relevant error or warning messages. Using the GMF API, we also implemented a copy-cut-paste functionality for graphical views, which is not automatically supported by GMF. Thus, the designer can use KSE to edit graphically any part of a statechart, even copying and pasting parts from statecharts of different models.

The IAC2Monas transformation tool has been built using the Xpand language, which is used to define the templates for the required C++ classes. These templates are instantiated using information from the `Statechart` metamodel elements, for example name, label, activity, type, and children of a node.

5 KSE Evaluation

To obtain an empirical evaluation of our CASE tool, 28 ECE undergraduate students taking the Autonomous Agents class at the Technical University of Crete were asked to use KSE and evaluate it in one of their laboratory sessions. The plan of this 2-hour lab session was to go through a short tutorial on using KSE, study a complete SPL Goalie behavior as an example (shown in Figure 6 without the transition expressions), and finally develop their own SPL Attacker behavior using KSE and the functionalities of the Goalie behavior. The predefined functionalities were contained in a Monas source code repository. The students worked in small teams of two or three people per team. None of them had any prior experience with CASE tools, KSE, Monas, SPL, or RoboCup in general. This lab session was run three times to accommodate all students in the four

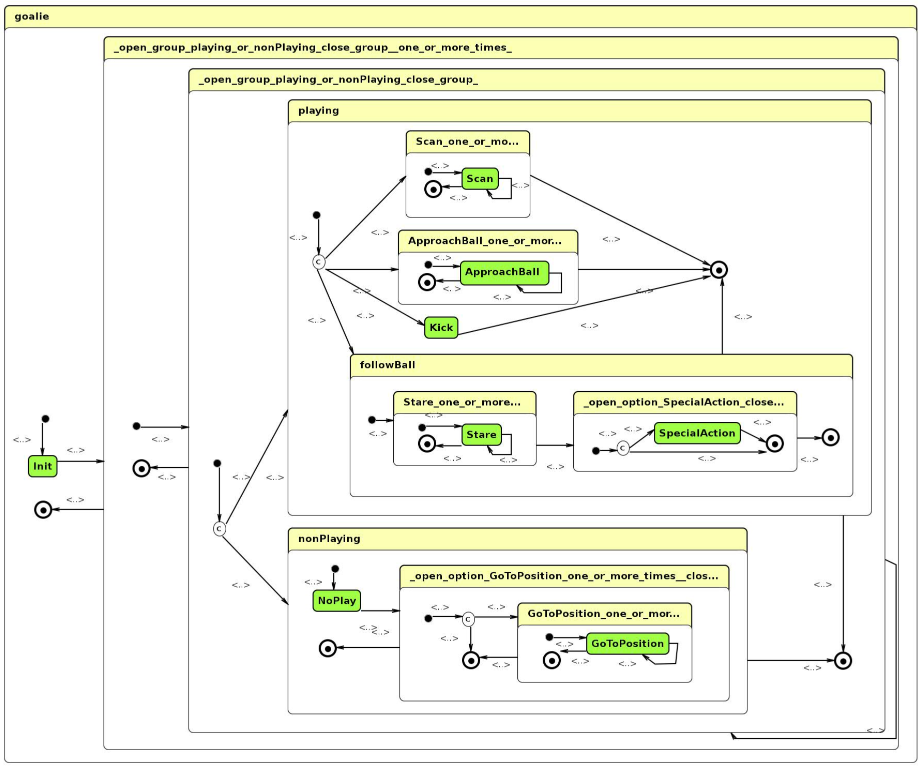

goalie
_open_group_playing_or_nonPlaying_close_group__one_or_more_times_
_open_group_playing_or_nonPlaying_close_group_
playing
Scan_one_or_mo...
Scan
ApproachBall_one_or_mor...
ApproachBall
Kick
followBall
Stare_one_or_more...
Stare
_open_option_SpecialAction_close...
SpecialAction
Init
nonPlaying
NoPlay
_open_option_GoToPosition_one_or_more_times__clos...
GoToPosition_one_or_mor...
GoToPosition

Fig. 6. The statechart of the provided SPL Goalie behavior (expressions not shown)

available work stations. At the end of each lab session, a quick SPL game took place with the four developed attackers split in two teams of two players each.

The results were in general positive for KSE as a CASE tool, but also for the concept of ASEME-based behavior development. Both seemed to be pretty understandable, even though most students were not familiar with Agent-Oriented Software Engineering. All student teams were able to go through the provided material and deliver the requested SPL Attacker behavior. The great bet, won by KSE in this evaluation, was that all student participants succeeded to create a simple SPL robot behavior and even enjoyed watching their players in a game without having to go through the typical lengthy training procedures required for student members of an SPL team.

The participating students were asked to fill in an anonymous user satisfaction questionnaire after the lab session. The total number of responders was 19. A small sample of the user responses is shown in Table 1. The overall assessment of KSE was positive. The main negative comment was that the long transition expressions on the model were cluttering the view of the statechart graph. Based on this comment, in the latest version of KSE the user can choose to hide part(s) of each expression to improve readability of the model.

Table 1. Summarized responses to the KSE user satisfaction questionnaire

Question	Very Easy	Easy	Normal	Difficult	Very Difficult
The liveness formulas was ... to edit.	21.05%	**63.16%**	15.79%	0.00%	0.00%
The statechart was ... to edit.	0.00%	31.58%	**57.89%**	10.53%	0.00%
The navigation to the KSE menu was ...	10.53%	42.11%	**47.37%**	0.00%	0.00%
The use of KSE was ... in general.	0.00%	**57.89%**	26.32%	15.79%	0.00%

6 Related Work and Discussion

Our research revealed two CASE tools that relate most to our work, namely XabslEditor and Yakindu. We briefly review these tools below.

The Extended Agent Behavior Specification Language (XABSL) [13,14] is a simple text-based language for describing behaviors of autonomous agents based on hierarchical finite state machines. XABSL was originally developed for soccer robots behavior specification, but can be used for all kinds of autonomous robots or virtual agents. XabslEditor[4] is a text editor for XABSL, having also the capability to represent graphically the hierarchical finite state machines that describe the agents' behavior. XabslEditor also provides a compiler for XABSL.

The Yakindu toolkit supports the development of both reactive, event-driven and data flow-oriented systems with the help of finite state machines, statecharts (according to Harel), and block diagrams. Yakindu provides graphical modeling tools with integrated validation and simulation, which allow for the early assessment of the models and offers efficient code generators for a target platform.

The main features of the KSE, XabslEditor, and Yakindu tools are summarized in Table 2. KSE compares favorably with the other tools in terms of supported features. A major advantage of KSE for the design process is the analysis tool, which enables the user to abstractly and compactly define the desired behavior using liveness formulas. A small set of liveness formulas can lead to a large statechart model, therefore the user can save a significant amount of time by seeding the design through the analysis tool. As an example, the statechart of the SPL Goalie behavior in Figure 6 was initiated by only four liveness formulas.

The user can configure KSE and choose his/her favorite text editor or programming environment for viewing and editing activities. The default configuration of KSE uses our IAC2Monas code generator for integrating statecharts into our Monas architecture, but KSE can be reconfigured to use any desired source code generator and any grammar for transition expressions in order to generate code for another platform. For example, we have configured KSE to generate code for the popular JADE agent platform using the IAC2JADE tool [15] and a grammar assuming FIPA-ACL[5] communication between the agents.

To facilitate rapid behavior updates, we have configured KSE to store all statecharts and models within the source code repository of our Monas architecture. Thus, all developed statecharts (behaviors) are available at any time and the developer can choose, on the fly, which statechart to execute on the robot.

[4] XabslEditor has been developed by Nao Team Humboldt and is freely available: `www.naoteamhumboldt.de/en/projects/xabsleditor`

[5] The Foundation for Intelligent Physical Agents - Agent Communication Language Specification: `www.fipa.org`

Table 2. Feature comparison of XabslEditor, Yakindu, and KSE

Feature	**XabslEditor**	**Yakindu**	**KSE**
Supported Platforms	java	eclipse helios	linux, windows
Open Source	√	free-ware	√
Model Validation	√	√	√
Analysis Tool			√
Model Simulation		√	
Multiple Editing Tabs	√	√	√
Symbol Auto-Completion	√		
Graphical Editing		√	√
Reusability of Graphical Components			√
Code Generation	√	√	√
Integrated Code Editing	√		√
Customization of Code Generator			√

In general, it is important that the user-defined activities are generic enough to be re-used in a variety of statecharts. This way, a developer can design and test new behaviors, which do not entail the definition of new functionality, just by editing statecharts and reusing existing functionalities.

Recently, we proposed the use of liveness formulas and statecharts also for specifying agent interaction protocols [2]. We showed how an attack protocol can be modeled and then inserted in the respective robot behavior. This feature is fully supported by KSE, as the designer can define the coordinated action and then copy and paste the relevant parts to the interacting robots' statecharts, thus ensuring the correct execution of the protocol. Consider, for example, the following simple attack protocol defining three roles, two instances of *center-for* and one of *center*: the *center* assumes control of the ball and then passes the ball to one of the *center-for*s. The liveness formula for this protocol begins with:

```
attack = center || center-for || center-for
```

Then, in a robot's IAC model the designer defines in a liveness formula that the player will assume either a *center* or a *center-for* role, when attacking:

```
attack = center | center-for
```

When the statechart is generated in the second case, the designer can simply copy the relevant states from the previously edited protocol definition statechart and paste them over the auto-generated *center* and *center-for basic*-states.

7 Conclusion

In this paper, we presented the Kouretes Statechart Editor (KSE), a graphical CASE tool for robot behavior development. KSE offers a number of features that make it suitable for domains, such as RoboCup, where behavior modifications are frequent and require quick and error-proof solutions. The KSE tool is freely available along with other ASEME-based tutorials and tools developed by the RoboCup SPL team Kouretes from `www.kouretes.gr/aseme`.

Acknowledgments. The authors would like to thank Alexandros Paraschos and all members of the Kouretes team for their valuable assistance and our sponsor Chipita S.A.–Molto.

References

1. Baik, J., Boehm, B.: Empirical analysis of CASE tool effects on software development effort. ACIS International Journal of Computer and Information Science 1, 1–10 (2000)
2. Paraschos, A., Spanoudakis, N.I., Lagoudakis, M.G.: Model-driven behavior specification for robotic teams. In: Proceedings of The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Valencia, Spain (June 2012)
3. Harel, D., Naamad, A.: The Statemate semantics of statecharts. ACM Transactions on Software Engineering and Methodology 5, 293–333 (1996)
4. Spanoudakis, N.I., Moraitis, P.: Using ASEME methodology for model-driven agent systems development. In: Weyns, D., Gleizes, M.-P. (eds.) AOSE 2010. LNCS, vol. 6788, pp. 106–127. Springer, Heidelberg (2011)
5. Paraschos, A.: Monas: A flexible software architecture for robotic agents. Diploma thesis, Technical University of Crete, Greece (2010)
6. Vazaios, E.: Narukom: A distributed, cross-platform, transparent communication framework for robotic teams. Diploma thesis, Technical University of Crete, Greece (2010)
7. Spanoudakis, N.I., Moraitis, P.: The agent modeling language (AMOLA). In: Dochev, D., Pistore, M., Traverso, P. (eds.) AIMSA 2008. LNCS (LNAI), vol. 5253, pp. 32–44. Springer, Heidelberg (2008)
8. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys 35, 114–131 (2003)
9. Hayes-Roth, B.: A blackboard architecture for control. Artificial Intelligence 26(3), 251–321 (1985)
10. Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000)
11. Spanoudakis, N.I., Moraitis, P.: Gaia agents implementation through models transformation. In: Yang, J.-J., Yokoo, M., Ito, T., Jin, Z., Scerri, P. (eds.) PRIMA 2009. LNCS, vol. 5925, pp. 127–142. Springer, Heidelberg (2009)
12. ISO/IEC: Extended Backus-Naur form (EBNF). 14977 (1996)
13. Loetzsch, M., Risler, M., Jungel, M.: XABSL - a pragmatic approach to behavior engineering. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5124–5129 (October 2006)
14. Risler, M.: Behavior Control for Single and Multiple Autonomous Agents Based on Hierarchical Finite State Machines. PhD thesis, Technische Universität Darmstadt, Germany (2009)
15. Spanoudakis, N., Moraitis, P.: Modular JADE agents design and implementation using ASEME. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), Toronto, Canada (2010)

A Distributed Cooperative Reinforcement Learning Method for Decision Making in Fire Brigade Teams

Abbas Abdolmaleki[1,2], Mostafa Movahedi, Nuno Lau[1,3], and Luís Paulo Reis[2,4]

[1] IEETA – Institute of Electronics and Telematics Engineering of Aveiro, Portugal
[2] LIACC – Artificial Intelligence and Computer Science Lab., Porto, Portugal
[3] UA – University of Aveiro, Campus Universitário de Santiago, 3810 193 Aveiro, Portugal
[4] EEUM - School of Engineering, University of Minho - DSI, Portugal
Campus de Azurém 4800-058 Guimarães, Portugal
{Abbas.a,nunolau}@ua.pt, mr.mos.movahedi@gmail.com,
lpreis@dsi.uminho.pt

Abstract. Decision making in complex, multi-agent and dynamic environments such as disaster spaces is a challenging problem in Artificial Intelligence. This research paper aims at developing distributed coordination and cooperation method based on reinforcement learning to enable team of homogeneous, autonomous fire fighter agents, with similar skills to accomplish complex task allocation, with emphasis on firefighting tasks in disaster space. The main contribution is applying reinforcement learning to solve the bottleneck caused by dynamicity and variety of conditions in such situations as well as improving the distributed coordination of fire fighter agent's to extinguish fires within a disaster zone. The proposed method increases the speed of learning; it has very low memory usage and has a good scalability and robustness in the case that the number of agents and complexity of task increases. The effectiveness of the proposed method is shown through simulation results.

Keywords: RoboCup Rescue Simulation, Multi agent system, Fire Brigade, Decision Making, Reinforcement Learning.

1 Introduction

Several authors have proposed general models for flexible coordination of agents. However, most of the approaches either are not sufficiently reactive to perform efficiently in real time and dynamic domains or do not provide agents with sufficiently developed social behavior to perform intelligently as members of a team in continuous, multi-objective and complex multi-agent environments [1]. Notable research may be recognized in Stone's and Veloso's work [2] that has been applied with success to RoboCup soccer and network routing.

In order to achieve complex behavior acquisition using machine learning methods, Stone and Veloso [3] proposed to introduce a layered learning system with basic skills such as "shootGoal", "shootAway", "dribbleBall", and so on. Kleiner et al [4] also proposed a multi-layered learning system for behavior acquisition of a soccer robot.

Most implementations of multi-agent coordination frameworks rely on domain specific coordination. Some relevant exceptions may be identified however. An example is Jennings' joint responsibility framework [5], which is based on a joint commitment to the team's joint goal and was implemented in the GRATE* system. One other approach to automated planning is discussed in hierarchical task networks (HTN) where the dependency among actions can be given in the form of networks [6]. Regarding one possible and commonly used test bed for the framework, the Robotic Soccer competition [7, 8, 9], the UVA Trilearn team's coordination graphs provide a way to parameterize a coordination structure over a continuous domain [10].

To manage coordination between agents three options are proposed in [11]: environment partitioning, centralized direct supervision and decentralized mutual adjustment. Among these three approaches, the decentralized approach is more flexible but this does not necessarily provide better results. Considering Robocup rescue simulation as another test bed, in [12] a hybrid approach is proposed to take advantage of both centralized and decentralized approaches. In order to make a decision about the number of ambulances which should cooperate in rescue civilian, evolutionary reinforcement learning is utilized by [13]. In [14] a fire brigade learns to do task allocation and how to choose the best building to extinguish to maximize the final score. In [15] fire brigades learn how to distribute themselves within the city using neural reinforcement learning.

The authors recently developed a reinforcement learning based model for optimizing the task of a single fire fighter agent [16]. This model uses a reinforcement learning method based on the Temporal Difference (TD) methodology. Temporal difference methods update the estimated data without waiting for the final outcome, allowing prompt reactions in complex environments like disaster spaces. In this paper we have extended this approach and developed a distributed coordination and cooperation method based on reinforcement learning to enable a team of homogeneous, autonomous fire fighter agents, with similar skills to accomplish complex allocation of tasks, with emphasis on extinguishing fires task in disaster space. The major contribution made in this paper is to introduce a distributed cooperative algorithm based on reinforcement learning for firefighting agents to save a city and civilians from fire at a fire site[1]. It is a distributed coordination because each agent makes its decision by itself and there is no central command. The rest of this paper is organized as follow. In next section the problem is detailed and assumptions are presented. In Section 3 we explain the test bed which is used to implement and test our approach. Section 4 discusses how to design and model the problem for applying reinforcement learning, in detail. In Section 5 details of implementation are presented and achieved results are shown. Section 6 concludes.

2 Problem Formulation

Disaster space firefighting presents a complex task allocation problem in a very dynamic and uncertain environment. Fire brigades are responsible for controlling fire. The most important issue is to select and extinguish the best fire point to reduce damage of the civilians and city. We have decomposed the firefighting task (Fig. 1) into some important subtasks. The two most important subtasks are:

[1] A fire site is an area within a neighborhood consisting of burning buildings.

1- In which way to allocate large number of resources (Firefighters agents) to various fire sites.
2- How allocated agents cooperate and coordinate in a decentralized way at each site to control the fire.

In this paper we propose a solution for the second subtask which is seeking an acceptable distributed cooperation strategy for firefighter agents to control a fire site using reinforcement learning.

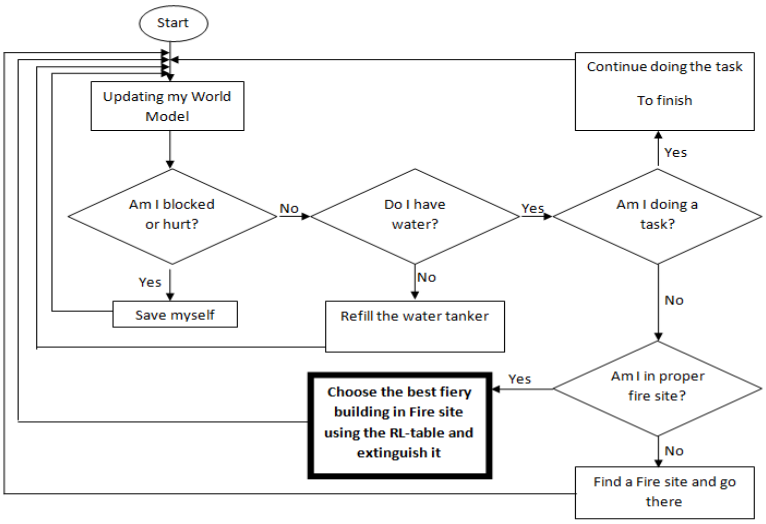

Start
Updating my World Model
Continue doing the task To finish
Yes
Am I blocked or hurt?
No
Do I have water?
Yes
Am I doing a task?
Yes
Save myself
No
Refill the water tanker
No
Yes
Am I in proper fire site?
Choose the best fiery building in Fire site using the RL-table and extinguish it
No
Find a Fire site and go there

Fig. 1. Flowchart of firefighting task, the bold box is the sub-task which we have focused on in this paper

Like humans, robots potentially work better as a team than working alone in challenging domains. By cooperation, robots can complete the tasks faster, with more robustness and also complete tasks which are impossible for a single robot. However for effective cooperation, there are many difficulties in uncertain and dynamic environments, such as the presence of noise, communication problems, resource failures, and difficulties of mobility and computation.

Considering the aforementioned challenges we make some assumptions to present a distributed cooperative reinforcement learning method, which are as follows:

#Assumption 1: All the agents or resources have the same knowledge of the fire sites and fiery buildings. It means we assume that the communication between robots is good enough to share their knowledge about the environment.

#Assumption 2: No routes are blocked, so there is no problem of mobility for agents.

#Assumption3: As firefighter agents need to refill their water tanks, we assume there are stations from which firefighters can do so, so there is no resource failure.

In a rescue simulation, fire fighter agents must perform a sequence of actions to carry out their firefighting task at a fire site efficiently. Each action alters the environment and also influences their choice of the next action. The goal of the agents is to achieve the best score at the end of the simulation. Agents should select a sequence of actions which maximizes the score. Due to the limits of cognition of the agents and the fact that agents don't know a priori the effect their action will have upon the environment, using reinforcement learning is a suitable method to solve this problem. The mechanism which we have used to teach the agents is based on the Temporal Difference method of reinforcement learning [17]. Since TD methods update their estimates based in part on other learned estimates, without waiting for a final outcome (bootstrapping), this method is applicable for complex environments such as rescue simulations in which an action should be selected in each cycle.

Unfortunately, finding optimal planning solution for Multi-agent systems challenges are typically NP-hard. The goal then, is to produce acceptable and efficient distributed cooperative planning solutions based on reinforcement learning.

Our algorithm is based on an offline look-ahead search to find the best action to execute at each possible situation. In this problem, first we should define proper states and actions so that the learning task is to find the best action for each situation. The output of the learning would be a table in which for each possible situation there is an associated action.

3 Test Bed

RoboCup rescue simulation is a simulation of environment of a city where an earth quake has happened. The aim of creating this environment is to learn the best rescue strategies for humans and also fire extinguishing tactics to be used within earthquakes in real life. It is currently a major league in RoboCup simulation competitions. The simulation consists of a kernel as the main part of the simulation, and some simulators which simulate the earthquake, fire,traffic, blockades and civilians of the city and also a viewer which shows how the city changes over time. There are three kinds of agent: Police Force, Ambulance and Fire Brigade Agents which are present in the simulation to perform the rescue operation. Police Force Agents control the traffic in the city and open blockades of closed roads. The Ambulance agent's duty is to locate the injured civilians and give first-aid to them until they are retrieved from collapsed buildings as well as carrying them to refuges. Fire simulator used to simulate the spread of fire over buildings, taking into account building material, wind and temperature of building. The main job of the Fire Brigade Agents is to extinguish fires by spraying water in its tanker (the capacity is limited) on buildings which decreases the temperature of buildings. If fire fighter agent sprays water on a cool building that building will get water damage. Traffic simulator determines how agents navigate around on roads and buildings. The performance score of a simulation is computed from the quantity of surviving civilians and the extent of damage to the city. When the fire brigade agents want to extinguish fire they have to prioritize which of the burning buildings to extinguish and in which order. We have applied our method to this environment to find an policy for finding an action in each environmental state.

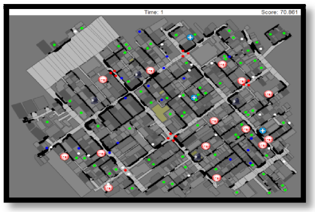


Fig. 2. Rescue Simulation Environment

Fig. 2 shows a screenshot of the simulation environment. It displays map of Kobe city after an earthquake. The blue, white and red circles represent the Police force agents, Ambulance agents and Fire brigade agents respectively. The bright and dull green circles display healthy and hurt civilians and black circle represent the dead civilian. The yellow, orange, dark red and black building represents level of fieriness of a building. Yellow shows the lowest level of fieriness of a building and black shows a burned out building. There is also a special type of buildings: the buildings that are marked with home icon are refuges where saved civilians are taken. Black areas on the roads represent blockades. The simulation runs for 300 time steps.

In order to reduce the complexity of simulation process we designed and implemented a simple rescue simulation environment according to assumption 1-3. This system has all features we need such as a fire and traffic simulator. Our simple rescue simulation is much faster than the official rescue simulation while maintaining the necessary capabilities. In Fig. 3 a screen shot of the simple rescue simulation is displayed. This environment has a fire simulator with the same algorithms of official Rescue Simulation since it is a standard system. Burning of buildings and fire spread is the same as the original Rescue Simulation. Also the algorithm to calculate the health of civilians is the same as the one in misc simulator[2] of Rescue Simulation.

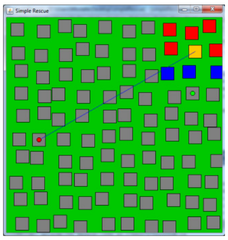


Fig. 3. Simple rescue simulation

The learning processes of agents and test phases are performed within this simulator and the results potentially can be used in original rescue simulation.

4 Design

This section presents necessary issues including a description of the environment, the design of the reward function and the learning algorithm used in our approach. Finally a proposed learning procedure for the problem is presented. But first a brief explanation of reinforcement learning is presented.

4.1 Description of Environment

In this section we define actions for agents and possible states for environment.

First the possible actions for agents should be determined. So agents in each state can perform one of following actions:

[2] The misc simulator calculates the health of civilians within Rescue Simulation.

1- extinguishEasyestBuilding: Extinguish the building that has lowest temperature in firesite.
2- extinguishNearestToCivilian: Extinguish the nearest fiery building to a civilian in firesite.
3- extinguishNearestToCenter: Extinguish the fiery building in a firesite which is closest to the center of the city.
4- extinguishNearestToMe: Extinguish the nearest building in a firesite to the firefighter.

These four actions are the most reasonable actions that a firefighter agent can take in different situations.

Although in Temporal Differential methods the model of the environment is not required, it needs a clear description of the states of the model. So in this section we describe the rescue simulation environment with some discrete and finite states. The states of environments are modeled by the following features.

1- freeEdges: The number of edges that fire can spread from represented by 0,1,2,3 or 4. For example in Fig.3 the fire can spread only from the left side, so the number of freeEdges is 1.
2- distanceFromCenter: The distance of nearest fiery building to the center of city. This is low, medium or high.
3- distanceFromCivilian: The distance from the nearest fiery building to a civilian. This is low, medium or high
4- volumeOfFieryBuildings: The total volume of fiery buildings, classified as either veryLow, low, medium, high, veryHigh or huge.
5- IsExtinguishEasyestBuilding: If currently any firefighter is extinguishing the easiest building (action 1) in firesite. This is True or False.
6- IsExtinguishNearestToCivilian: If currently any firefighter is extinguishing the nearest fiery building to a civilian (action 2) in the firesite. This is True or False.
7- IsExtinguishNearestToCenter: If currently any firefighter is extinguishing the nearest fiery building to the city center (action 2) in the firesite. This is True or False.
8- IsextinguishNearestToMe: If currently any firefighter is extinguishing nearest fiery building to itself (action 4). This is True or False.

The states 1 to 4 are enough to describe the environment for an agent. To allow agents to cooperate effectively they need to know about other agents. So states 5 to 8 describe the other agents in an environment. These states give an idea to agents what the other agents are doing at each step which provides them with a distributed cooperation. Good communication between agents is therefore required, and this is assumed in the problem formulation section.

There are 5×3×3×7×2×2×2×2=4320 different environmental states. These states can describe all situations in all fiery scenarios.

The problem is to find the best action in each state that leads to best score at the end of the simulation.

4.2 Reward Function

The reward function plays an important role in reinforcement learning as it directs the learning process towards a solution. In a burning city, the world will never improve with time; hence the agent will always incur a penalty in the reward function. The fire brigade must minimise the penalty score.

The reward function used is defined as follows:

- For each *waterDamaged* building (a building which is extinguished and is damaged because of too much water) in each time step the agent receives a reward -1.
- For each fiery building in each time step the agent receives a reward -2.
- For each burned out building in each time step the agent receives a reward -3.
- For each damaged civilian in each time step the agent takes reward *HP-1000*. The *HP* (Health Points) of each healthy civilian is 1000 and if the building that contains a civilian ignited, the civilians' *HP* reduces over the time.

The first reward causes the fire brigade to extinguish fire without wasting water. The second and third rewards force the fire brigade to extinguish fire as quickly as possible and the final reward motivates the fire brigade to save civilians from fire.

4.3 Learning Algorithm

Fire brigade agents learn which of their possible actions are most valuable (leads to the best score) for a particular environmental state. To do this, the agents are taught by an on-policy TD control method called SARSA [17]. Reward values (Q values) of each action in each state were updated by formula (1) considering their rewards or penalties.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{1}$$

In above equation s_t is previous state before taking action a, s_{t+1} is current state after action a, r is reward received from the environment after taking action a in previous state and reaching the current state, a is the last executed action, Q is the Q-Table, *alpha* is learning rate and *gamma* is discount factor.

4.4 Learning Procedure

In reinforcement learning methods rewards provide guidance to the agent. If an agent receives high rewards it understands that it performed a useful action sequence; conversely poor rewards suggest a poor choice of actions. If an agent earns low rewards most of the time, convergence is poor and the best action cannot be found. In a very complicated environment such as rescue simulation, the problem search space is huge. In this case the agent may continually get bad rewards and so not learn the optimum policy. To overcome this problem we carried out our learning phase step by step. First we trained just a single agent in 24 scenarios ranging in difficulty from easy to complicated which is explained in [16]. Then we used the obtained Q-table (ignoring cooperation states which reference other agents) from the previous step as the initial knowledge of four agents. Next, we started training four agents in 8 different scenarios. In each scenario the firefighters randomly use one of the obtained Q-Tables from its previous learning scenarios as the initial knowledge to output an improved Q-Table containing more records and which has been informed by a greater number of states.

5 Implementation and Results

This section presents the results of using the previously explained method in simplified rescue simulation. In this section the scalability and robustness of the method will be discussed. We will discuss the results in three subsections. First the learning procedure and results in training scenarios will be presented. Then the effectiveness of the method on three test scenarios is shown. Finally we will demonstrate the scalability of method by changing the number of agents.

5.1 Cooperative Multi-agent Learning

In order to train the firefighters, the parameters γ and α of the Q-Table are initially set to 0.5 and 0.7 respectively, based on trial and error. The value of e factor in the *e-greedy* selection algorithm is set in each episode based on the formula $e = e * 0.9^{episode}$. This formula causes agents in early episodes to favor exploration of the search space and in later episodes to use obtained experience from previous episodes to try to converge to a solution. First we trained just a single agent in scenarios ranging from easy to difficult [16]. Then, in order to teach coordination between agents, 8 scenarios, each using 4 agents were simulated. In the first scenario the initial Q-Table for all agents is the Q-Table obtained from the previous training of a single agent. In successive simulated scenarios the initial Q-Table is the same for all agents and is chosen randomly from the recorded Q-Tables of each agent at the end of the preceding scenario. The end condition of each training phase for a scenario is when agents converge to a policy which does not change for 20 episodes. This will be considered as a learned policy by agents. It was interesting to note, that at the end of all eight scenarios, all the agents had converged to produce the same Q-tables. Agents learn appropriate actions for each state after training in all scenarios, because the scenarios and learning procedure have been carefully designed so that every state is visited several times. In table 1, characteristics of training scenarios and the result of training is presented. Score in table 1, is calculated based on number of dead civilians and burnt out buildings.

Table 1. Main characteristics of the scenarios used for cooperative learning.and results

Scenarios	FireSite Position	Initialized amount of fire	Position of civilians related to Firesite	Convergence Episode	Score
Scenario1	City Center	Small	Surrounding	55	9.977 out of 10
Scenario2	City Center	Medium	Surrounding	101	8.795 out of 10
Scenario3	City Center	Medium	West	116	10.787 out of 11
Scenario4	City Center	Medium	East	128	8.611 out of 9
Scenario5	North West corner	Small	East and South	41	9.891 out of 10
Scenario6	North West corner	Medium	East and South	74	9.629 out of 10
Scenario7	North West corner	Big	South	73	8.973 out of 10
Scenario8	South	Big	Surrounding	59	11.31 out of 12

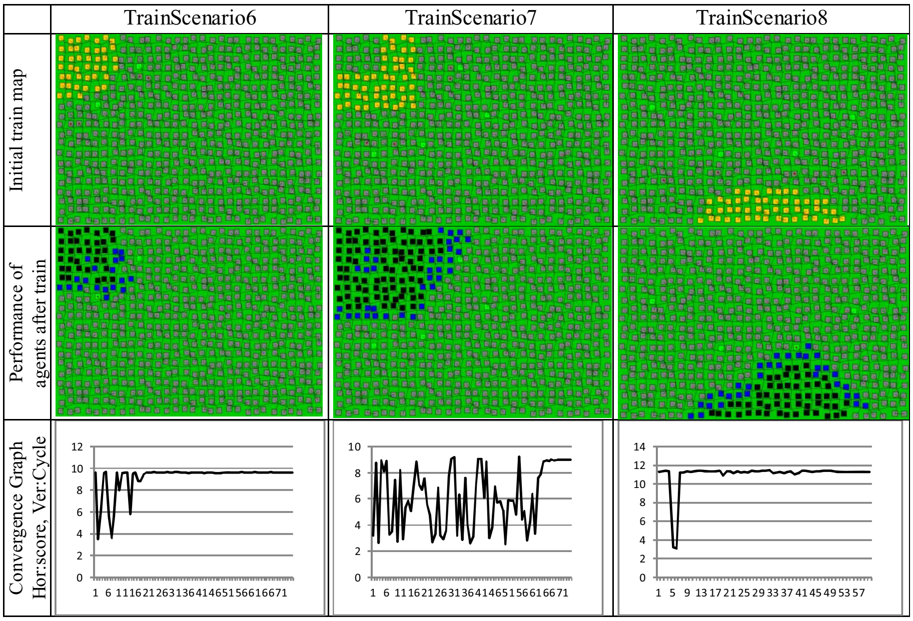


Fig. 4. Training scenarios(6,7,8) and performance of agents on train scenario after learning as well as convergence graphs

Fig.4 shows graphically Training scenarios and performance of agents on 3 train scenarios after learning as well as convergence graphs. The achieved result proved that agents after training can control fire effectively.

5.2 Evaluation in Test Scenarios

After training 4 agents in 8 different scenarios, we tested these 4 agents in 3 new scenarios and compared the performance of these agents with agents which were trained using a single agent approach [16]. To analyze the performance of trained agents, the test scenarios used were more difficult than the scenarios in which the agents were trained. Table 2 presents the test results and Figure 5 shows graphically the performance of agents in 2 train scenarios (1 and 2). The results show the cooperatively trained agents have much better performance than agents which were trained in non-cooperative mode. Also, the results achieved by cooperative agents are very good for complex scenarios, as the agents saved all civilians and controlled fire effectively, limiting fire damaged areas. This shows that the method provided a robust approach which can be applied in scenarios that are very different from those used in the training phase.

Table 2. Results in test scenarios for cooperative and non cooperative learning approaches

Scenarios	Time to complete task		Number of burned out buildings		Score		Max score
	Without Coord.	With Coord.	Without Coord.	With Coord.	Without Coord.	With Coord.	
Scenario 1	436	290	410	278	2.017	9.15	12
Scenario2	490	203	541	115	2.564	10.808	12
Scenario3	473	410	570	337	0.27	5.582	10

5.3 Evaluation of Scalability

To test the scalability of proposed method, we used scenarios where the number of firefighters is different than the number of agents used in the training scenarios. The results (presented in table 3) show that the agents can cooperate well and also that their performance is better than firefighters trained in non-cooperative mode.

Table 3. Results in test scenarios where the number of firefighters is different than the number of agents used in the training scenarios for cooperative and non cooperative learning approaches

Scenarios	Number of firefight ers	Time to complete task		Number of burned out buildings		Score		Max score
		Without Coord.	With Coord.	Without Coord.	With Coord.	Without Coord.	With Coord.	
Scenario 1	7	361	246	268	167	8.221	10.257	12
Scenario2	8	292	113	160	77	10.34	11.198	12
Scenario3	8	456	294	318	112	6.76	8.788	10

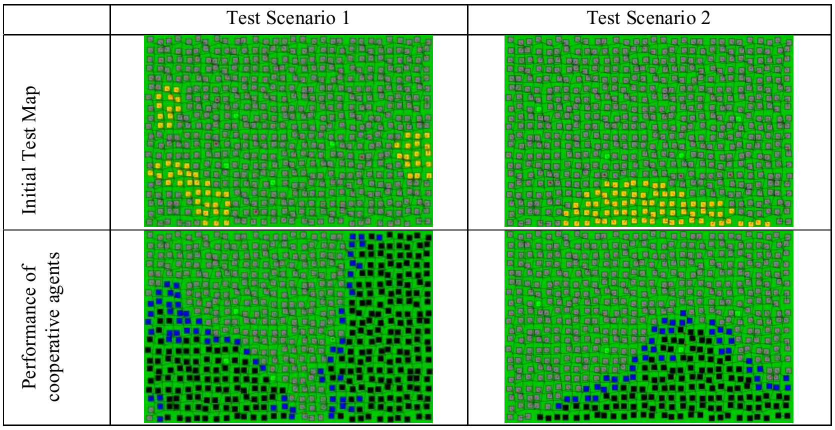


Fig. 5. Results in test scenarios, Comparison of cooperative agents and non-cooperative agents in test scenario

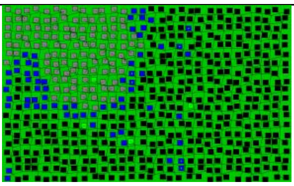
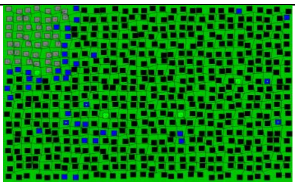
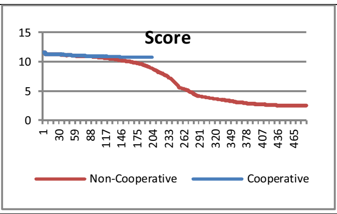
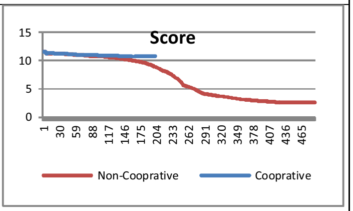
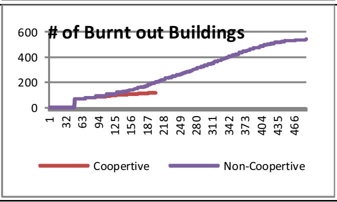
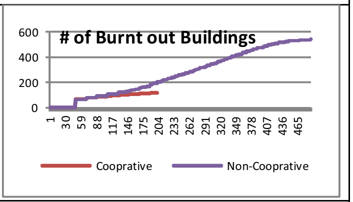
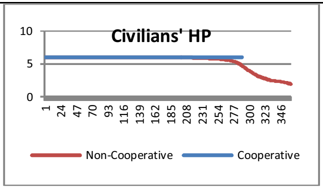
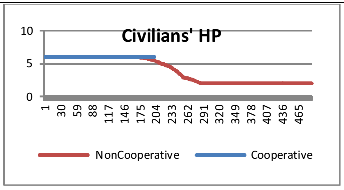
Performance of non-cooperative agents		
The comparison of archived scores Ver:Score,Hor:Cycle		
Comparison of burnt out buildings Ver: # of burnt out buildings , Hor:Cycle		
Comparison of Civilian´s HP Ver: Civilian´s HP , Hor:Cycle		

Fig. 5. (*continued*)

6 Conclusions

In this paper we discussed the use of temporal difference learning to find the optimum policy for fire extinguishing tasks of a group of firefighter agents, and results showed that agents trained by TD exhibit a good performance at extinguishing fires. This method has a good robustness and scalability.

Acknowledgments. The first author is supported by FCT under grant SFRH/BD/81155/2011. This work has been partially funded by FCT Project ACORD - Adaptative Coordination of Robotic Teams (PTDC/EIA/70695/2006).

References

1. Reis, L.P., Lau, N., Oliveira, E.C.: Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents. In: Hannebauer, M., Wendler, J., Pagello, E. (eds.) Reactivity and Deliberation in MAS. LNCS (LNAI), vol. 2103, pp. 175–197. Springer, Heidelberg (2001)

2. Stone, P., Veloso, M.: Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork. Artificial Intelligence 110(2), 241–273 (1999)
3. Stone, P., Veloso, M.: Layered approach to learning client behaviors in the robocup soccer server. Applied Artificial Intelligence 12(2-3) (1998)
4. Kleiner, A., Dietl, M., Nebel, B.: Towards a Life-Long Learning Soccer Agent. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 126–134. Springer, Heidelberg (2003)
5. Jennings, N.R.: Controlling Cooperative Problem Solving in Industrial Multiagent Systems using Joint Intentions. Artificial Intelligence 75(2), 195–240 (1995)
6. Lekavý, M., Návrat, P.: Expressivity of STRIPS-Like and HTN-Like Planning. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2007. LNCS (LNAI), vol. 4496, pp. 121–130. Springer, Heidelberg (2007)
7. Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.): RoboCup 2005. LNCS (LNAI), vol. 4020. Springer, Heidelberg (2006)
8. Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.): RoboCup 2006. LNCS (LNAI), vol. 4434. Springer, Heidelberg (2007)
9. Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.): RoboCup 2007. LNCS (LNAI), vol. 5001. Springer, Heidelberg (2008)
10. Kok, J.R., Spaan, M.T.J., Vlassis, N.: Multi-Robot Decision Making using Coordination Graphs. In: Proceedings of 11th International Conference on Advanced Robotics (ICAR), Coimbra, Portugal, pp. 1124–1129 (2003)
11. Paquet, S., Bernier, N., Chaib-draa, B.: Comparison of different coordination strategies for the roboCupRescue simulation. In: Orchard, B., Yang, C., Ali, M. (eds.) IEA/AIE 2004. LNCS (LNAI), vol. 3029, pp. 987–996. Springer, Heidelberg (2004)
12. Mohammadi, Y.B., Tazari, A., Mehrandezh, M.: A new hybrid task sharing method for cooperative multi agent systems. In: Canadian Conf. on Electrical and Computer Engineering (May 2005)
13. Martínez, I.C., Ojeda, D., Zamora, E.A.: Ambulance decision support using evolutionary reinforcement learning in robocup rescue simulation league. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 556–563. Springer, Heidelberg (2007)
14. Paquet, S., Bernier, N., Chaib-draa, B.: From global selective perception to local selective perception. In: AAMAS, pp. 1352–1353 (2004)
15. Amraii, S.A., Behsaz, B., Izadi, M.: S.o.s 2004: An attempt towards a multi-agent rescue team. In: Proc. 8th RoboCup Int'l Symposium (2004)
16. Abdolmaleki, A., Movahedi, M., Salehi, S., Lau, N., Reis, L.P.: A Reinforcement Learning Based Method for Optimizing the Process of Decision Making in Fire Brigade Agents. In: Antunes, L., Pinto, H.S. (eds.) EPIA 2011. LNCS, vol. 7026, pp. 340–351. Springer, Heidelberg (2011)
17. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)

Active Scene Text Recognition for a Domestic Service Robot

José Antonio Álvarez Ruiz*, Paul Plöger, and Gerhard K. Kraetzschmar

University of Applied Sciences Bonn-Rhine-Sieg
Computer Science Department
Sankt Augustin, Germany
jose.alvarez@smail.inf.h-brs.de,
{paul.ploeger,gerhard.kraetzschmar}@h-brs.de

Abstract. We developed a scene text recognition system with active vision capabilities, namely: auto-focus, adaptive aperture control and auto-zoom. Our localization system is able to delimit text regions in images with complex backgrounds, and is based on an attentional cascade, asymmetric adaboost, decision trees and Gaussian mixture models. We think that text could become a valuable source of semantic information for robots, and we aim to raise interest in it within the robotics community. Moreover, thanks to the robot's pan-tilt-zoom camera and to the active vision behaviors, the robot can use its affordances to overcome hindrances to the performance of the perceptual task. Detrimental conditions, such as poor illumination, blur, low resolution, etc. are very hard to deal with once an image has been captured and can often be prevented. We evaluated the localization algorithm on a public dataset and one of our own with encouraging results. Furthermore, we offer an interesting experiment in active vision, which makes us consider that active sensing in general should be considered early on when addressing complex perceptual problems in embodied agents.

Keywords: Scene text recognition, active vision, domestic robot, pan-tilt, auto-zoom, auto-focus, adaptive aperture control.

1 Introduction

Beyond being a simple commodity, domestic service robots might open the doors to a more fulfilling life to the elderly and handicapped. However, for robots to perform assistance tasks under very complex and *dynamic* environments, great *robustness* and *flexibility* are required. For example, the agents should gain information about the environment, the agent's situation in it, and that of other agents. For this, perceptual processes have to turn the raw sensory data into higher-level representations. In this investigation, we work towards the acquisition of information from a source seldom exploited in robotics, text embedded in

* The Master's degree of José Antonio Álvarez Ruiz was funded with a CONACYT-DAAD scholarship.

images, commonly referred as *scene text*. Text is a valuable source of information because: **1)** It is readily available in human made environments **2)** Humans make extensive use of it **3)** It contains semantic information. In the end, text provides us humans with the information needed to identify and compare products at the supermarket, find our path at the airport, ordering at a restaurant, etc. A potential application of *Scene Text Recognition* (*STR*) in robotics is product identification, which is generally performed with some sort of appearance based classification. However, appearance based methods suffer of an *inherent lack of generalization* because the appearance of products changes over time and across vendors. If a robot could read text written on boxes and bottles, and understand it, it would result in a more general solution.

STR is a very challenging topic –not to be confused with traditional *Optical Character Recognition* (*OCR*)– because scene text is known to have a large intra-class variance in terms of font, color, layout, symbol repertoire, etc. and the presence of background clutter. Nevertheless, advances in *STR* are not only applicable in robotics, but also profitable by visually impaired and blind humans. Therefore, we consider *STR* to be an important research topic and hopefully, with this article, we raise more interest in it within the robotics community. Although we deal with a perceptual task, our approach diverts respect to the traditional and still often applied conception in computer vision that: sensation, perception and cognition are isolated processes previous to actuation. Under such paradigm, commonly referred as *passive vision*, the perceptual system is limited to operate using the raw data captured by the sensors "as is". This "sensation-followed-by-idea-followed-by-movement" lacks on "psychological adequacy" according to [6]: *"We begin not with a sensory stimulus, but with a sensorimotor coordination... In a certain sense it is the movement which is primary, and the sensation which is secondary, the movement of the body, head, and eye muscles determining the quality of what is experienced"*. Therefore, our robot does not obtain information by plain observation, but also by interaction and selection of stimuli using a *Pan-Tilt-Zoom* (*PTZ*) camera, in such a way, that the agent gains control of *"what to see"* and *"how to see it"* [13].

2 Related Work

Fibonacci search was introduced as an effective methodology for searching optimal focus values using the tenengrad operator in [11]. A system to optimize focus and aperture, based on a hierarchy of artificial neural networks (ANN) was described in [13]. In [14], a system for the extraction of low-resolution text was developed. The system first locates text areas and then uses a *PTZ* camera to capture and assemble a high-resolution mosaic of each region. The use of a polynomial zoom model is mentioned but no details were given. Recently, [17] developed a text localization system for a robot. That work is very relevant because it also includes the extraction of semantic information from the text using probabilistic models and textual web-search. [10] created a robot system with text reading capabilities for aiding visually impaired humans in navigation tasks. The robot, equipped with a *Pan-Tilt Unit* (*PTU*), was designed to read room numbers using a template matcher. However, the authors placed very strong assumptions, such as: possible characters are limited to numbers and to A-E characters. [18, 19] developed a text localization system based on *Discrete Cosine Transform* (*DCT*), and a text tracking system. A *PTU* was used to take a panoramic capture from which text is detected. In those publications active vision is limited to the capture of a mosaic to increase the *Field of View* (*FOV*), but no adaptive actions were performed. [4] introduced a set of features for text discrimination. Those features are calculated from *sub-regions (blocks)* embedded within the detection window that were found to exhibit a distinctive behavior for text. Several statistical measures were computed and combined from the blocks and used to train a *Cascade of Boosted Classifiers* (*CoBC*) with asymmetric adaboost as stage classifiers. [15] used a similar block layout to delimit regions from which *Histogram of Oriented Gradients* (*HOG*) features were extracted to train the first layers of a *CoBC*. The successive layers used *Local Binary Patterns* (*LBP*) and multi-scale *LBP*. [16] extended their previous work in [15] to use two *Conditional Random Fields* (*CRF*) to filter non-text connected components. An image operator called *Stroke Width Transform* (*SWT*), which proved to be very useful and yield better results than other *STR* methods was introduced in [7]. We introduced a *Connected Component* (*CC*) based *STR* system in [1, 3]. However, it performs poorly on low-resolution text. Besides, being a passive *STR* system, it is unable to adapt to different image acquisition conditions, which limits its usefulness in the real-world.

3 Text Localization

Text localization, i.e. to identify and delimit image regions that contain text, is generally the first step in *STR*. This task is difficult because of the large *intra-class variance* of text, lack of prior-knowledge on the scale, orientation, etc. and the presence of background clutter, which might generate similar visual stimuli as text, e.g. windows of a building, a fence, etc. To find text regions, we pass a sliding window through the image at each possible location and at different scales. At each location and scale, a set of discriminative features are extracted from the image inside the detection window. Then, a classifier uses the feature values to assign a *confidence score* to each detection window, higher scores indicate higher probability of text and conversely. Finally, a *confidence map* generated during classification is thresholded and smoothed, and the bounding rectangle of each remaining segment stands for a text region. Localization is a canonical example for *rare-event detection* [21], in which the expected amount of content for the positive class (*text*) is much smaller than for the negative class (*non-text*). Such conditions, along with the number of detection windows that need to be classified, also make of text localization a difficult problem. We use a *CoBC* [22] with *asymmetric adaboost* [21] as stage classifiers and *decision trees* [2, 20] of depth two as weak learners. This particular classification framework is specially well suited for problems with skewed class distributions. Besides, the *CoBC* is computationally efficient in comparison to other methods because it performs the

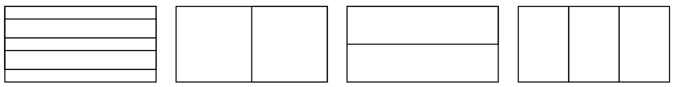


Fig. 1. Example of blocks, each inner rectangle represents a block. The outer rectangles represent the detection window.

classification in stages of increasing complexity. In the initial stages, the stage classifiers are very simple and operate on low-dimensional feature spaces, and still they are able to discard a large amount of the non-text detection windows. Later stages, require of more complex classifiers operating over feature spaces of a higher dimensionality. However, thanks to the decision tree algorithm, features are calculated as needed, instead of pre-calculating the complete feature vectors. Unfortunately, these concepts will not be discussed in detail due to space constraints.

3.1 Feature Space

To train the *CoBC*, we used features that have been reported to perform well in our domain. The features are extracted from sub-regions within the detection window called *blocks* [4]. The blocks are arranged in such way, that the features extracted from them exhibit low entropy, see Fig. 1. A first set of features we use, introduced in [4], is based on mean and standard deviation values either of the intensity image, the intensity gradient magnitude or the x or y intensity gradients; we will refer to these features as "Chen". A second set of features is *HOG* [5], having a *HOG* per-block as defined in [15]. Roughly, the gradient orientations are decomposed in a set of bins. Then, each pixel within a block, casts an orientation vote in the corresponding bin in the *HOG* of that block. The contribution of each pixel is weighted by the magnitude of the gradient at that location. To minimize aliasing effects, we interpolate the values accumulated in each histogram. From now on, Chen and HOG features will be referred as "raw". Furthermore, the raw feature values are processed to transform them into *log-likelihood ratios* [4]. To this end, for each raw feature and possible combination of pairs of raw features, we estimate the conditional probability density of the text and non-text classes. In the case of feature pairs, this process creates new features from the raw features. To model each probability density distribution we use a *Gaussian Mixture Model* (*GMM*) trained with the *Expectation Maximization* (*EM*) algorithm. The optimal number of components per-mixture is estimated with the *Bayesian Information Criterion* (*BIC*) [8].

4 Active Vision Module

Images of poor quality can easily hinder the performance of a *STR* system, for example, due to a loss of contrast between text and the background, or lack of spatial resolution for the task. To cope with this we developed three active vision behaviors, namely: **1)** *Auto-focus*, to prevent blur by defocus **2)** *Adaptive Aperture Control* (*AAC*), to widen the dynamic range of the capture; both aimed to retain the contrast of the text regions by optical means, and **3)** *Auto-zoom*, to acquire high-resolution images. The sensory system we use consists of a SONY camera model VFW-VL500 and a Directed Perception *PTU* model 46-17. The sensory space we consider is formed by the parameters: pan, tilt, zoom, aperture and focus. The active vision behaviors have three major components, either implicitly or explicitly: **1)** *Quality metrics*, that assign a quantitative value to the quality of an image **2)** *Actuation*, to change the configuration of the sensory system by manipulating its electrical and mechanical components **3)** *Search strategies*, to explore the configuration space of the sensory system and find desirable configurations.

Our active vision module has an *initialization phase* and a *recognition phase*. In the initialization phase, the camera is prepared to localize candidate text regions in the scene. First, we set the zoom to its minimum value to maximize the *FOV* and set the focus to its maximum value to produce sharp images of relatively distant objects. Furthermore, the *AAC* behavior corrects the camera aperture according to the illumination conditions of the scene. Once the sensory system has been set up for the scene, we store the current configuration of the sensory system. Afterwards, we capture a frame and localize text regions in it using the algorithm described in Sect. 3 to obtain a set of bounding rectangles of candidate text areas; most of which correspond to real text, and eventually some false positive regions. Finally, a priority calculated from the text confidence map is assigned to each candidate region. During the recognition phase, the regions are attended one by one in order of descending priority as follows. Each candidate region is centered and zoomed-in, in order to capture a high-resolution image. Then, the aperture is optimized again and auto-focus is executed to acquire a sharp image. After processing each region, the sensory system is set back to the previously stored configuration before the next candidate region is processed, see Fig. 2.

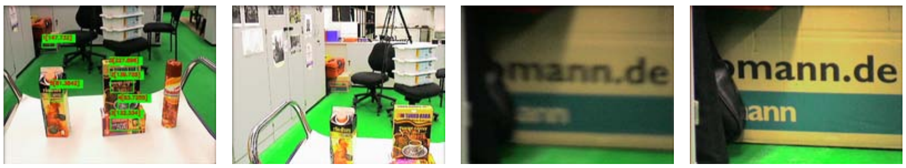


Fig. 2. Example of active STR. From left to right: Localization results after the initialization phase, centering candidate region 1 (a carton box at approximately 3 m w.r.t the camera, the top-left candidate region), after zooming-in, after AAC and auto-focus.

4.1 Auto-focus

Real world cameras require of lenses to focus the light passing through the aperture against the *image plane*, where the imaging sensor is located. However, fixed lenses cannot adequately focus light coming from arbitrary distances. In general, given a lens with a focal length f, only objects located at a distance d_{out} in front of the lens will produce a sharp image on a plane behind the lens located at a distance d_{in}, called *focal plane.* Therefore, objects at different distances will have different focal planes. Light rays coming form objects that lie either closer or farther than d_{out} will be projected as a circle of radius r instead of a point over the focal plane. However, since the sensor resolution is limited, if the radius of a blur circle is small enough, the defocus effects will not be resolved by the sensor and will not be observable; this range of distances is known as *Depth of Field* (*DOF*). Auto-focus consist thus, of changing the distance d_{in} so that the focal plane of a certain object of interest is aligned with the image sensor. The acquisition of focused images is desirable in *STR* because text can be considered as *high-spatial-frequency* content, which is smoothed due to defocus, leading thus to weak intensity gradients. To optimize the focus, we measure the image quality using the *thresholded gradient magnitude operator* (also known as tenengrad operator) defined in Eq. 1, and use *Fibonacci search* to find the maximum [11].

$$Tenengrad(|\nabla\mathcal{I}|) = \sum_x \sum_y |\nabla\mathcal{I}|(x,y) \ \text{ for } |\nabla\mathcal{I}|(x,y) > \tau \ . \tag{1}$$

Where $|\nabla\mathcal{I}|$ is the magnitude of the intensity gradient and τ is a threshold [1]. For a certain *DOF*, the tenengrad operator will exhibit its maximum when an object in the *Region of Interest* (*ROI*) within that *DOF* is focused. Nevertheless, the tenengrad operator can have local maximum if the *ROI* contains objects at different *DOF*s. It is also important to know that the *DOF* decreases as the magnification increases, making it harder to focus magnified objects.

4.2 Adaptive Aperture Control

The aperture controls the amount of light entering the camera, and must be set accordingly to the the scene's illumination and structure. Otherwise, the captured image might exhibit a narrow dynamic range and thus poor contrast. Two extreme manifestations of this phenomena are *overexposure* and *underexposure.* Whereas for a fixed aperture and under certain conditions, underexposure can be corrected with the use of additional light sources, this does not occur with overexposure. However, the opposite also holds true, if there is not enough light in the scene, opening the aperture will not prevent underexposure on its own unless the robot illuminates the scene. In this investigation, we use the entropy of the intensity image as a quality measure [9] and Fibonacci search to maximize the entropy. In the end, well illuminated images generally present a more evenly distributed intensity histogram.

4.3 Auto-zoom

Digital cameras capture a 2D *discrete approximation* of a 3D continuum. The spatial resolution with which an image region is captured, depends on the number

[1] We use temporal averaging and set $\tau = 0$ instead [11].

of pixels in the imaging sensor and the distance between the camera and the points in 3D space within that region. In general, low-resolution images are challenging to deal with in computer vision, and are often found when working in problems involving small structures[2] observed from a distant viewpoint. For this reason we implemented auto-zoom, which allows to acquire high-resolution images of a *ROI*. Our auto-zoom algorithm begins by centering the *ROI* in the camera frame, which is achieved through iterative correction of the camera external orientation using a *PTU*. At each step, the image of the *ROI* becomes a *template*. Then, a small pan and tilt correction towards the camera center is performed (0.1° each) and the improvement in centering is measured in vertical and horizontal direction by finding the *ROI* in a new capture using a cross-correlation based template matcher. Using the x and y displacement of the *ROI* respect its previous location and the pan and tilt angles, we estimate new pan and tilt commands. The necessity for an iterative solution arises because we assume an uncalibrated camera at this point. Later on, the zoom parameter value that would allow a capture of the highest resolution of the *ROI* can be found using Eq. 2.

$$z_{max}(z_0, R_w, R_h) = Z_v\left(\zeta M\left(z_0\right)\min\left(\frac{w}{R_w}, \frac{h}{R_h}\right)\right) \quad . \tag{2}$$

Where z_0 is the initial zoom parameter value, R_w and R_h are the width and height of the *ROI* respectively, Z_v is a polynomial model [14, 23] that maps magnification factors to zoom parameter values, M is a polynomial model that maps zoom parameter values to magnification factors, w and h are the width and height of the image respectively, and ζ is a multiplier on the resulting magnification. The optimal degree of these *cubic polynomials* was estimated using *BIC*. The regression was performed on 32 data points obtained by recording the corresponding zoom parameter value of 8 different magnification factors for 4 different calibration targets, where each calibration target was a black circle printed on a white sheet of paper. For each calibration target, the camera position and orientation was first manually set so that at zoom parameter 0, the target fits withing a square of known size overlaid in the camera images. Then, the zoom was increased so that the calibration target fits in a larger square and so on. The side length ratio of each of the squares respect to the smallest one corresponds to the magnification factor. Auto-focus was performed at each step but the focus parameter values were not used for regression.

5 Experimental Evaluation

We trained a *CoBC* of four stages, for which we first assembled training and validation image collections using images of the ICDAR train dataset [12] and hundreds of images of scenes such as parks, streets, kitchens, living-rooms, grocery products, etc. obtained from the Internet. With this, we attempt to capture

[2] Such as text written on a can.

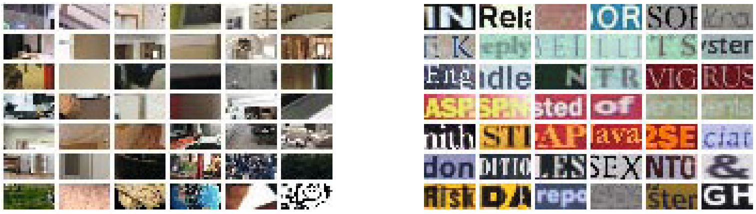


Fig. 3. Normalized training images. On the left for the negative class, and on the right for the positive class.

the high variability of the text and non-text classes. Afterwards, we generated a set of normalized training and validation images for the positive and negative classes (see Fig. 3) from which feature vectors for the training and validation examples were extracted. The normalized images have the same width and height as the detection window (24×12 pixels) and resemble the detection windows classified by the *CoBC*. The training datasets per-stage were formed of 3,378 examples of each class and the validation datasets contained 2,218 examples of the positive class and 30,000 examples for the negative class. The feature space had 7,180 dimensions, formed by 160 raw features extracted from 20 blocks, which after being turned into log-likelihoods ratios over the individual and pairwise combinations of raw features produced additional 7,056 features. In order to avoid a higher dimensionality in the data, the raw feature combinations were only performed over features of the same kind. Finally, the confidence map threshold and the threshold of the last stage classifier were optimized on a validation set of images.

The localization algorithm was evaluated on the ICDAR dataset as well as on a dataset of images of grocery products (referred as grocery images). All of the grocery images are RGB images of 640×480 pixels captured at a distance of 60 cm w.r.t to the camera. The performance was measured in terms of pixel-wise precision and recall. In general, precision is defined as $p = \frac{|C|}{|E|}$, recall as $r = \frac{|C|}{|T|}$ and their harmonic mean $h = \frac{1}{\alpha/p+(1-\alpha)/r}|\alpha = 0.5$, where C stands for the correct detections, E for all detections, and T for the target detections according to the ground truth. Since our method does not implement word grouping and the ground truth consists of bounding rectangles of each word the precision estimate will result pessimistic. Nevertheless, word grouping is very hard to realize on low-resolution images and is of minimal practical use for our application. Moreover, we compare our algorithm against the *literate_pr2*[3] package placed in the public domain and available in the *Robot Operating System* (*ROS*) repositories. The literate_pr2 package was used with OCR validation disabled. The results of the evaluation are given in Table 1. Although our algorithm performed better in both datasets, it also resulted slower than the literate_pr2 package. We attribute this drawback to the small validation datasets used to create the *CoBC*. Some localization results for our method can be seen in Fig. 4. The recall

[3] The algorithm is based on [7].

Table 1. Text localization results in terms of pixel-wise precision, recall and harmonic mean. Complementary, the average execution time per-input image is given.

Method	**Dataset**	**p**	**r**	**h**	**time(s)**
Presented method	ICDAR	0.68	0.59	0.63	6.08
	Grocery Images	0.66	0.77	0.71	1.58
literate_pr2	ICDAR	0.45	0.67	0.54	0.18
	Grocery Images	0.43	0.75	0.5361	0.03

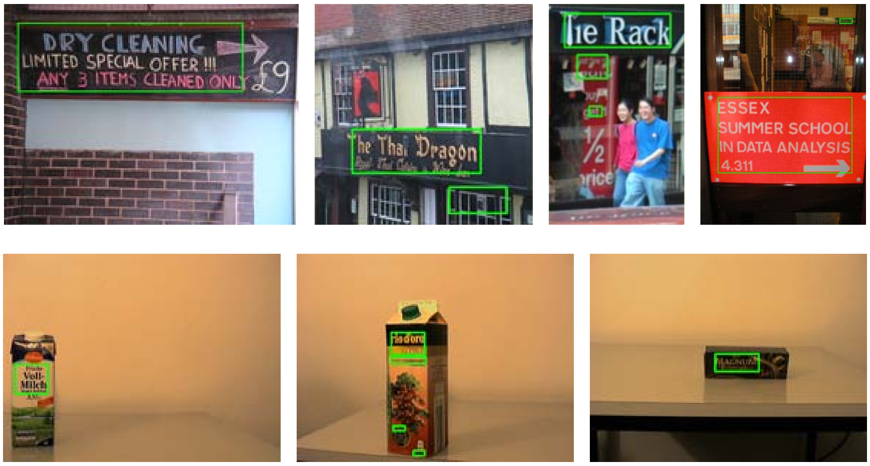


Fig. 4. Examples of the performance of our localization method on the ICDAR dataset on top and on the grocery images at the bottom.

of our method was poor in images of the ICDAR dataset in which only one or two characters occupy an entire image.

5.1 Adaptive Aperture Effect in the Localization Algorithm

To validate the usefulness of our active vision module, we devised an experiment intended to resemble one of many situations a robot can face under operation in the real-world. For this, we placed the camera at a distance of approximately 60 cm from a table in a room with normal indoor illumination. The camera sensory system was prepared as in the initialization phase described in Sect. 4. We call the resulting aperture *initial aperture value*, or *reference*. We observed that these aperture values produced well illuminated images of the scene (see Fig. 5a). Then, we turned-off the light in the room and turned-on a table lamp and made a new capture called *passive*; note that the camera was still configured with the initial aperture value (see Fig. 5b). Finally, we performed AAC to optimize the camera aperture to the new illumination conditions and made a final capture called *active* (see Fig. 5d). We repeated the same process to make a series of captures of different products on the table. Each of the images (reference, passive and active)

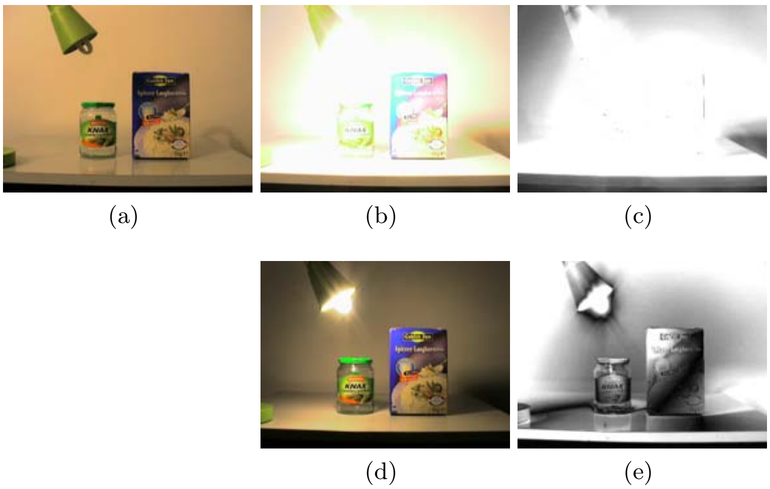


Fig. 5. AAC allows captures under different illumination conditions to be more consistent. Fig. 5a, image captured using the initial aperture, being the table lamp off. Fig. 5b, passive image captured with the initial aperture and the table lamp on. Fig. 5d, active capture with the table lamp on, after executing the AAC behavior again. Fig. 5c and Fig. 5e are the pixel-wise Euclidean distances in RGB space, between Fig. 5a and Fig. 5b, and Fig. 5a and Fig. 5d respectively. Brighter values indicate a larger Euclidean distance.

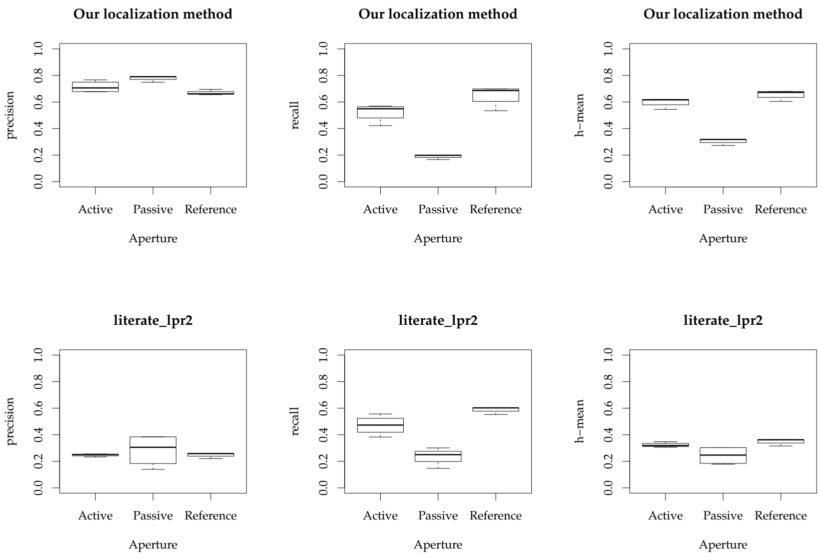


Fig. 6. A series of captures were made following the same procedure as in Fig. 5. On top, results of our algorithm. The literate_pr2 results are displayed at the bottom. From left to right: pixel-wise precision, recall and harmonic-mean. The recall of both algorithms decreases more significantly if the aperture is not adapted to the new illumination conditions.

were captured in 4 variants: one shot, one shot with Gaussian filtering, and temporal average of 2 and 5 frames. Hence, the images exhibit different degrees of noise and blur to ensure that the different results are due to illumination. The performance of the localization algorithms in these images is depicted in Fig. 6.

6 Conclusions

In this investigation we devised and evaluated an active *STR* system with text localization, auto-zoom, auto-focus and *AAC* capabilities. Our evaluation on a public dataset and on a new dataset gives evidence of the performance of our localization method. Moreover, we demonstrated how the ability to adapt to changes in the environment is crucial to the performance of *STR* systems. Since harsh acquisition conditions are often problematic in other similar tasks, we are convinced that active vision, and *active sensing* in general will play a crucial role in the development of robotics and should, whenever possible, be considered when working on difficult classification problems. In our experience, it is more effective to do so than to devise more complex passive perceptual systems. Further improvements to our system include the addition of a controllable external light source.

References

1. Álvarez Ruiz, J.A.: Learning to Discriminate Text from Synthetic Data. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 270–281. Springer, Heidelberg (2012)
2. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees, 1st edn. Chapman and Hall/CRC (January 1984)
3. Breuer, T., Giorgana Macedo, G., Hartanto, R., Hochgeschwender, N., Holz, D., Hegger, F., Jin, Z., Müller, C., Paulus, J., Reckhaus, M., Álvarez Ruiz, J.A., Plöger, P., Kraetzschmar, G.: Johnny: An autonomous service robot for domestic environments. Journal of Intelligent & Robotic Systems 66, 245–272 (2012), 10.1007/s10846-011-9608-y
4. Chen, X., Yuille, A.: Detecting and reading text in natural scenes. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, June 27-July 2, vol. 2, pp. II-366–II-373 (2004)
5. Dalal, N.: Finding people in images and videos. PhD thesis, Institut National Polytechnique de Grenoble (July 2006)
6. Dewey, J.: The reflex arc concept in psychology. Psychological Review 3(4), 357 (1896)
7. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 2963–2970 (June 2010)
8. Fraley, C., Raftery, A.E.: MCLUST version 3 for R: Normal mixture modeling and model-based clustering. Technical Report 504, University of Washington, Department of Statistic (2006) (revised 2009)
9. Huber, R., Nowak, C., Spatzek, B., Schreiber, D.: Adaptive aperture control for image enhancement. In: 2003 IEEE International Workshop on Computer Architectures for Machine Perception, pp. 7–11 (May 2003)

10. Iwatsuka, K., Yamamoto, K., Kato, K.: Development of a guide dog system for the blind people with character recognition ability. In: Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol. 1, pp. 453–456 (August 2004)
11. Krotkov, E.: Focusing. International Journal of Computer Vision 1, 223–237 (1987)
12. Lucas, S., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R., Ashida, K., Nagai, H., Okamoto, M., Yamamoto, H., et al.: ICDAR 2003 robust reading competitions: entries, results, and future directions. International Journal on Document Analysis and Recognition 7(2), 105–122 (2005)
13. Micheloni, C., Foresti, G.: Active tuning of intrinsic camera parameters. IEEE Transactions on Automation Science and Engineering 6(4), 577–587 (2009)
14. Mirmehdi, M., Clark, P.: Extracting low resolution text with an active camera for OCR. In: IX Spanish Symposium on Pattern Recognition and Image Processing, pp. 43–48 (2001)
15. Pan, Y.-F., Hou, X., Liu, C.-L.: A Robust System to Detect and Localize Texts in Natural Scene Images. In: The Eighth IAPR International Workshop on Document Analysis Systems, pp. 35–42 (September 2008)
16. Pan, Y.-F., Hou, X., Liu, C.-L.: Text Localization in Natural Scene Images Based on Conditional Random Field. In: 10th International Conference on Document Analysis and Recognition, pp. 6–10 (July 2009)
17. Posner, I., Corke, P., Newman, P.: Using text-spotting to query the world. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3181–3186 (October 2010)
18. Shiratori, H., Goto, H., Kobayashi, H.: An efficient text capture method for moving robots using dct feature and text tracking. In: International Conference on Pattern Recognition, vol. 2, pp. 1050–1053 (2006)
19. Tanaka, M., Goto, H.: Autonomous text capturing robot using improved dct feature and text tracking. In: International Conference on Document Analysis and Recognition, vol. 2, pp. 1178–1182 (2007)
20. Therneau, T., Atkinson, E.: An introduction to recursive partitioning using the RPART routines. Technical Report, Technical Report 61 (1997), `http://www.mayo.edu/hsr/techrpt/61.pdf`
21. Viola, P.: Fast and robust classification using asymmetric adaboost and a detector cascade. In: Advances in Neural Information Processing Systems (2002)
22. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2001, pp. I-511–I-518 (2001)
23. Willson, R.G.: Modeling and calibration of automated zoom lenses. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, UMI Order No. GAX94-19735 (1994)

Evaluation of Colour Models for Computer Vision Using Cluster Validation Techniques

David Budden, Shannon Fenn, Alexandre Mendes, and Stephan Chalup

School of Electrical Engineering and Computer Science
Faculty of Engineering and Built Environment
The University of Newcastle, Callaghan, NSW, 2308, Australia
`{david.budden,shannon.fenn}@uon.edu.au,`
`{alexandre.mendes,stephan.chalup}@newcastle.edu.au`

Abstract. Computer vision systems frequently employ colour segmentation as a step of feature extraction. This is particularly crucial in an environment where important features are colour-coded, such as robot soccer. This paper describes a method for determining an appropriate colour model by measuring the compactness and separation of clusters produced by the k-means algorithm. *RGB*, *HSV*, YC_bC_r and *CIE L*a*b** colour models are assessed for a selection of artificial and real images, utilising an implementation of the Dunn's-based cluster validation index. The effectiveness of the method is assessed by qualitatively comparing the relative correctness of the segmentation to the results of the cluster validation. Results demonstrate a significant variation in segmentation quality among colour spaces, and that $\mathrm{YC}_b\mathrm{C}_r$ is the best choice for the DARwIn-OP platform tested.

Keywords: Image segmentation, colour representations, colour space analysis, clustering, cluster validation, pattern recognition.

1 Introduction

In computer vision, a mapping from an arbitrary 3-component colour space C to a set of colours M assigns a class label $m_i \in M$ to every point $c_j \in C$ [18]. If each channel is represented by an n-bit value and $k = |M|$ represents the number of defined class labels, then

$$C \rightarrow M, \tag{1}$$

where

$$C = \{0, 1, \ldots, 2^n - 1\}^3 \quad \text{and} \quad M = \{m_0, m_1, \ldots, m_{k-1}\}\,. \tag{2}$$

Concretely, in a colour space C, each pixel in an image is represented by a triplet with each value representing the contribution of each component to the overall colour of that pixel. Projecting the pixel values into the colour space constructed by the orthogonal component axes results in a projected colour space distribution of the original image. Points within the projected distribution are typically clustered about centroids, which represent the predominant colours within the image. Therefore, in an image where the colour clusters are compact and well separated, a clustering algorithm such as k-means [7, 12, 20] is able to automate the process of colour segmentation. This is particularly applicable in an environment where important features are uniquely coloured, such as robot soccer [15]. Where computational resources are limited, the colour segmentation process is performed off-line, with the resultant mapping represented in the form of a $2^n \times 2^n \times 2^n$ look-up table (LUT). This LUT can then be used for efficient, real-time colour classification.

Many techniques exist for effective colour segmentation, including mean-shift and mode finding clustering [4, 7, 16, 19, 20]. Unfortunately, there are no general algorithms or colour models that are suited to all colour images [3]. Instead, this paper compares four common colour models: *RGB* [2, 3, 7, 20]; *HSV* [2, 3, 7, 13, 19, 20], YC_bC_r [3, 13, 20] and *CIE L*a*b** [3, 7, 20]) and proposes a method to assess their suitability for unsupervised colour segmentation by measuring the compactness and separation of clusters produced by the k-means algorithm. Robot soccer is chosen as a suitable colour-coded environment for testing the procedure [15], but it is anticipated that the procedure could be extended to determine the optimal colour model in other equivalent scenarios.

This paper is organised as follows: Sect. 2 describes the colour models tested; Sect. 3 describes the performance metrics used to validate the clusters obtained; Sect. 4 describes the test images; Sect. 5 presents the computational results; and finally, Sect. 6 and Sect. 7 provide a discussion and qualitative assessment of the results, followed by the conclusion.

2 Colour Models

2.1 RGB

The RGB colour model is a space in which each colour is represented by a combination of tristimuli R (red), G (green) and B (blue) [3]. Any colour can be created by exactly one combination of these three colour bases, which are defined according to their wavelength (700.0 nm for red, 546.1 nm for green and 435.8 nm for blue [20]). Although it is intuitive to represent colour in such a manner, RGB suffers from sensitivity to variations in illumination due to the high correlation of its three components [2–4, 19].

2.2 HSV

The HSV colour model separates information regarding the chrominance and intensity values of a colour by projecting the RGB colour space onto a non-linear chroma value H (hue), a radial saturation percentage S (saturation) and a luminance-inspired value V (value) [3, 7, 20]. The HSV colour model is frequently used for colour segmentation as the individual colour components are independent of the image brightness [3, 16, 19], resulting in more uniform clusters for similar chrominance values [16]. Among the disadvantages, the H component is an angular value and therefore wraps around from 2π to zero, potentially resulting in the splitting of clusters to opposite ends of the hue axis. Furthermore, the nonlinear transformation results in high susceptibility to noise for low values of V [3].

2.3 YC$_b$C$_r$

Similarly to HSV, YC$_b$C$_r$ separates chrominance information into two channels C_b (blue chroma) and C_r (red chroma), and intensity into a third channel Y (luma) [3, 20]. The YC$_b$C$_r$ colour space can be obtained applying the following linear transformation to the RGB space:

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}, \qquad (3)$$

where R', G' and B' are 8-bit gamma-compressed colour components [20]. Although much of the component correlation found in RGB is removed, it still exists in part due to the linear nature of the transformation [3].

2.4 CIE L*a*b*

The CIE (Commission International de l'Eclairage) XYZ colour system, similarly to RGB, defines three tristimulis X, Y and Z, which can be combined to create any colour. Any colour can be created by linear combination of positive quantities of each component [3, 7, 20] (unlike RGB, which requires a negative amount of red light to be added to obtain certain colours in the blue-green range [20]). The CIE XYZ colour space can be obtained by applying the following linear transformation to the RGB space [20]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \qquad (4)$$

The CIE L*a*b* (CIELAB) colour space applies nonlinear transformations to CIE XYZ to more accurately reflect the logarithmic manner in how humans perceive differences in chromaticity and luminance [3, 7, 20]. For a nominal white value (X_n, Y_n, Z_n) (chosen as the CIE D65 standard $(0.9642, 1, 0.8249)$), the L^* (lightness) component is defined as [20]

$$L^* = 116 f\left(\frac{Y}{Y_n}\right), \quad f(t) = \begin{cases} t^{1/3} & \text{, if } t > \delta^3 \\ t/(3\delta^2) + 2\delta/3 & \text{, otherwise} \end{cases} \qquad (5)$$

where $\delta = 6/29$, resulting in a value in the range $[0, 100]$. Similarly, a^* and b^* are defined as [20]

$$a^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right] \quad \text{and} \quad b^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right]. \qquad (6)$$

2.5 Linear Transformation Orthogonality

As the transformations from RGB to HSV and CIELAB are nonlinear, it is intuitive that k-means clustering within each of these spaces will yield different results for a given image. This is not inherently the case for the linear transformation from RGB to $\mathrm{YC}_b\mathrm{C}_r$, therefore it should be demonstrated that the transformation is not orthogonal (does not preserve Euclidean distance). Given a linear transformation M, it is known from linear algebra that

$$\sigma_{min}\|v\| \leq \|Mv\| \leq \sigma_{max}\|v\|, \tag{7}$$

where σ_{min} and σ_{max} are the minimum and maximum *singular values* resulting from the singular value decomposition of M [8]. For (3), these values are equal to 0.4589 and 0.8039 respectively, resulting in a condition number of $\sigma_{max}/\sigma_{min} = 1.7518 \neq 1$. Therefore it can be said with confidence that k-means will yield clusters of varying quality for every colour space described in Sect. 2.

3 Performance Metrics

The performance of a clustering algorithm within a given feature space is reflected in the quality of the resulting clusters in that space. The process of evaluating the quality of a cluster is referred to as *cluster validation*, of which three main categories exist [11]:

- *External Criteria:* Involves evaluating the clustering results based on human-defined "expected results", such as correctly identified colours within an image.
- *Internal Criteria:* Involves evaluating the clustering results in terms of the input data itself, by measuring values such as *compactness* (density of a cluster, also known as *intracluster similarity*) and *separation* (distance between clusters, also known as *intercluster dissimilarity*).
- *Relative Criteria:* Involves evaluating the clustering results in terms of its similarity to other results produced by the same algorithm, but with different parameter values.

This paper deals primarily with the application of internal criteria to produce quantitative results (see Sect. 5), with a discussion of external criteria provided in Sect. 6. Many internal criteria performance metrics have been suggested for assessing the validity of cluster partitions [9], including the *Silhouette method* [1, 17], *Dunn's based index* [1, 6, 9], *Davies-Bouldin index* [1, 5, 9] and *C-index* [1, 9, 14]. In this paper, the Dunn's index was chosen as most suitable for validating cluster partitions within the colour spaces described in Sect. 1, as it has low computational complexity and does not rely on presumptions regarding the shape or relative sizes of clusters [9].

3.1 Dunn's Index

For any partition $X = X_1 \cup \dots X_i \cup \dots X_k$, with k clusters, where X_i represents the i^{th} cluster of such a partition, the Dunn's index, D, is defined as [1, 6, 9]

$$D(X) = \min_{1 \leq i \leq k} \left\{ \min_{\substack{1 \leq j \leq k \\ j \neq i}} \left\{ \frac{\delta(X_i, X_j)}{\max_{1 \leq l \leq k} \{\Delta(X_l)\}} \right\} \right\}, \tag{8}$$

where $\delta(X_i, X_j)$ is the intercluster distance between clusters i and j; and $\Delta(X_l)$ is the intracluster distance for cluster l. The aim of this metric is maximise $D(X)$, by maximising the minimum intercluster distance $\delta(X_i, X_j)$, whilst minimising the maximum intracluster distance. As Dunn's index is calculated using only two distances it is particularly susceptible to outliers [9], therefore is is important that the distance functions $\delta(X_i, X_j)$ and $\Delta(X_l)$ be chosen in such a way as to minimise the impact of chromatic noise.

3.2 Methods of Calculating Distances

Among the possible distance metrics, *Euclidean distance* was chosen to determine the distance between two sample points, as required for the intercluster and intracluster distance calculations (discussed below). Euclidean distance was chosen for consistency with the k-means implementation utilised.

Intercluster Distances: Several methods have been suggested for calculating the intercluster distance between clusters X_i and X_j [1, 11], including:

- *Single linkage*: Shortest distance between any two objects, each belonging to separate clusters X_i and X_j.
- *Complete linkage*: Greatest distance between any two objects, each belonging to separate clusters X_i and X_j.
- *Average linkage*: Average distance between all pairs of points belonging to separate clusters X_i and X_j.
- *Centroid linkage*: Distance between the centroids of clusters X_i and X_j.
- *Average to centroids*: Average distance between the centroid of cluster X_i and all the points belonging to cluster X_j.

Of these, the average to centroids linkage was chosen as a good trade-off between outlier sensitivity and computational complexity. For two clusters X_i and X_j belonging to partition X, the average to centroids linkage $\delta(X_i, X_j)$ is defined as

$$\delta(X_i, X_j) = \frac{1}{|X_i| + |X_j|} \left(\sum_{x \in X_i} d(x, C_{X_j}) + \sum_{y \in X_j} d(y, C_{X_i}) \right), \tag{9}$$

where the cluster centroids C_{X_i} and C_{X_j} are defined as

$$C_{X_i} = \frac{1}{|X_i|} \sum_{x \in X_i} x \qquad C_{X_j} = \frac{1}{|X_j|} \sum_{y \in X_j} y. \tag{10}$$

Intracluster Distances: As with the intercluster distances, there exists several methods of calculating the intracluster distance of a cluster X_i, including [1]:

- *Complete diameter*: Greatest distance between any two points belonging to the cluster.
- *Average diameter*: Average distance between all pairs of points belonging to the cluster.
- *Centroid diameter*: Average distance between the cluster centroid and all points belonging to that cluster.

Of these, centroid diameter was chosen, again as a good trade-off between sensitivity and complexity. For a cluster X_i belonging to partition X, the centroid diameter $\varDelta(X_i)$ is defined as

$$\varDelta(X_i) = 2\left(\frac{\sum_{x \in X_i} d(x, C_{X_i})}{|X_i|}\right), \tag{11}$$

where the cluster centroid C_{X_i} is defined as in (10).

4 Test Images

This paper utilises a series of images[1], some artificially generated and some captured using the DARwIn-OP's 2MP camera [10]. Robot soccer [15] was chosen as an example of a colour-coded environment, where important features to be identified are assigned different colours. As not all features will be present in every vision frame, two images were selected to represent common soccer scenarios. The first image (see Table 1, bottom-right corner) has the camera pointed towards the ground, and contains only the field, field lines and the ball. Therefore, only three main colours are present. The second image (see Table 2, bottom-right corner) demonstrates the camera pointed toward the goal, adding a robot and goalposts to the image and therefore increasing the number of main colours to five.

To understand how the system reacts to the presence of noise in an "ideal" image, a set of artificial data was generated for each image. They consist of homogeneous blocks of colours representative of those of the image features at varying levels of Gaussian noise (10%, 20% and 30%, see Tables 1 and 2). As the feature spaces for clustering are exactly the colour spaces described in Sect. 2, the results are independent of pixel position, therefore the Gaussian noise should have an effect similar to the introduction of a variety of coloured, unimportant objects into actual vision frames. Common examples of such "noise" include a cable extended over the field or a spectator watching the match.

5 Experimental Results

The experimental results are illustrated by two groups of images, containing three and five colours respectively. k-means clustering was applied to all test

[1] Available at: `http://www.davidbudden.com/research/colour-model-evaluation/`

images described in Sect. 4 in each colour space described in Sect. 2. The range for the number of clusters for each image was chosen to be no less than the number of features identifiable in the images, therefore the values of k tested were chosen as $k = \{3, 4, \ldots, 10\}$ for the images in Table 1, and $k = \{5, 6, \ldots, 12\}$ for those in Table 2.

The Dunn's index was applied as a method of evaluating the performance of the clustering in each colour space by calculating the compactness and separation of resultant clusters, as described in Sect. 3. The process of clustering and validation for each image, colour space and k value was repeated 100 times to allow for variations in performance due to the random initial placement of the cluster centroids in the k-means algorithm. Tables 1 and 2 demonstrate the mean values over those 100 runs, with the optimal values indicated in bold.

Table 1 demonstrates that for a small number of predominant colours and a low level of chromatic noise, clusters within the RGB colour space are overall more compact and separated. Additional chromatic noise resulted in better clustering within the $\mathrm{YC_bC_r}$ colour space and for a larger k value, equal to four. It should also be noted that the standard deviation values were quite low, implying that either all 100 k-means runs converged to the same local minimum, or to several local minima with similar Dunn's index values.

Table 2 shows the results for a larger number of predominant colours. $\mathrm{YC_bC_r}$ outperforms RGB, HSV and CIELAB colour spaces in all images, irrespective of noise level. RGB also performs quite well. The standard deviation values are considerably higher than for three predominant colours; in particular, the standard deviation for $\mathrm{YC_bC_r}$ (k=5) is very high. After a thorough analysis of a repeated trial, four local optima with observed, with Dunn's index values of 0.32 (obtained 29 times), 0.31 (35 times), 0.09 (10 times) and 0.04 (26 times). Despite $\mathrm{YC_bC_r}$ appearing to be the best colour space, further tests with a larger number of real images should be conducted in the future to determine the reliability of this outcome.

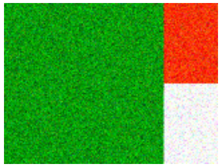
Analysing Tables 1 and 2 as a whole, it can be concluded that RGB and $\mathrm{YC_bC_r}$ demonstrate consistently better results than HSV and CIELAB, irrespective of the the image tested. A qualitative assessment of these results is presented in Section 6.

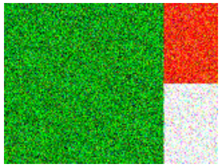
6 Discussion

The results in Sect. 5 demonstrate that for images with either more than three predominant features or greater than 20% chromatic noise, k-means produced clusters with maximum compactness and separation when performed in the $\mathrm{YC_bC_r}$ colour space. However, for this to be an accurate measure of the colour segmentation performance, it is crucial that the clusters correspond with the actual set of feature colours. In order to verify how representative the clusters found by the k-means are of the real colours in the image, a series of images are presented in Fig. 1. As an example, the image in Table 2 contains five uniquely coloured features - the green field, white lines, black robot, yellow goalposts and

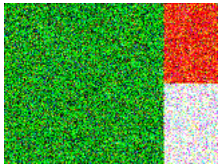
Table 1. Cluster quality evaluation using the Dunn's index for the images with three dominant colours. The four colour spaces were tested using four images: three artificially created, with Gaussian noise levels of 10%, 20% and 30%; and one original image, captured with the DARwIn-OP camera [10]. The figures represent the mean of 100 runs of the k-means algorithm; the standard deviation is indicated in parentheses. The RGB colour space demonstrated better performance for low noise levels, whereas YC_bC_r performed better for higher noise levels and the original image.

	3 colours - 10% noise								**3 colours - 20% noise**							
Colour	**Number of clusters (k)**								**Number of clusters (k)**							
space	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RGB	**3.66 (0.00)**	0.50 (0.00)	0.23 (0.11)	0.16 (0.11)	0.11 (0.08)	0.10 (0.06)	0.09 (0.02)	0.08 (0.03)	**1.01 (0.00)**	0.59 (0.00)	0.30 (0.01)	0.32 (0.02)	0.34 (0.09)	0.32 (0.09)	0.30 (0.04)	0.29 (0.01)
HSV	0.15 (0.00)	0.38 (0.06)	0.12 (0.03)	0.21 (0.01)	0.50 (0.04)	0.24 (0.02)	0.23 (0.04)	0.15 (0.08)	0.34 (0.00)	0.17 (0.01)	0.31 (0.08)	0.22 (0.03)	0.22 (0.00)	0.23 (0.00)	0.18 (0.06)	0.11 (0.02)
YC_bC_r	3.00 (0.00)	0.56 (0.00)	0.39 (0.26)	0.40 (0.10)	0.28 (0.06)	0.20 (0.10)	0.16 (0.11)	0.11 (0.09)	0.83 (0.00)	0.72 (0.00)	0.35 (0.00)	0.55 (0.00)	0.34 (0.02)	0.34 (0.07)	0.34 (0.05)	0.32 (0.03)
CIELAB	1.11 (0.00)	0.25 (0.00)	0.18 (0.03)	0.21 (0.00)	0.23 (0.03)	0.29 (0.06)	0.22 (0.09)	0.19 (0.04)	0.32 (0.00)	0.35 (0.00)	0.19 (0.00)	0.23 (0.00)	0.29 (0.00)	0.32 (0.02)	0.19 (0.02)	0.15 (0.00)


3 colours - 10% noise


3 colours - 20% noise

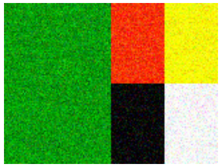
	3 colours - 30% noise								**3 colours - Original image**							
Colour	**Number of clusters (k)**								**Number of clusters (k)**							
space	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RGB	0.52 (0.00)	0.58 (0.00)	0.32 (0.00)	0.45 (0.00)	0.40 (0.05)	0.38 (0.02)	0.37 (0.02)	0.35 (0.02)	0.11 (0.00)	0.11 (0.01)	0.09 (0.00)	0.04 (0.00)	0.05 (0.01)	0.05 (0.01)	0.05 (0.01)	0.04 (0.01)
HSV	0.33 (0.00)	0.38 (0.00)	0.18 (0.01)	0.20 (0.01)	0.21 (0.01)	0.13 (0.01)	0.14 (0.03)	0.16 (0.03)	0.11 (0.00)	0.08 (0.00)	0.06 (0.00)	0.04 (0.00)	0.04 (0.00)	0.05 (0.01)	0.05 (0.00)	0.05 (0.01)
$\mathbf{YC_bC_r}$	0.47 (0.00)	**0.75 (0.00)**	0.42 (0.00)	0.53 (0.00)	0.30 (0.01)	0.32 (0.02)	0.35 (0.02)	0.41 (0.03)	0.11 (0.00)	**0.12 (0.00)**	0.12 (0.01)	0.05 (0.01)	0.05 (0.01)	0.05 (0.01)	0.04 (0.01)	0.04 (0.01)
CIELAB	0.26 (0.00)	0.23 (0.01)	0.24 (0.00)	0.18 (0.00)	0.18 (0.01)	0.17 (0.01)	0.19 (0.00)	0.16 (0.03)	0.10 (0.00)	0.11 (0.00)	0.07 (0.00)	0.07 (0.02)	0.07 (0.02)	0.04 (0.01)	0.04 (0.01)	0.03 (0.00)

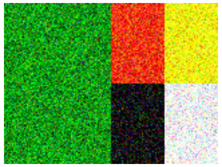

3 colours - 30% noise

3 colours - Original

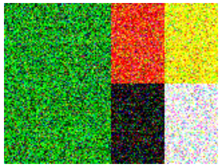
Table 2. Cluster quality evaluation using the Dunn's index for images with five dominant colours. The YC_bC_r colour space demonstrated the best performance, irrespective of the noise level. In addition, the optimal Dunn's index value was obtained with k = 5 on all four images. It should be noted that RGB also maintained a good overall performance.

Colour space	5 colours - 10% noise: Number of clusters (k) 5	6	7	8	9	10	11	12	5 colours - 20% noise: Number of clusters (k) 5	6	7	8	9	10	11	12
RGB	1.86 (0.00)	0.53 (0.00)	0.74 (0.24)	0.32 (0.10)	0.29 (0.07)	0.27 (0.06)	0.26 (0.05)	0.26 (0.03)	0.96 (0.00)	0.53 (0.01)	0.33 (0.03)	0.33 (0.02)	0.33 (0.07)	0.30 (0.05)	0.29 (0.04)	0.28 (0.03)
HSV	0.21 (0.00)	0.21 (0.01)	0.45 (0.08)	0.13 (0.02)	0.20 (0.06)	0.17 (0.08)	0.10 (0.06)	0.06 (0.02)	0.28 (0.04)	0.24 (0.02)	0.19 (0.00)	0.23 (0.03)	0.22 (0.03)	0.22 (0.03)	0.16 (0.05)	0.13 (0.04)
YC_bC_r	**2.23 (0.00)**	0.60 (0.00)	1.07 (0.25)	0.48 (0.15)	0.33 (0.08)	0.29 (0.05)	0.29 (0.05)	0.27 (0.05)	**1.22 (0.00)**	0.63 (0.00)	0.43 (0.14)	0.50 (0.07)	0.33 (0.01)	0.37 (0.02)	0.38 (0.04)	0.33 (0.06)
CIELAB	0.24 (0.03)	0.33 (0.02)	0.26 (0.16)	0.17 (0.06)	0.18 (0.04)	0.19 (0.03)	0.18 (0.04)	0.19 (0.05)	0.20 (0.00)	0.26 (0.00)	0.24 (0.10)	0.21 (0.05)	0.22 (0.05)	0.27 (0.00)	0.27 (0.02)	0.30 (0.01)


5 colours - 10% noise


5 colours - 20% noise

Colour space	5 colours - 30% noise: Number of clusters (k) 5	6	7	8	9	10	11	12	5 colours - Original image: Number of clusters (k) 5	6	7	8	9	10	11	12
RGB	0.97 (0.00)	0.47 (0.07)	0.44 (0.03)	0.41 (0.04)	0.40 (0.04)	0.38 (0.06)	0.37 (0.05)	0.36 (0.05)	0.22 (0.04)	0.10 (0.04)	0.07 (0.02)	0.06 (0.01)	0.06 (0.01)	0.07 (0.02)	0.07 (0.03)	0.07 (0.03)
HSV	0.30 (0.00)	0.24 (0.02)	0.26 (0.03)	0.38 (0.10)	0.32 (0.04)	0.22 (0.01)	0.21 (0.01)	0.20 (0.01)	0.19 (0.00)	0.12 (0.01)	0.12 (0.01)	0.13 (0.04)	0.07 (0.02)	0.06 (0.02)	0.06 (0.03)	0.05 (0.03)
YC_bC_r	**0.99 (0.00)**	0.67 (0.00)	0.84 (0.08)	0.43 (0.08)	0.42 (0.05)	0.41 (0.04)	0.40 (0.04)	0.40 (0.04)	**0.25 (0.12)**	0.06 (0.00)	0.06 (0.01)	0.06 (0.02)	0.07 (0.02)	0.07 (0.03)	0.08 (0.03)	0.07 (0.03)
CIELAB	0.34 (0.00)	0.29 (0.04)	0.32 (0.00)	0.33 (0.00)	0.31 (0.02)	0.38 (0.04)	0.44 (0.02)	0.36 (0.03)	0.07 (0.02)	0.06 (0.02)	0.05 (0.01)	0.05 (0.01)	0.05 (0.02)	0.05 (0.01)	0.05 (0.02)	0.06 (0.03)


5 colours - 30% noise

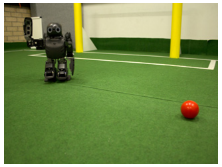
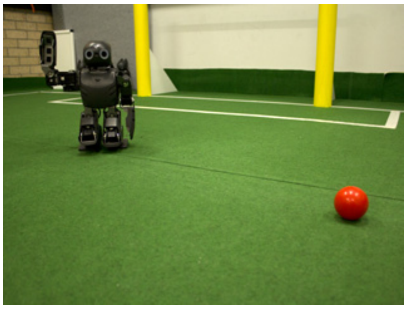
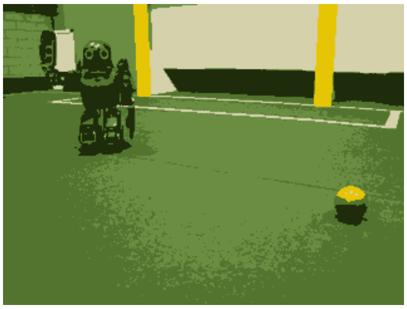
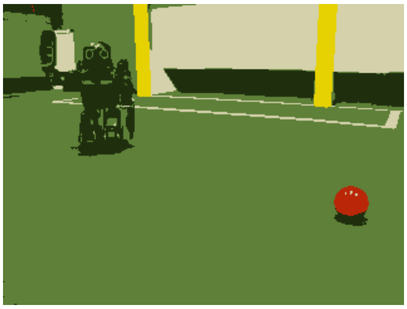
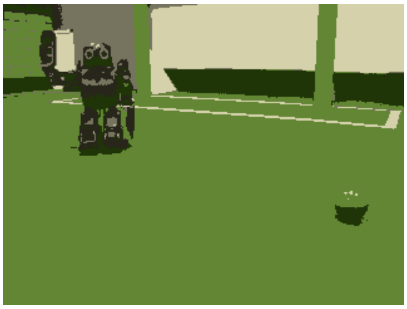

5 colours - Original

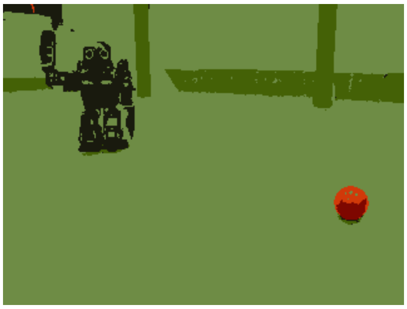
Fig. 1. Original image mapped to the corresponding five clusters for each colour space. (a) Original image; (b) 5-colour mapping using the RGB colour space; (c) 5-colour mapping using YC_bC_r; (d) 5-colour mapping using HSV; (e) 5-colour mapping using CIELAB. For (b-e), the five colours represent the centroid of their respective clusters. Note the best performance for the YC_bC_r space, which successfully captures red, white, yellow and light green, with the black colour represented as a darker shade of green. Note also the presence of some artifacts, such as the ball being mapped into three colours: yellow, light green and dark green for RGB; and two shades of red for CIELAB.

(a)


(b)


(c)


(d)


(e)


red ball. A correct segmentation should therefore map the pixels comprising each of these features to a unique colour label.

The pixels in the images in Fig. 1b-e were painted with the centroid values of the clusters that they were assigned to. However, given that 100 runs of the k-means algorithm were executed for each configuration of image and colour space, it is vital to determine with set of centroids was most representative of the overall results. After analysing the results for a given configuration, it was determined that the number of different solutions was never greater than five - i.e. the same sets of centroids were reached several times in those 100 runs. With that in mind, the centroid values used in each configuration depicted in Fig. 1b-e are the solutions that occurred most frequently.

Table 2 suggests the YC_bC_r colour space allows better clustering performance than the other three colour spaces for Fig. 1a. That assertion is confirmed by qualitatively assessing the result of applying k-means determine the main colours in the image (for $k = 5$). Fig. 1b shows that clustering in RGB failed to correctly label the red ball, which contains pixels that belong to three different clusters (yellow and two shades of green). In addition, it associated the field and the robot with three separate clusters, at different levels of green. In contrast, clustering in YC_bC_r (Fig. 1c) labeled each feature correctly. Figure 1d depicts the result for HSV; it fails to identify the goal post, assigning to it the same colour of the field. In addition, the ball contains two shades of green. Finally, Figure 1e depicts the result for CIELAB. It identifies the ball and the robot, but fails to identify the ball or field lines. This agreement between external and internal cluster validation criteria supports cluster validation as an effective method for assessing the appropriateness of a colour model for segmentation.

7 Conclusion

This work analysed the use of different colour spaces in the quality of colour classification in a computer vision system. The application of a Dunn's index cluster validation technique demonstrated that the k-means algorithm produced clusters of maximum compactness and separation in the YC_bC_r colour space, for images with five uniquely coloured features. RGB, HSV and CIELAB yielded poorer results independent of the level of chromatic noise present. For images with three uniquely coloured features, YC_bC_r yielded the best results for higher levels of chromatic noise, whereas RGB had a better performance for low levels of noise.

The quantitative results were also qualitatively assessed by visualising the colour segmentation for each colour model. In that case, the qualitative analysis of the YC_bC_r results confirmed the numerical results. Correct segmentation corresponded with higher values of the Dunn's-based index, therefore supporting cluster validation as an effective technique of assessing the performance of a colour model for segmentation within a colour-coded environment.

Future research will focus on automating the LUT generation process for DARwIn-OP platform, allowing for efficient real-time colour image segmentation whilst minimising the requirement for human supervision.

Acknowledgement. David Budden and Shannon Fenn would like to thank the University of Newcastle's Faculty of Engineering and Built Environment for the support provided by their Summer Research Scholarship program.

References

1. Bolshakova, N., Azuaje, F.: Cluster validation techniques for genome expression data. Signal Processing 83(4), 825–833 (2003)
2. Brusey, J., Padgham, L.: Techniques for obtaining robust, real-time, colour-based vision for robotics. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup 1999. LNCS (LNAI), vol. 1856, pp. 243–253. Springer, Heidelberg (2000)

3. Cheng, H., Jiang, X., Sun, Y., Wang, J.: Color image segmentation: advances and prospects. Pattern Recognition 34(12), 2259–2281 (2001)
4. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5), 603–619 (2002)
5. Davies, D., Bouldin, D.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence (2), 224–227 (1979)
6. Dunn, J.: Well-separated clusters and optimal fuzzy partitions. Journal of Cybernetics 4(1), 95–104 (1974)
7. Forsyth, D., Ponce, J.: Computer vision: a modern approach. Prentice Hall (2002)
8. Golub, G., Van Loan, C.: Matrix computations, vol. 3. Johns Hopkins University Press (1996)
9. Günter, S., Bunke, H.: Validation indices for graph clustering. Pattern Recognition Letters 24(8), 1107–1113 (2003)
10. Ha, I., Tamura, Y., Asama, H., Han, J., Hong, D.: Development of open humanoid platform DARwIn-OP. In: 2011 Proceedings of the SICE Annual Conference (SICE), pp. 2178–2181. IEEE (2011)
11. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2), 107–145 (2001)
12. Hartigan, J., Wong, M.: Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28(1), 100–108 (1979)
13. Henderson, N., King, R., Chalup, S.: An automated colour calibration system using multivariate gaussian mixtures to segment HSI colour space. In: Proceedings of the 2008 Australasian Conference on Robotics & Automation (ACRA 2008) (2008)
14. Hubert, L., Schultz, J.: Quadratic assignment as a general data analysis strategy. British Journal of Mathematical and Statistical Psychology 29(2), 190–241 (1976)
15. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: The robot world cup initiative. In: Proceedings of the First International Conference on Autonomous Agents, pp. 340–347. ACM (1997)
16. Park, J., Lee, G., Park, S.: Color image segmentation using adaptive mean shift and statistical model-based methods. Computers & Mathematics with Applications 57(6), 970–980 (2009)
17. Rousseeuw, P.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, 53–65 (1987)
18. Sridharan, M., Stone, P.: Real-time vision on a mobile robot platform. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 2148–2153. IEEE (2005)
19. Sural, S., Qian, G., Pramanik, S.: Segmentation and histogram generation using the HSV color space for image retrieval. In: Proceedings of the 2002 International Conference on Image Processing, vol. 2, pp. II–589. IEEE (2002)
20. Szeliski, R.: Computer vision: algorithms and applications. Springer-Verlag New York Inc. (2010)

Using Saliency-Based Visual Attention Methods for Achieving Illumination Invariance in Robot Soccer

F. Serhan Daniş, Tekin Meriçli, and H. Levent Akın

Department of Computer Engineering
Boğaziçi University
Istanbul, Turkey
{serhan.danis,tekin.mericli,akin}@boun.edu.tr

Abstract. In order to be able to beat the world champion human soccer team in the year 2050, soccer playing robots will need to have very robust vision systems that can cope with drastic changes in illumination conditions. However, the current vision systems are still brittle and they require exhaustive and repeated color calibration procedures to perform acceptably well. In this paper, we investigate the suitability of biologically inspired saliency-based visual attention models for developing robust vision systems for soccer playing robots while focusing on the illumination invariance aspect of the solution. The experiment results demonstrate successful and accurate detection of the ball even when the illumination conditions change continuously and dramatically.

1 Introduction

As the deadline for achieving the ultimate goal of RoboCup approaches, where a team of autonomous humanoid robots is expected to play soccer on a standard soccer field against the most recent winner of the World Cup and win the game, robustness to changing visual circumstances remains one of the biggest challenges for developing reliable vision systems for autonomous robots. Using color segmentation as the basis of the developed vision systems still appears to be the most popular approach among the teams of various RoboCup leagues although most of the color segmentation based techniques are not robust against changing illumination conditions. Since the successful operation of the robots primarily depends on the reliability of the vision system, the teams spend a considerable amount of time for color calibration even though the games are still played under carefully controlled illumination conditions.

Biological systems, on the other hand, are very successful in solving such problems; therefore, they have long been the primary source of inspiration for robot vision researchers. The visual attention mechanism is one of the most studied sub-systems of biological vision. Following the visual attention phenomena, researchers aim to obtain more efficient, intelligent, and robust artificial vision systems [1]. In alignment with this goal, in this paper, we present the results of a primary investigation on the suitability of saliency-based visual attention models for the robot soccer domain, focusing on the object detection performance under continuously and drastically changing illumination conditions. Our experiments demonstrate successful detection of the ball even under extreme changes in the illumination conditions where color segmentation based approaches fail to do so.

The rest of this paper is organized as follows. Section 2 gives an overview of the related work in the literature. The methodology followed for this contribution is explained in Section 3, and the details of the experiments and the obtained results are given in Section 4. Section 5 summarizes and concludes the paper while pointing out to potential future work.

2 Related Work

One of the biggest challenges for autonomous mobile robots that perceive their environments through standard cameras is the changing illumination conditions. As a workaround, either restricted configurations in structured domains are considered, or specific models of segmentation and recognition that do not provide generalized solutions [2] are used. Various approaches to address this challenge include describing the problem in terms of illumination [3], surface reflectance [4], and sensor sensitivity [5]. Bayesian decision theory and hierarchical model based approaches also exist in the literature [6, 7]. Methods that do not require domain specific tuning are shown to be computationally less complex and more adaptive, whereas usually the opposite is shown to be true for the classical and model based methods [2]. Illumination invariance has been studied in the robot soccer domain as well since the overall performance of the teams heavily depend on the successful perception of the environment [7, 8].

Visual attention-based approaches are usually used for preprocessing the visual sensory data to determine the parts of the image to further process. For instance, in the work of Rasolzadeh *et al* [9], the visual attention module gets executed prior to the object detection and recognition modules to direct the head saccades and help the robot figure out where to search for important objects. Frintrop *et al* proposed a similar approach [10, 11], where the regions of interest are detected using both bottom up and top down saliency extraction followed by a fast-classifier that classifies regions for object recognition purposes. They applied this method to the problem of detecting balls in a robot soccer environment, and showed that this approach yields to a faster execution compared to a standard classifier and reduces the false detection rates significantly; however, they did not investigate the problem of changing lighting conditions thoroughly.

In a similar setup that we present in this paper, Garcia *et al* [12] used an attention mechanism to detect balls in the "any ball challenge" scenario of the RoboCup Standard Platform League (SPL), where balls of various sizes, textures, and colors are scattered over the field and the robot is expected to detect them and score by kicking them into the opponent goal. They used the saliency map to extract balls on the field as the regions of the images containing the balls popped out as the "salient regions" compared to the plain green field carpet. Their work differs from the original method of saliency map generation by Itti *et al* [13] in two aspects. First, they use only the color and intensity information and discard the other channels such as motion, orientation, and flicker. Second, the sizes of the images are reduced using the fovea mask for computational efficiency purposes. They present a model performance improvement and a neat integration of the method to the SPL domain; however, they do not consider the changing illumination conditions as we investigate in detail in this paper.

3 Methodology

In this section, we present the working principles of both the saliency-based visual attention and color segmentation and scanline based object detection methods as the comparison of these two approaches constitute the main motivation behind this work.

3.1 Saliency-Based Visual Attention

Although primates have neuronal hardware with limited speed, they are capable of interpreting complex scenes in real time. Such capability is believed to be achieved by the selection of a subset of available visual information by higher visual areas before further processing [14]. Inspired by the remarkable scene interpretation ability of primates and building on a biologically plausible architecture presented by Koch and Ullman [15], which explains human visual search strategies [16], Itti *et al* proposed a model of attention that is based on the same working principles [13]. In this computational model of saliency-based visual attention, the visual input (i.e. the image) is decomposed into feature maps; primarily color, intensity, and orientation, which compete for the final saliency map. During the saliency map generation process, the input image is first used to generate color, intensity, and orientation layers. These different layers are used to generate multi-scale Gaussian pyramids, which correspond to progressive low-pass filtering and sub-sampling into lower resolutions. The feature maps are then obtained by a set of linear center-surround operations and across-scale combinations of these multi-scale pyramids. Finally, these feature maps are linearly combined to generate the final saliency map [17–19]. This model is depicted in Fig. 1.

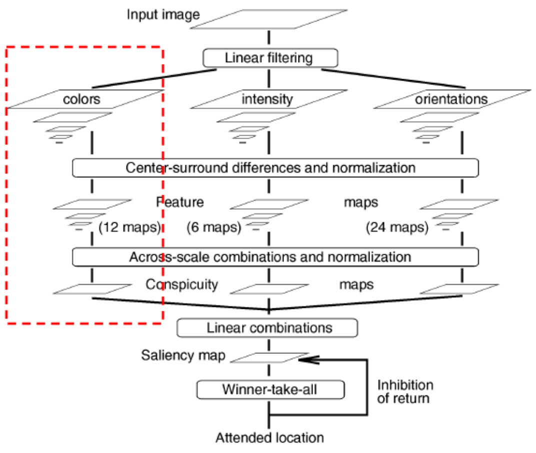


Fig. 1. The attention model proposed by Itti *et al* [13]

Based on the attention model proposed by Itti *et al* [13], in this work, we use the color conspicuity map (marked with the red frame in Fig. 1) to investigate whether it is possible to detect important objects in the image even when the illumination conditions of the environment fluctuate drastically. We also employ and test the performance of a slightly modified version of this model, which uses wavelet low-pass pyramids instead of Gaussian ones [20]. The steps of the saliency map generation process are explained in detail in the following sections.

Generation of Color Conspicuity Maps. The first step is to obtain the intensity image I by averaging the R, G, and B color channels, which we denote as matrices of the same size as the original image. Pixel values that are smaller than $1/10$ of the maximum intensity are set to zero before further processing as it gets difficult to perceive the color information of a pixel when its intensity value is very small. New color components, RN, GN, BN, and YN are then computed in terms of R, G, and B as follows.

$$RN = R - (G+B)/2 \tag{1}$$
$$GB = G - (R+B)/2 \tag{2}$$
$$BN = B - (R+G)/2 \tag{3}$$
$$YN = G+R-|R-G|-B \tag{4}$$

Negative values of these color components are set to zero and the component pyramids are constructed as RN_k, GN_k, BN_k and YN_k by either using the Gaussian low pass filter and progressively down-scaling the image into its half size [13] or using the wavelet low pass filter [20]. The subscript k denotes the level of the pyramid, where level 0 is the top level color component of the size of the original image.

Obtaining the Feature Maps. Feature maps are computed using a method that is inspired by the working principles of the "color double-opponent cells" that were proven to exist in the human primary visual cortex. These neurons are excited by one color in the center of their receptive fields, and inhibited by another, while the opposite is true in the surround. Human primary visual cortex is shown to have such spatial and chromatic opponency for the red/green, green/red, blue/yellow, and yellow/blue color pairs [21]. Analogously, we compute the center surround differences to obtain the feature maps, where $c \in \{2,3,4\}$ are centers and $s = c + p$ are their surrounds with $p \in \{3,4\}$, and * denotes the pyramid levels that are resized to a finer resolution, which in this work is determined by the finest available resolution in the data.

$$RG_{c,s} = |(RN_c - GN_c) - (RN_s^* - GN_s^*)| \tag{5}$$
$$BY_{c,s} = |(BN_c - YN_c) - (BN_s^* - YN_s^*)| \tag{6}$$

While the across-scale combination step of the conspicuity map generation process in the model of Itti *et al* [13] is performed by integrating all color feature maps at different scales, the color feature maps in the work of Li *et al* [20] are first resized to the size of the original image and then squared, resulting in few redundant salient areas. At each step of the algorithm, normalizations should also be applied on the feature maps in order to eliminate modality-dependent amplitude differences. These two proposed methods are identical aside from the functions used for the generation of the color component pyramids, namely wavelet transform and Gaussian filter, and the additional square operation used in the method proposed by Li *et al.*

Merging into a Saliency Map. In our experiments, we used five different saliency maps; three of them are generated using the method proposed by Itti *et al* [13] (*M1)*, and the other two are obtained using the method proposed by Li *et al* [20] (*M2)*.

Table 1. Listing of saliency maps

Map	Description
M1-a	Color conspicuity map from (*M1)*
M1-b	Combination of the color and intensity conspicuity maps
M1-c	Only the red-green channel of the color conspicuity map
M2-a	Color conspicuity map from (*M2)*
M2-b	Squared *M2-a*

The first saliency map corresponds to the color conspicuity map from *M1.* The second one is the saliency map generated by equally combining the color conspicuity map with the intensity map. Although Garcia *et al* [12] used the two conspicuity maps to generate the final saliency map for detecting the balls on the field, we anticipated that color feature channels would give better results for objects of specific colors. On the basis of this anticipation, we used a third map that is obtained by only utilizing the red-green (*RG*) channel, which made sense considering that our task is finding a red-orange ball on a green field. The fourth map is obtained directly through *M2,* and the fifth one is generated by taking the squares of the pixel values of the fourth map, with the expectation of reducing the number of redundant salient areas as reported by Li *et al* [20]. Table 1 summarizes the compounds of the generated maps.

3.2 Color Segmentation and Scanline Based Object Detection

Considering computational efficiency and real-time constraints of the robot soccer domain, it is not feasible to process each pixel of the image to find the objects of interests. Therefore, scanlines are used to process the image in a sparse manner, hence speeding up the entire process. This method is especially popular within the RoboCup SPL community as a commercially available standard robot platform with very limited computational resources need to be used for the competitions. A previously trained and stored color table (*CT*) is utilized for checking the colors of the pixels that a scanline runs through. We utilize a Generalized Regression Neural Network (GRNN) [22] for mapping the real color space to the pseudo-color space composed of a smaller set

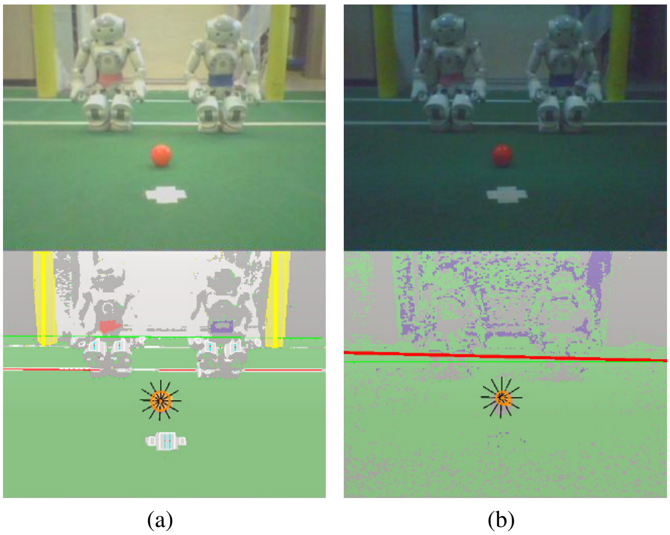


Fig. 2. Color segmentation and scanline based object detection. The classified image looks very clear and the important objects in the image are successfully detected when a suitable color table is used (a); however, this approach becomes unreliable when the lighting conditions change (b).

of pseudo-colors, namely, white, green, yellow, blue, robot-blue, orange, red, and "ignore". In order to obtain the outputs of the trained GRNN in a time-efficient manner, a look up table is constructed for all possible inputs. Y, U, and V values are used to calculate the unique index and the value at that index gives the color group ID to determine the color group of a pixel.

Regions of interest are formed by grouping the same-colored scanline segments that are spatially adjacent and "touching" each other; that is, these segments are on two consecutive scanlines and either of them has a start or end point within the borders of the other one. These regions are then passed to the so called the *region analyzer* module to be further filtered and processed for the detection of the ball, the field lines and intersections of them, the goal posts, and the robots. The ball detector uses additional star-shaped scanlines that originate from the centers of the candidate regions to find the borders of the region and use these border points to check whether the region has circular properties by using a voting-based circle fitting algorithm. Fig. 2(a) shows the result of this process when the used color table matches the lighting conditions; however, this approach may fail when the lighting conditions change as shown in Fig. 2(b), which constituted the main motivation behind this research. The objects are either not detected at all (e.g. lines and goal posts), or the detection result is misleading (e.g. smaller-than-actual size of the ball, which results in a farther-than-actual projected ball location on the field).

4 Experiments

For our experiments, we utilize the iLab Neuromorphic Vision C++ Toolkit (iNVT) software developed and released by Itti *et al* [23]. The experiments are run on the grayscale saliency maps extracted via the methods mentioned in Section 3 as well as the raw images processed by our color classification and scanline based vision module that utilizes previously trained color tables, which is explained in Section 3.2. We particularly focus on the detection of the ball in the images.

4.1 The Robot Platform

We performed our initial experiments offline on the images captured from one of the cameras of the *Nao V3* humanoid robot manufactured by Aldebaran Robotics [24], which has been used as the common robot platform of the Standard Platform League (SPL) of RoboCup [25] since 2008. The *Nao*'s camera is capable of providing images with 640×480 resolution at 30Hz; however, most teams prefer using 320×240 images due to the processing power limitations of the robot. Our team also utilizes the images provided in 320×240 resolution for the competitions; therefore, in order to be able to compare the performances of the saliency-based methods with the performance of the color segmentation based method, we kept the image resolutions identical for the two methods in our experiments.

4.2 Illumination Configurations

In our experiments, we use 6 different illumination configurations controlled by 3 factors. We denote these factors as *fluo* for the fluorescent lamps, *spot* for the spot lights, and *day* for the daylight coming in through the windows. The combinations of these configurations are labeled as $\{C_1, ..., C_6\}$. Table 2 summarizes these configurations. Even though there are 8 possible combinations of these 3 factors, in our experiments, we exclude the $\langle \neg fluo, spot, \neg dayl \rangle$ and $\langle \neg fluo, spot, dayl \rangle$ configurations as the presence of the spot lights provides the majority of the lux value, which is covered by the cases C_5 and C_6.

Table 2. Listing of the illumination configuration used in our experiments

Configuration	Tuple	Description	Illuminance
C_1	$\langle \neg fluo, \neg spot, \neg dayl \rangle$	no lights	39 lux
C_2	$\langle \neg fluo, \neg spot, dayl \rangle$	only daylight	134 lux
C_3	$\langle fluo, \neg spot, \neg dayl \rangle$	only fluorescent lights	200 lux
C_4	$\langle fluo, \neg spot, dayl \rangle$	fluorescent lights and daylight	350 lux
C_5	$\langle fluo, spot, \neg dayl \rangle$	fluorescent and spot lights	908 lux
C_6	$\langle fluo, spot, dayl \rangle$	all lights	1067 lux

The scene we set up for our experiments includes the essential visual elements of the SPL; which are one red and one blue player placed in front of the yellow goal, the field lines and the cross-shaped penalty mark, and the ball placed between the robots and the penalty mark. During our experiments, we keep the robot stationary while the contributions of the different light sources to the environment's illumination characteristics are changed to generate different lighting conditions. The sample color images from these configurations are given in Fig. 3 with the corresponding average lux values of the environment.

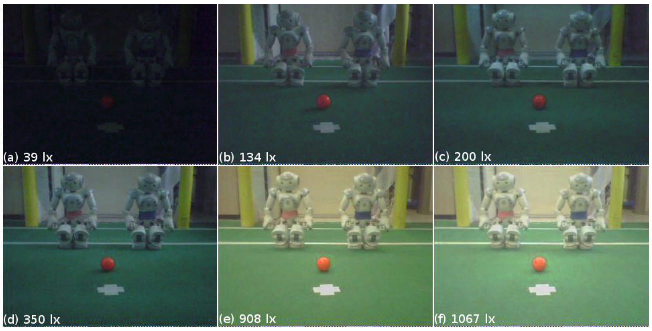
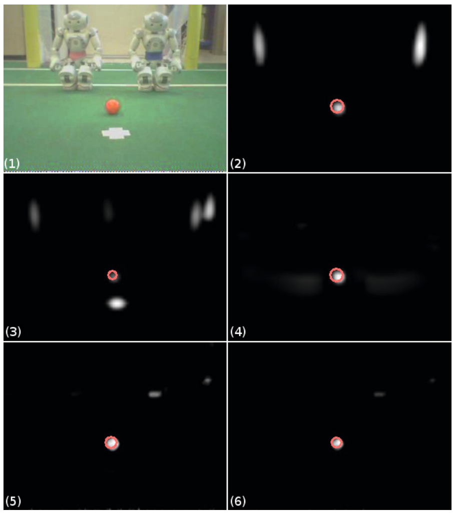


Fig. 3. Scenes captured by the robot under different illumination configurations: (a) C_1 - no lights, (b) C_2 - only daylight, (c) C_3 - only fluorescent lights, (d) C_4 - fluorescent lights and daylight, (e) C_5 - fluorescent and spot lights, and (f) C_6 - all lights

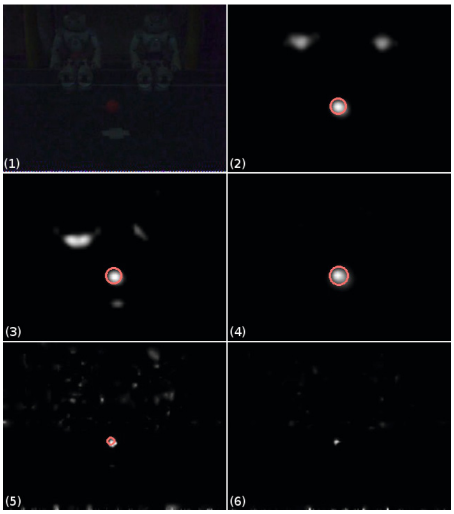
4.3 Results

We used the five different saliency map generation methods listed in Table 1 in our saliency-based experiments. Sample saliency maps for configurations C_1 and C_6 are shown in Fig. 4. Ball detection is performed by trying to fit a circle to the salient regions after performing a simple thresholding on them. The circle fit operation is considered successful if the error value is lower than 10 pixels. Detected balls on the saliency maps are also shown as orange circles in Fig. 4. Additionally, an error analysis is performed by utilizing human perception as the source of the ground truth information and reporting the difference between the ground truth and the output of the detection algorithm. For each image, we report a *hit* when the actual ball is found accurately (center and radius errors are below 8 pixels), a *false location* when some other regions is confused for the ball, and a *miss* when the ball is not detected at all. The left column and the right column of Fig. 5 show the *hit*, *false location*, and *miss* rates obtained after running the color segmentation based and saliency based algorithms, respectively, on 80 frames captured under each of the six different illumination configurations.

It can be interpreted from Fig. 5(a), 5(c), and 5(e) that the color segmentation based methods work quite well when the color table used matches the illumination condition; however, usually even small changes in the illumination characteristics results in failure, as also shown in Fig. 2. Although it is possible to prepare several color tables for various illumination conditions and switch between them based on some image statistics, this method becomes ineffective for continuously changing illumination conditions.



(a) Configuration $C_1 = \langle fluo, spot, dayl \rangle$: (1) original frame, saliency methods (2) I-a, (3) I-b, (4) I-c, (5) II-a, and (6) II-b.



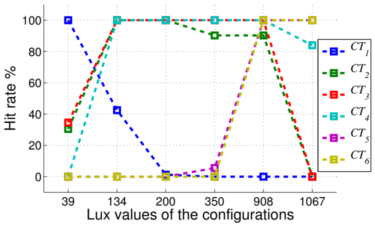
(b) Configuration $C_6 = \langle \neg flue, \neg spot, \neg dayl \rangle$: (1) original frame, saliency methods (2) I-a, (3) I-b, (4) I-c, (5) II-a, and (6) II-b.

Fig. 4. Sample saliency maps

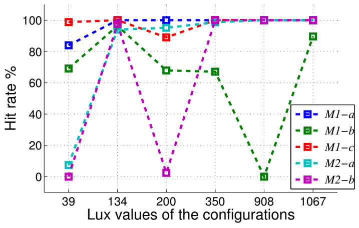
In Fig. 5(b), 5(d), and 5(f), we see that using only the color channels yields better results in general. Using the intensity map combined with the color map, which corresponds to *M1-b*, results in low hit rates and high false location rates. The results obtained with *M2-a* and *M2-b* show that *M2* works well for object recognition in bright environments; however, it performs poorly when there is not enough light. The highest accuracy in finding the ball is achieved with *M1-c*.

Fig. 6 shows the mean errors between the radius reported by the color segmentation based and the saliency based methods and the radius marked by a human as the ground truth. Considering that Fig. 6(a) shows the consistency of only the rare occasions that the ball is detected, saliency based methods tend to be more consistent whereas the color segmentation based method reports inconsistent results especially for the lighting configurations that the used color table is not trained for. The most consistent results seem to be achieved with *M1-c*. In addition to the reported radius consistency analysis, we also performed a consistency analysis for the reported center of the ball, the results of which can be seen in Fig. 7. The color segmentation based method reports a consistent center for the detected ball when a ball is found in the image; however, the biggest problem with this method is that it usually cannot find the ball at all when there is a mismatch between the color table and the illumination configuration (Fig. 5(e)).

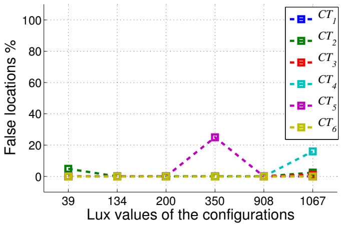
Table 3 summarizes the success rates obtained when the most successful saliency based methods *M1-a* and *M1-c*, and the color segmentation based method running with the available color tables are applied on a dataset of 618 frames collected under continuously changing illumination conditions. Perfect and near perfect hit rates are achieved with *M1-a* and *M1-c*, respectively.



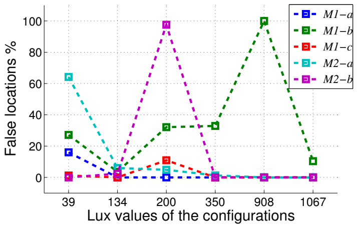
(a) Segmentation based hit rate



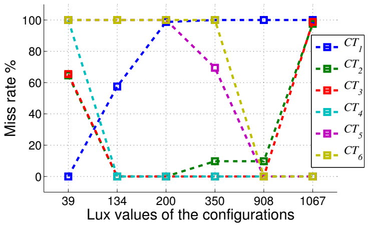
(b) Saliency based hit rate



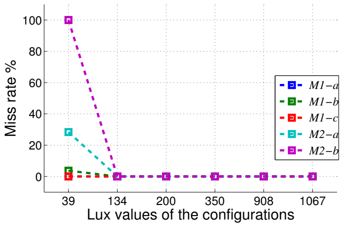
(c) Segmentation based false location rate



(d) Saliency based false location rate

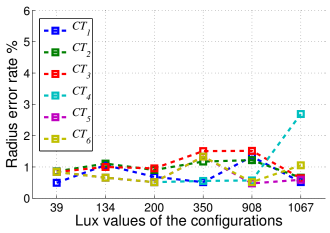


(e) Segmentation based missed ball rate

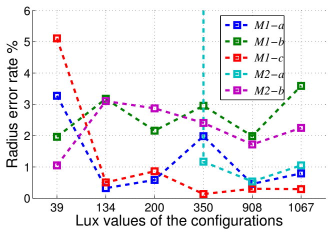


(f) Saliency based missed ball rate

Fig. 5. Performances of the color segmentation based (left) and saliency based methods (right)

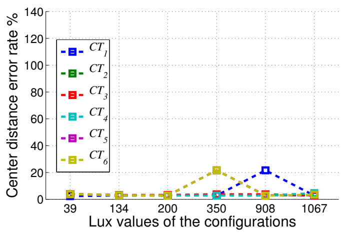


(a) Color segmentation based

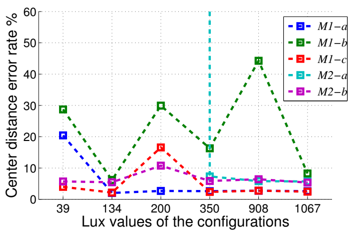


(b) Saliency based

Fig. 6. Radial consistency check for all methods



(a) Color segmentation based



(b) Saliency based

Fig. 7. Central consistency check for all methods

Table 3. Comparison of the ball detection performances (%) of the individual color tables (CT_i) and *M1-c* on a dataset collected under continuously changing illumination conditions

	CT_1	CT_2	CT_3	CT_4	CT_5	CT_6	*M1-a*	*M1-c*
hit rate	13.13	10.37	89.14	90.76	8.91	2.59	**100**	**99.68**
false locations	18.48	0.49	0.16	9.24	1.94	0	0	0.32
miss rate	68.39	89.14	10.69	0	89.14	97.40	0	0

5 Conclusions and Future Work

Illumination independent robust visual perception of the environment has been one of the biggest challenges for computer and robot vision researchers. Being capable of solving this problem almost effortlessly, biological systems have been a great source of inspiration for the proposed solutions thus far. In this paper, we make use of one such biologically inspired saliency based method with some modifications to investigate its suitability for illumination independent object detection in robot soccer domain. Our experiments demonstrate successful and consistent detection of the ball even when the lighting conditions of the environment change drastically, while the standard color classification based methods fail in such cases. Even though the experiments were performed on an off-board computer as the processor of the available *Nao* robot platform cannot meet the real-time requirements when executing the saliency based method, this approach can still be applicable in other leagues of RoboCup, such as the Middle Size League, where robots equipped with more powerful computational resources are used. Potential future work includes the development of a computationally efficient version of this method for achieving real-time on-board computations, a complete object detection framework with additional sanity checks and filters, and testing of the method on a moving robot in a regular robot soccer game.

Acknowledgments. This project was supported by Boğaziçi University Research Fund project 09M105.

References

1. Frintrop, S., Rome, E., Christensen, H.I.: Computational Visual Attention Systems and Their Cognitive Foundations: A Survey. ACM Transactions on Applied Perception 7(1), 1–39 (2010)
2. Sridharan, M., Stone, P.: Color learning and illumination invariance on mobile robots: A survey. Robotics and Autonomous Systems 57(6-7), 629–644 (2009)
3. Forsyth, D.A.: A novel algorithm for color constancy. International Journal of Computer Vision 5(1), 5–35 (1990)
4. Klinker, G.J., Shafer, S.A., Kanade, T.: A physical approach to color image understanding. International Journal of Computer Vision 4, 7–38 (1990)
5. Finlayson, G.D., Hordley, S.D., Hubel, P.M.: Color by correlation: A simple, unifying framework for color constancy. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(11), 1209–1221 (2001)
6. Brainard, D.H., Freeman, W.T.: Bayesian color constancy. Journal of the Optical Society of America A, Optics, Image Science, and Vision 14(7), 1393–1411 (1997)
7. Schulz, D., Fox, D.: Bayesian color estimation for adaptive vision-based robot localization. In: IROS (2004)
8. Luan, X., Qi, W., Song, D., Chen, M., Zhu, T., Wang, L.: Illumination invariant color model for object recognition in robot soccer. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010, Part II. LNCS, vol. 6146, pp. 680–687. Springer, Heidelberg (2010)
9. Rasolzadeh, B., Björkmann, M., Huebner, K., Kragic, D.: An Active Vision System for Detecting, Fixating and Manipulating Objects in the Real World. The International Journal of Robotics Research 29(2-3), 133–154 (2009)
10. Frintrop, S.: VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search. LNCS (LNAI), vol. 3899. Springer, Heidelberg (2006)
11. Frintrop, S., Nüchter, A., Pervölz, K., Surmann, H., Mitri, S., Hertzberg, J.: Attentive Classification. International Journal of Applied Artificial Intelligence in Engineering Systems 1(1) (2009)
12. Garcia, J.F., Rodríguez, F.J., Matellán, V., Fernández, C.: Saliency map based attention control for the RoboCup SPL. In: Workshop of Physical Agents (2010)
13. Itti, L., Koch, C., Niebur, E.: A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(11), 1254–1259 (1998)
14. Tsotsos, J.K., Culhane, S.M., Kei Wai, W.Y., Lai, Y., Davis, N., Nuflo, F.: Modeling visual attention via selective tuning. Artificial Intelligence 78(1-2), 507–545 (1995)
15. Koch, C., Ullman, S.: Shifts in selective visual attention: towards the underlying neural circuitry. Human Neurobiology 4(4), 219–227 (1985)
16. Treisman, A.M., Gelade, G.: A feature-integration theory of attention. Cognitive Psychology 136(12), 97–136 (1980)
17. Itti, L.: Models of Bottom-Up and Top-Down Visual Attention. PhD thesis, California Institute of Technology (2000)
18. Itti, L., Koch, C.: A saliency-based search mechanism for overt and covert shifts of visual attention. Vision Research 40(10-12), 1489–1506 (2000)
19. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(11), 1254–1259 (2002)
20. Li, Z., Fang, T., Huo, H., Zhu, J.: Color conspicuity map based on wavelet low-pass pyramid for popping out contours of salient objects. Optical Engineering 49(5), 050502 (2010)

21. Engel, S., Zhang, X., Wandell, B.: Colour tuning in human visual cortex measured with functional magnetic resonance imaging. Nature 388(6637), 68–71 (1997)
22. Specht, D.F.: A general regression neural network. IEEE Transactions on Neural Networks 2(6), 568–576 (1991)
23. Itti, L., Rees, G., Tsotsos, J.K.: Models of bottom-up attention and saliency. Neurobiology of Attention 582, 1–11 (1980)
24. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of NAO humanoid. In: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 2124–2129. IEEE Press, Piscataway (2009)
25. The RoboCup Standard Platform League, `http://www.tzi.de/spl`

A Robust Place Recognition Algorithm Based on Omnidirectional Vision for Mobile Robots

Huimin Lu, Kaihong Huang, Dan Xiong, Xun Li, and Zhiqiang Zheng

College of Mechatronics Engineering and Automation,
National University of Defense Technology, Changsha, China
{lhmnew,lixun,zqzheng}@nudt.edu.cn

Abstract. In this paper, bag-of-features, a popular and successful approach in pattern recognition community, is used to realize place recognition based on omnidirectional vision for mobile robots by combining the real-time local visual features proposed by ourselves for omnidirectional vision and support vector machines. The panoramic images from the COLD database were used to perform experiments to determine the best algorithm parameters and the best training condition. The experimental results show that the robot can realize robust place recognition with high classification rate in real-time by using our algorithm.

1 Introduction

In recent years, along with the development of image understanding and pattern recognition, visual place/scene recognition has attracted more and more researchers' interest, and many progresses have been achieved [1][2]. Place recognition can be applied to realize robot topological localization. If the nodes of topological maps are represented by the places like kitchen, corridor, and bathroom, once these places are recognized and classified by robots, topological localization is also realized for the robots. Besides topological localization, place recognition is also important for solving the loop closing in visual odometry, visual SLAM and the kidnapping problem in robot localization.

Pronobis, Caputo, Luo, et al. proposed a robust place recognition algorithm based on SVMs classifier, combined with local visual features computed using a Harris-Laplace detector and the SIFT descriptor in [3]. Because the number of the local visual features in an image is not fixed, the local descriptors are used as the input of SVMs via a match kernel. Then the classifiers can be trained for place classification and recognition. The local visual features are used as the input of SVMs directly, so large memory space is needed to store those features used as support vectors. Therefore, they proposed a memory-controlled incremental SVMs by combining an incremental extension of SVMs with a method reducing the number of support vectors needed to build the decision function without any loss in performance introducing a parameter which permits a user-set trade-off between performance and memory in [2]. They also built up several image databases to provide standard benchmark datasets for the development, evaluation and comparison of different place recognition algorithms: the INDECS [1], the IDOL [1] and the COLD [4] database. All the images in these databases were acquired in indoor environments and with different conditions like different robot platforms, different lighting conditions, and different labs across Europe. Although the COLD database includes panoramic images acquired by the omnidirectional vision system, only perspective images were used to perform experiments to test their place recognition methods in [2][3]. The omnidirectional vision system can provide a 360° view of the robot's surrounding environment in a single image, and it is especially suitable to be a sensor of navigation for mobile robots in large scale environment.

In this paper, bag-of-features [5][6], a popular and successful approach in pattern recognition community, is used to realize robust place recognition based on omnidirectional vision for mobile robots by combining two novel real-time local visual features [7] proposed by ourselves and support vector machines (SVMs) [8]. Some researchers used or extended the bag-of-features method to realize qualitative localization [9], global localization [10], or topological SLAM [11], which are the most similar research with our work in this paper. Only perspective images were used in their work. Furthermore, the local visual features used in their work can not be extracted in real-time, so their algorithms can not be run in real-time actually.

2 Two Real-Time Local Visual Features

Local visual features have become increasingly popular in recent years, and they have been applied very well in many computer/robot vision problems. Although a number of algorithms have been proposed with respect to feature detectors and feature descriptors, a common deficiency for most of the existing algorithms is that their computation costs are usually high. This deficiency limits the actual application of local visual features, especially in those situations with high real-time requirements, such as robot navigation, self-localization. When local visual features are applied to omnidirectional vision, the original algorithms should be modified because of its special imaging character, especially in determining the feature regions [12].

To deal with these problems, we proposed two novel real-time local visual features for omnidirectional vision [7]. Features from Accelerated Segment Test (FAST) [13] is used as the feature detector to detect corner features in the panoramic image. Then we adopted the feature region determining method proposed in [14] to achieve rotation invariance. Rectangular image regions surrounding corner features are firstly determined in the radial direction, and then rotated to a fixed orientation, as shown in Fig. 1(a) and Fig. 2(a). Finally, local binary pattern (LBP) [15] and center-symmetric local binary pattern (CS-LBP) [15] are used as feature descriptors to compute vectors to describe the information of feature regions. So two algorithms named FAST+LBP and FAST+CSLBP were designed. FAST, LBP and CS-LBP are computationally simple, so they can be the basis of our real-time local visual features.

We performed feature matching experiments by using the panoramic images in the COLD database to determine the best algorithm parameters and to compare with SIFT [16]. The final FAST+LBP and FAST+CSLBP with best algorithm parameters are shown in Fig. 1 and Fig. 2 respectively. The descriptor dimension of FAST+LBP and FAST+CSLBP are 236 and 72. The experimental results in [7] show that our algorithms have better performance than SIFT. The computation time needed to extract all the features in an image by FAST+LBP or FAST+CSLBP is from 5ms to 20ms, so our local visual features can be extracted in real-time, and they can be applied to computer/robot vision tasks with high real-time requirements like place recognition for mobile robots in this paper. Their performance will be compared with SIFT and SURF [17] when applied to place recognition in Section 4.2.

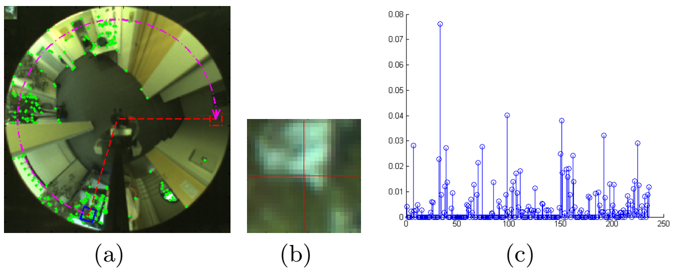


Fig. 1. The final FAST+LBP algorithm. (a) A feature region on the panoramic image. The green points are the detected corner features. (b) The scale-up feature region. (c) The resulting feature descriptor. The descriptor dimension is 236.

3 The Proposed Place Recognition Algorithm Based on Omnidirectional Vision

Bag-of-features [5][6] is a popular approach in computer vision/pattern recognition community, and has been applied successfully to object recognition, video retrieval, scene classification, etc. SVMs [8] is one of the most successful classifier learning methods in pattern recognition. In this section, we try to use bag-of-features to achieve robust and real-time place recognition based on omnidirectional vision for mobile robots by combining the real-time local visual features presented in Section 2 and SVMs. Our place recognition algorithm is divided into two phases: the phase of off-line training and the phase of on-line testing. The diagram of the algorithm is demonstrated in Fig. 3.

In the phase of off-line training, we assume that the number of the panoramic images for training is m, the number of place categories is M, and the corresponding place category of each image is also known. The local visual features f_i are extracted from each training image, where $i = 1 ... \sum_{j=1}^{m} n_j$, and n_j is the number of the local visual features extracted from the jth training image. After clustering these features with K-means clustering algorithm, we get clustering

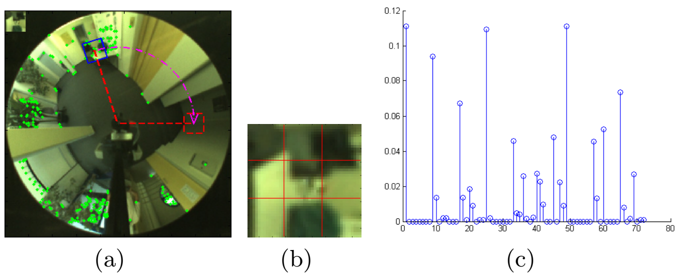
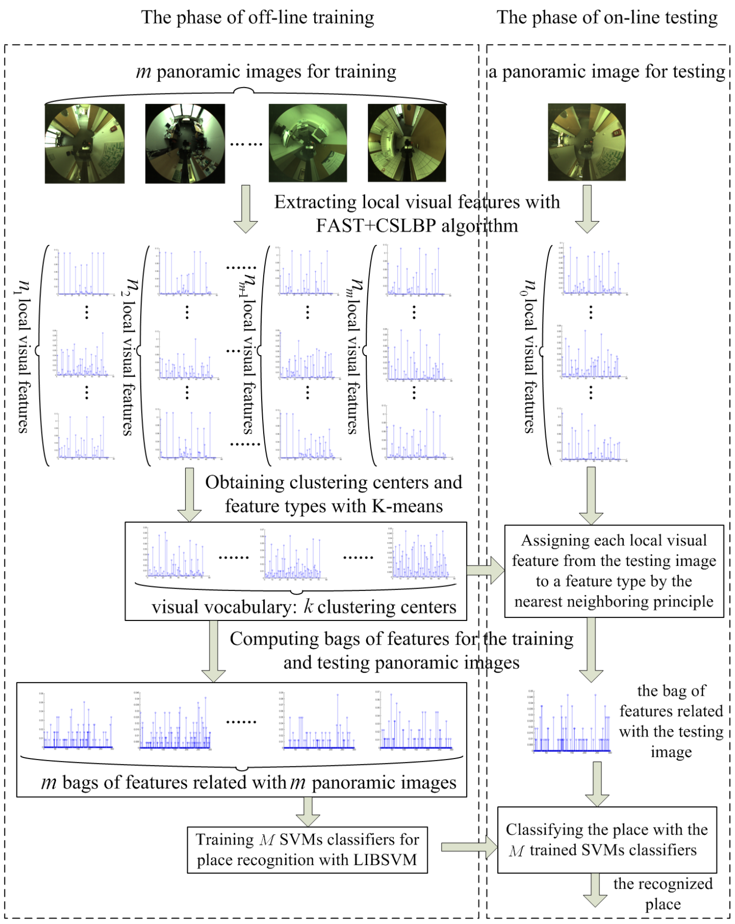


Fig. 2. The final FAST+CSLBP algorithm. (a) A feature region on the panoramic image. The green points are the detected corner features. (b) The scale-up feature region. (c) The resulting feature descriptor. The descriptor dimension is 72.



The phase of off-line training
The phase of on-line testing
m panoramic images for training
a panoramic image for testing
Extracting local visual features with FAST+CSLBP algorithm
n1 local visual features
n2 local visual features
nm-1 local visual features
nm local visual features
n0 local visual features
Obtaining clustering centers and feature types with K-means
visual vocabulary: k clustering centers
Assigning each local visual feature from the testing image to a feature type by the nearest neighboring principle
Computing bags of features for the training and testing panoramic images
m bags of features related with m panoramic images
the bag of features related with the testing image
Training M SVMs classifiers for place recognition with LIBSVM
Classifying the place with the M trained SVMs classifiers
the recognized place

Fig. 3. The diagram of our place recognition algorithm based on omnidirectional vision and local visual features

centers C_i, where $i = 1...k$, and k is the clustering number. These clustering centers form the visual vocabulary which is similar as the word vocabulary for text categorization.

After the clustering is finished, each local visual feature from the training image has also been assigned to the corresponding cluster, which means that the feature type is obtained. A feature vector x_j is computed by normalizing the histogram constructed with the number of occurrences of each type of the features from the visual vocabulary in the jth training image, where $x_j \in R^k$. The feature vector is an effective representation of the image information, and it is named as bag of features. Then the bags of features and the corresponding place categories of all training images are used to learn M classifiers by applying the famous SVMs software - LIBSVM [8] according to the *one-vs-all* strategy.

During the training process mentioned above, the algorithm setup like the different local visual features, the different clustering numbers, the different kernel functions, and the completeness of the visual vocabulary, will affect the performance of our place recognition algorithm. We will determine the best algorithm setup by experiments in the next section.

In the phase of on-line testing, the local visual features are extracted from the testing panoramic image (or the image acquired on-line by the robot's vision system), and then each local visual feature is assigned to a feature type according to its distances to the clustering centers by the nearest neighboring principle. The bag of features of the testing image is computed by normalizing the histogram constructed with the number of occurrences of each type of the features from the visual vocabulary in the testing image. Finally, this bag of features is used as the input of the learned M classifiers, and the outputs are the classification results and the corresponding classification probability. The classification result with the largest classification probability is used as the final place category.

Omnidirectional vision is used in our algorithm, and better performance in place recognition should be achieved than those methods only using perspective images, because omnidirectional vision can provide a 360^o view of the robot's surrounding environment in a single image, which will be verified by the experimental results in the next section.

4 The Experimental Results

In this section, we will introduce the experimental setup firstly, and then test and analyze that how the algorithm performance will be affected by the factors like the choice of the local visual feature, the clustering number, the kernel function, and the training condition. Therefore, the best algorithm parameters and the best training condition can be determined. The performance will be presented in detail when the best parameters and the best training condition are used. Finally the real-time performance will be discussed.

4.1 Experimental Setup

COLD is a freely available database which provides a large-scale, flexible testing environment for vision-based place recognition. COLD contains 76 image sequences acquired in three different indoor environments across Europe. The images are acquired by the same perspective and omnidirectional vision in different rooms and under various illumination conditions. We will use the following six sequences of the panoramic images in COLD-Saarbruecken to perform our experiments: seq3_cloudy1, seq3_cloudy2, seq3_night1, seq3_night2, seq3_sunny1, seq3_sunny2. The “cloudy”, “night”, and “sunny” indicate the corresponding illumination conditions under which the image sequences are acquired. Four places are included in each of these image sequences: corridor, one-person office, printer area, and bath room. Although there are only four places, the sequences are long-term, and over 700 panoramic images are included in each sequence. More details about COLD can be found in [4].

4.2 The Choice of the Local Visual Feature

In this experiment, we compare the algorithm performance when using different local visual features: FAST+LBP, FAST+CSLBP, SIFT, and SURF. The clustering number was set to be 200, and linear kernel was used in SVMs. During the experiment, we used the image sequence seq3_cloudy1, seq3_night2, seq3_sunny2 for training respectively, and then used seq3_cloudy2, seq3_night1, seq3_sunny1 for testing respectively. Because there is a certain degree of randomness in the clustering results obtained by using K-means clustering algorithm, the training and testing processes were run several times to get the average place classification rate. The experimental results are shown in table 1 when different local visual feature was chosen. We see that the overall performance is much better when using FAST+LBP or FAST+CSLBP than using SIFT or SURF, which also validates that the discriminative power of FAST+LBP and FAST+CSLBP are good. There is not much difference in the overall performance when using FAST+LBP or FAST+CSLBP. However, the descriptor dimension of FAST+CSLBP is 72, and it is much smaller than that of FAST+LBP, which causes the lower computation cost of the place recognition algorithm. So we choose FAST+CSLBP as the local visual feature in the following experiments.

4.3 The Clustering Number

In this experiment, we compare the algorithm performance affected by different clustering numbers. FAST+CSLBP and linear kernel were used in the algorithm. During the experiment, we used the image sequence seq3_cloudy1 for training, and then used seq3_cloudy2, seq3_night1, seq3_sunny1 for testing respectively. The training and testing processes were also performed several times to get the average place classification rate. The experimental results are shown in Fig. 4 when the clustering number was set to be 50, 100, 150, 200, 250, 300, 350 and 400. In the general trend, the algorithm performance increases as the increase of

Table 1. The place classification rates when choosing different local visual feature

		training		
		seq3_cloudy1	seq3_night2	seq3_sunny2
seq3_cloudy2 for testing	FAST+LBP	0.9375	0.9245	0.8216
	FAST+CSLBP	0.9611	0.9364	0.9468
	SIFT	0.8168	0.7282	0.8361
	SURF	0.8313	0.7255	0.7954
seq3_night1 for testing	FAST+LBP	0.9472	0.9621	0.9149
	FAST+CSLBP	0.9254	0.8659	0.9470
	SIFT	0.8495	0.8342	0.8892
	SURF	0.7225	0.7849	0.8150
seq3_sunny1 for testing	FAST+LBP	0.8867	0.7238	0.7493
	FAST+CSLBP	0.8666	0.6802	0.6863
	SIFT	0.8083	0.7362	0.6335
	SURF	0.7760	0.7711	0.7813

the clustering number, which is consistent with the research results [5] in pattern recognition community. But the increase of the clustering number will make the vocabulary size larger and then cause higher computation cost in the testing process, so a compromise should be made between the classification rate and the clustering number. In the following experiments, the clustering number will be set to be 300 as the best parameter.

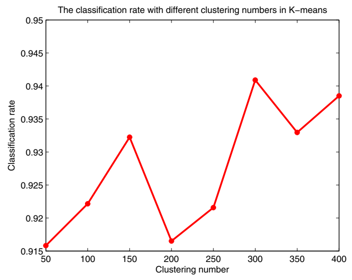


Fig. 4. The place classification rates when using different clustering number in the algorithm

4.4 The Choice of the Kernel Function in SVMs

In this experiment, we compare the algorithm performance affected by using different kernel functions in SVMs: linear kernel, RBF kernel, Sigmoid kernel. FAST+CSLBP was used, and the clustering number was set to be 300 in the algorithm. During the experiment, we used the image sequence seq3_cloudy1 for training, and then used seq3_cloudy2, seq3_night1, seq3_sunny1 for testing respectively. The training and testing processes were also performed several times to get the average place classification rate. The experimental results are shown in table 2. We see that there is not much difference in the overall performance when using different kernel functions. Because linear kernel is computationally simplest, it will be used as the best kernel function in the following experiments.

Table 2. The place classification rates when using different kernel function in the algorithm

	linear kernel	RBF kernel	Sigmoid kernel
seq3_cloudy2 for testing	0.9521	0.9632	0.9313
seq3_night1 for testing	0.9594	0.9477	0.9516
seq3_sunny1 for testing	0.9336	0.9475	0.9476

4.5 The Completeness of the Visual Vocabulary

In this experiment, the best algorithm parameters determined above were used, which means that FAST+CSLBP and linear kernel were chosen, and the clustering number was set to be 300. During the experiment, we used the image sequence seq3_cloudy1, seq3_night2, seq3_sunny2 for training respectively, and then used seq3_cloudy2, seq3_night1, seq3_sunny1 for testing respectively. The average place classification rates were acquired to compare which image sequence was best for training to achieve the best performance. The experimental results are shown in table 3. We clearly see that the highest classification rate is achieved when using the image sequence acquired under "cloudy" illumination condition for training. The same conclusion can also be obtained from the experimental results in Section 4.2. When the illumination condition is "night" or "sunny", and the robot is located in the position where the illumination is affected greatly by the natural light, the acquired image may be less-exposed or over-exposed. Then some local visual features cannot be extracted, which may cause that the visual vocabulary is incomplete. The incompleteness of the visual vocabulary will lead to the decrease of the place classification rate, which is the same as the situation in text categorization that the incompleteness of the word vocabulary will result in the decrease of the text categorization rate. The "cloudy" illumination is more stable than "night" and "sunny", so the image sequence acquired under "cloudy" illumination condition is best for training.

Furthermore, we used seq3_cloudy1, seq3_night2 and seq3_sunny2 jointly for training, and then used seq3_cloudy2, seq3_night1, seq3_sunny1 for testing respectively. The experimental results are also shown in table 3. The place classification rates are improved when using seq3_cloudy2 and seq3_night1 for testing. But when seq3_sunny1 is used for testing, the performance is still much worse than that when only using seq3_cloudy1 for training. So in the following experiments, seq3_cloudy1 will be used for training.

Table 3. The place classification rates when different image sequences acquired under different illumination conditions were used for training

	training			
	seq3_cloudy1	seq3_night2	seq3_sunny2	all seqs
seq3_cloudy2 for testing	0.9296	0.9366	0.9523	0.9634
seq3_night1 for testing	0.9550	0.7190	0.9516	0.9707
seq3_sunny1 for testing	0.9380	0.5919	0.6379	0.8362

4.6 The Performance with the Best Parameters and Training Condition

Through the experiments mentioned above, we have determined the best algorithm parameters, and the illumination condition under which the best training image sequence is acquired. In this experiment, the best parameters were used in the algorithm. The best image sequence seq3_cloudy1 was used for training, and seq3_cloudy2, seq3_night1, seq3_sunny1 were used for testing respectively, so the algorithm performance can be analyzed in detail.

Because of the randomness of the clustering process, we only demonstrate the best results after training several times. When seq3_cloudy2 was used for testing, the detailed result of place classification is shown in table 4, where the statistics of how many images being correctly and wrongly classified are listed. The place classification rate is 0.9806. Some panoramic images which were wrongly classified are shown in Fig. 5. When seq3_night1 was used for testing, the detailed result of place classification is shown in table 5. The place classification rate is 0.9594. Some panoramic images which were wrongly classified are shown in Fig. 6. When seq3_sunny1 was used for testing, the detailed result of place classification is shown in table 6. The place classification rate is 0.9429. Some panoramic images which were wrongly classified are shown in Fig. 7.

Table 4. The detailed result of place recognition when using seq3_cloudy2 for testing

real places	recognition results			
↓	corridor	one-person office	printer area	bath room
corridor	277	0	0	0
one-person office	5	106	0	0
printer area	2	0	77	0
bath room	7	0	0	246

From the experimental results, we clearly see that high place classification rate can be achieved by using our algorithm. Most of those panoramic images which were wrongly classified are acquired when the robot is located near the border of two different places. Because the omnidirectional vision system can provide a 360° view of the robot's surrounding environment, when the robot is located near the border, both of the scenes belonging to the two places will be

Table 5. The detailed result of place recognition when using seq3_night1 for testing

real places	recognition results			
↓	corridor	one-person office	printer area	bath room
corridor	290	11	0	1
one-person office	6	108	0	0
printer area	6	0	94	0
bath room	7	0	0	241

Table 6. The detailed result of place recognition when using seq3_sunny1 for testing

real places	recognition results			
↓	corridor	one-person office	printer area	bath room
corridor	253	9	4	2
one-person office	2	95	0	0
printer area	5	0	99	0
bath room	21	0	0	263

included in the panoramic image. Furthermore, the panoramic images are not labeled according to their content but to the position of the robot at the time of acquisition. So in this case, the classification error cannot be completely avoided.

In comparison with the place classification results in [3], where only the perspective images in the COLD database were used, better performance is achieved by our algorithm. This can be explained as follows: our method is based on omnidirectional vision, and the changes of the panoramic image with the different robot's positions are not so rapid as that of the perspective image, so omnidirectional vision is more suitable for place recognition than perspective camera; our FAST+CSLBP feature is discriminative and robust; the bag-of-features method itself is powerful for place recognition.

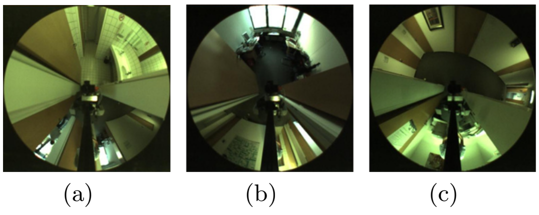


Fig. 5. Some wrongly classified images when using seq3_cloudy2 for testing. Bath room (a), one-person office (b), printer area (c) were wrongly classified as corridor.

4.7 The Real-Time Performance

The real-time performance is very important in the actual application for mobile robots. Because the training process of our algorithm is off-line, only the online testing process should be analyzed. In the testing process, the algorithm

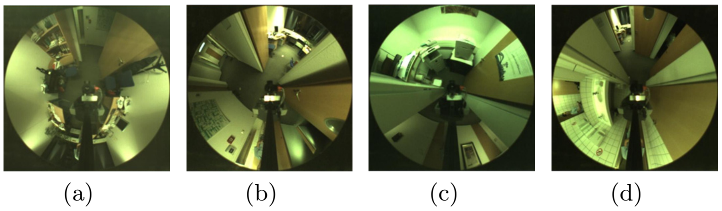


Fig. 6. Some wrongly classified images when using seq3_night1 for testing. One-person office (a), printer area (c), bath room (d) were wrongly classified as corridor. (b) Corridor was wrongly classified as one-person office.

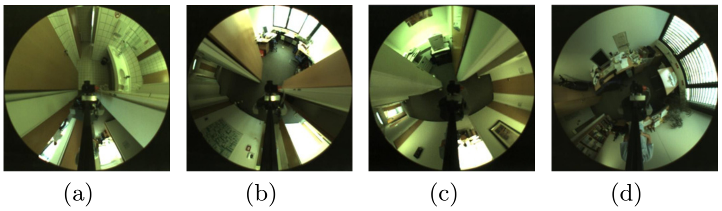


Fig. 7. Some wrongly classified images when using seq3_sunny1 for testing. Corridor was wrongly classified as bath room (a), one-person office (b), and printer area (c). (d) One-person office was wrongly classified as corridor.

consists of three parts: the extraction of local visual features, the construction of bag of features, and place classification with SVMs. The computer is equipped with 2.26 GHz Duo CPU and 1.0G memory. According to the experimental results in [7], the computation time needed to extract all the local visual features in a panoramic image by FAST+CSLBP is from 5 to 20 ms. When the best parameters in Section 4.5 are used, the construction of bag of features and place classification with SVMs can be finished in 10 ms. The whole place recognition can be finished in 30 ms, so our algorithm can be run in real-time. We can clearly see that using the real-time local visual features is very important to make our algorithm satisfy the real-time requirement.

5 Conclusion

In this paper, the bag-of-features method is used to solve place recognition based on omnidirectional vision for mobile robots by combining the real-time local visual features proposed by ourselves and SVMs. The panoramic images in the COLD database were used to perform experiments to test the affection on the performance by different algorithm factors like the choice of the local visual feature, the clustering number, the choice of the kernel function in SVMs, and the completeness of the visual vocabulary. So the best algorithm parameters and the illumination condition under which the best training image sequence was acquired were determined, and the performance of place recognition with these best parameters and the best training condition was analyzed in detail.

The real-time performance was discussed finally. The experimental results show that place recognition can be realized in real-time with high classification rate by using the proposed algorithm.

References

1. Pronobis, A., Caputo, B., Jensfelt, P., Christensen, H.I.: A realistic benchmark for visual indoor place recognition. Robotics and Autonomous Systems 58, 81–96 (2010)
2. Pronobis, A., Jie, L., Caputo, B.: The more you learn, the less you store: Memory-controlled incremental SVM for visual place recognition. Image and Vision Computing 28, 1080–1097 (2010)
3. Ullah, M.M., Pronobis, A., Caputo, B., et al.: Towards robust place recognition for robot localization. In: Proceedings of the 2008 IEEE ICRA, pp. 530–537 (2008)
4. Pronobis, A., Caputo, B.: COLD: The Cosy Localization Database. The International Journal of Robotics Research 28(5), 588–594 (2009)
5. Csurka, G., Dance, C.R., Fan, L., et al.: Visual categorization with bags of keypoints. In: Proceedings of ECCV 2004 Workshop on Statistical Learning in Computer Vision, pp. 59–74 (2004)
6. Sivic, J., Zisserman, A.: Video Google: A Text Retrieval Approach to Object Matching in Videos. In: Proceedings of the 9th IEEE ICCV, pp. 1–8 (2003)
7. Lu, H., Zheng, Z.: Two novel real-time local visual features for omnidirectional vision. Pattern Recognition 43, 3938–3949 (2010)
8. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), Software available at, `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
9. Filliat, D.: A visual bag of words method for interactive qualitative localization and mapping. In: Proceedings of the 2007 IEEE ICRA, pp. 3921–3926 (2007)
10. Fraundorfer, F., Engels, C., Nistér, D.: Topological mapping, localization and navigation using image collections. In: Proceedings of the 2007 IEEE/RSJ IROS, pp. 3872–3877 (2007)
11. Cummins, M., Newman, P.: FAB-MAP: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research 27, 647–665 (2008)
12. Svoboda, T., Pajdla, T.: Matching in Catadioptric Images with Appropriate Windows, and Outliers Removal. In: Skarbek, W. (ed.) CAIP 2001. LNCS, vol. 2124, pp. 733–740. Springer, Heidelberg (2001)
13. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006)
14. Andreasson, H., Treptow, A., Duckett, T.: Self-Localization in non-stationary environments using omni-directional vision. Robotics and Autonomous Systems 55, 541–551 (2007)
15. Heikkilä, M., Pietikäinen, M., Schmid, C.: Description of interest regions with local binary patterns. Pattern Recognition 42, 425–436 (2009)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (2004)
17. Bay, H., Ess, A., Tuytelaars, T., et al.: Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding 110, 346–359 (2008)

Ball Sensing in a Leg Like Robotic Kicker

Jonas Logghe, André Dias, José Almeida, Alfredo Martins, and Eduardo Silva

INESC TEC - INESC Technology and Science
(formerly INESC Porto) and ISEP/IPP - School
of Engineering, Polytechnic Institute of Porto
Rua Dr Antonio Bernardino de Almeida, 431, Porto, Portugal
{jonaslogghe,adias,jma,aom,eaps}@lsa.isep.ipp.pt
http://www.lsa.isep.ipp.pt

Abstract. The trend to have more cooperative play and the increase of game dynamics in Robocup MSL League motivates the improvement of skills for ball passing and reception. Currently the majority of the MSL teams uses ball handling devices with rollers to have more precise kicks but limiting the capability to kick a moving ball without stopping it and grabbing it. This paper addresses the problem to receive and kick a fast moving ball without having to grab it with a roller based ball handling device. Here, the main difficulty is the high latency and low rate of the measurements of the ball sensing systems, based in vision or laser scanner sensors.Our robots use a geared leg coupled to a motor that acts simultaneously as the kicking device and low level ball sensor. This paper proposes a new method to improve the capability for ball sensing in the kicker, by combining high rate measurements from the torque and energy in the motor and angular position of the kicker leg. The developed method endows the kicker device with an effective ball detection ability, validated in several game situations like in an interception to a fast pass or when chasing the ball where the relative speed from robot to ball is low. This can be used to optimize the kick instant or by the embedded kicker control system to absorb the ball energy.

Keywords: Middle Size League, Ball sensing, Leg like Kicker.

1 Introduction

Robocup is an international project that aims to promote robotics by providing a standard problem (soccer game) as a central topic of research, with the intention of producing innovations (hardware and software) to be applied to society and industry. The ultimate goal of the RoboCup project is "By 2050, to develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer." [3]. To achieve this, much research has yet to be done in several areas, such as mechatronics, perception in highly dynamic and noisy environments, intelligent control, cooperative work, players coordination, strategies adaptation and learning, only to name a few.

To play football, some required fundamental skills are ball control and manipulation, passing and receiving the ball, intercepting and kicking a ball.

In MSL, a lot of work has been done in this area. The kicker and ball handling mechanisms suffered several improvements at all levels, (mechanic and electronic), in

Robocup's past years. By 1997, the MSL robots had no kicker devices[13]. The robots only pushed the ball around, some using a passive finger like ball handler mechanisms. In the following years teams started to develop kicker mechanisms. Those could be based in spring[13], pneumatic[14] and solenoid[15][9] devices. By 2002/2003 some teams started using stronger spring[4][7] and solenoid kicker[8] devices in competitions. Due to the advantage of having, in competition, a strong shot many teams started to use similar systems. Ball handling mechanisms evolved from passive fingers to active fingers and roller mechanisms. There was always some controversy about the usage of active rollers. Although there were some attempts in the rules to restrict roller based devices, namely around 2003/2004, those rules' spirit was later on a bit distorted, leading to different rules were those devices are allowed if the ball "rotates in its natural direction of rotation".

Only few research efforts were done in different types of ball handling devices. In 2003, Mu-Wallabies team presented a robot with an arm like kicker[5], and Philips team had demonstrated some prototypes for ball stopping devices[7]. Later, some research was done concerning ball stopping devices [11][15]. But, approaches to control the ball without continuous robot-ball contact are more rare. In Robocup's 2010 technical challenge the Tech United team presented a control behavior to dribble forward the ball with small taps[15]. And in 2011 ISePorto's team started using only small kicks to move forward in the field and to intercept the ball[12].

Currently several MSL teams use sophisticated mechanic roller based ball handling devices to have more precise kicks[6], but that type of ball handling limits the capability to kick a moving ball without stopping it and "sucking/grabbing it". Additionally it is not adequate to dribble the ball in a noncontinuous contact. Having in mind the classic definition ball dribbling: "dribbling refers to the maneuvering of a ball around a defender through short skillful taps or kicks"[10]. And the desire to have games similar to human football games, associated to the increase of game dynamics in Robocup MSL League, motivates a radical change in the ball manipulation skills.

For development of skills like short skillful taps or kicks, first-touch control or one-touch play, one key problem is latency and low rate of the measurements provided by the ball sensing systems typically used, based in vision or laser scanner sensors.

Focus on the Robocup Middle Size League, the goal of this paper is to present the development and results of a novel ball sensing approach for a leg like kicker, developed to enable the robot to receive, intercept and kick a fast moving ball, and pass and dribble it forward with a similar behavior to that of a human soccer player.

The robots from ISePorto Team use a leg coupled through gears to a DC motor with an optical encoder, that acts simultaneously as the kicking device and as a low level ball sensor. The 2011 version of the system uses only the movement and velocity of the kicker to detect the ball. To receive a ball the kicker is moved to a receiving position in front of the robot, this way an incoming ball can move the kicker a certain distance. When kicker leg is pushed back by some threshold value (A on Fig.3) and the kicker reaches a certain velocity a kick is performed (B on Fig.3). This system suffers from a sensitivity problem. When the robot was driving and expecting a ball, the sensing mechanism sometimes reacted on the movement of the robot. Also it was not possible to intercept a moving ball by chasing after it and let it hit the kicker.

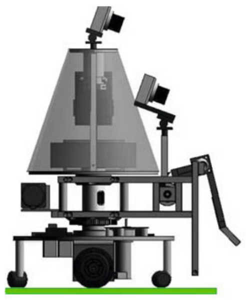


Fig. 1. Middle Size League Robot

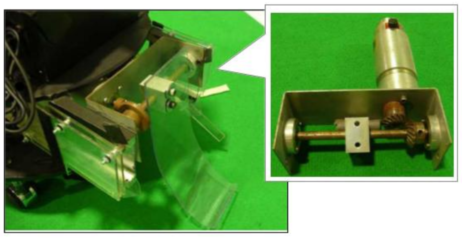


Fig. 2. Kicker Device

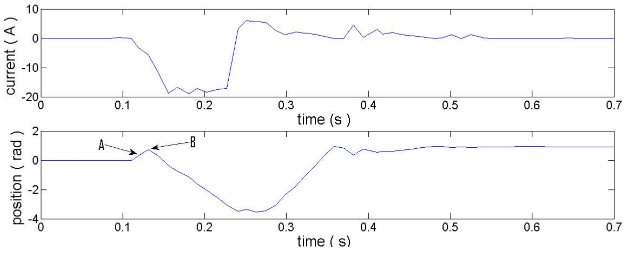


Fig. 3. kick with current detection system

The new ball sensing method presented in this paper, runs in the embedded kicker control system and improves the capability of ball sensing in the kicker, by combining higher rate measurements from the torque and energy in motor and angular position of kicker leg. It provides detection events within a few milliseconds, that can be used both in the kicker control, for ball reception or kick instant optimization, or by the robot control applications running in the main robot computer. The information provided by this new sensor is complementary to other available in the robot perception system, and is used in the low level feedback and in state transitions in the embedded kicker control system.

The outline of this paper is as follows. In section 2, the problem of low level ball sensing is analysed in order to identify a set of game situations that must be distinguished by the system. Then a model of the kicker is presented. In section 4, alternative detection methods, like: thresholds in motor current (proportional to the motor torque), derivative of the current, and integration of current (proportional to the energy) are proposed, tested and compared in simulation. Issues of the implementation of the detection methods in the embedded kicker control system are addressed in section 5, and results of the its application in the robots are presented in section 6. Finally, some conclusion are drawn about the implemented method and some future improvements are proposed.

2 Requirement Analysis

In this section, we will discuss the behavior of the kicker during a Middle Size League in order to typify game situations that could influence the low level ball detection.

- **Receiving a ball:** In a human soccer game, the player receives the ball by using the foot against the ball. During a game, this behavior (see Figure 4) occurs with high frequency, specially if the team is performing cooperative actions by passing and receiving the ball. The ability to sense the ball in this situation is harder when the ball comes with lower speed.
- **Robot acceleration:** When accelerating, the inertia of the kicker causes it to move in the opposite direction of the acceleration (see Figure 5). With the former detecting mechanism the kicker sometimes performed a kick in this situation.
- **Robot collision:** Sometimes during a game the robot collides with another robot. If this happens when the robot is in the receiving position the kicker will move as a result of the collision (see Figure 6). This situation was analysed but is not the priority of the new sensing mechanism.
- **Chasing a ball:** This situation is the hardest one to detect. When a ball is moving with a certain velocity and the robot wants to intercept it for kicking or receiving the ball while moving, he needs to chase the ball (see Figure 7). When the ball hits the kicker the relative velocity of the ball to the kicker is very small, becoming hard to detect.

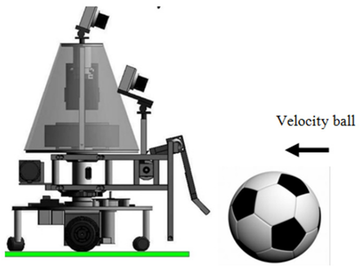


Fig. 4. Receiving a ball

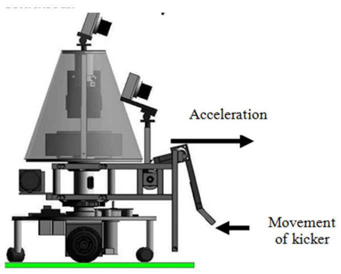


Fig. 5. Robot acceleration

3 Kicker Model

To be able to do quick testing and understanding how the dynamics of the robot and kicker work a model was built in Matlab/Simulink [1]. The model is based on some equations that were derived from the schematic in Fig.8.

The first equation is the one from the electrical circuit.

$$V = E + R \times i + \frac{di}{dt}L \tag{1}$$

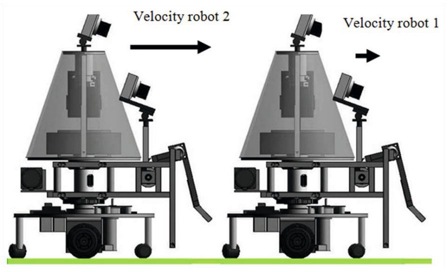


Fig. 6. Robot collision

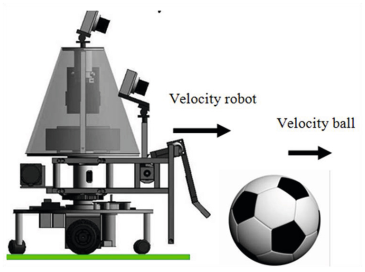


Fig. 7. Robot chasing a ball

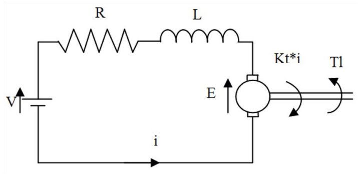


Fig. 8. schematic of the model

Here in R is the resistance, and L the inductance of the coil in the motor. E is the back electromotive force that is produced by the rotating motor and i the motor current. Equation 2 gives the relation of the back electromotive force to the velocity. K_{EMF} is the electromotive force constant.

$$E = \frac{d\theta}{dt} k_{EMF} \tag{2}$$

The torque produced by the motor T is related to the current in the armature by the torque constant k_t as shown in equation 3 .

$$T = k_t \times i \tag{3}$$

The torque produced by the motor is equal to the sum of all the torques that work in the opposite direction. In this sum the first term is the inertia of the rotor J that generates torque during a acceleration or deceleration. The second term is a torque by the damping of the system , here in b is the damping ratio of the motor. The third term is the sum of all the torques that are generated by the kicker system.

$$T = \frac{d^2\theta}{d^2} J + \frac{d\theta}{dt} b + T_l(\theta) \tag{4}$$

In equation 5 the sum of the torques generated by the system is presented. The first term is the torque generated by the inertia of the kicker, gears and connecting axles. The second term stands for the torque due to the friction and all the other influences that were not possible to calculate, so the c term was found by experiments. The last term is the torque created by the gravity of the kicker when it is not in vertical position.

$$T_l(\theta) = \frac{d^2\theta}{d^2} J_{kicker} + \frac{d\theta}{dt} c + T_{gravity}(\theta) + T_{load} \tag{5}$$

These equations resulted in a model that was tested and compared with the logs perform in the robot. The dynamic response of the model is comparable to real values. The developed model was aggregate into a subsystem where a position PID control is applied to. The model was originally designed to test PID settings of the kicker and not to test detection methods, although it can be used if taken in account that the results must be compared to reality. In this system different loads can be added in the T_{load} allowing to simulate different levels of charges (different types of kicker material).

4 Ball Sensing Methods

Having in mind the identified game situations and the requirements for the ball detection method, four detection methods were tested and compared in simulation using the developed kicker:

Peak current - The maximum current that occurs during a game situation;
Maximum derivative of the current - The current is differentiated and the maximum value used as a sensing measurement;
Average derivative - The derivative of the current between the moment that the current starts to rise until reaches its maximum;
Integrated current - The current is integrated from the moment it starts to rise until reaches its maximum.

Those four methods were applied to the model for three of the 4 situations. The situation where the robot collides with another robot is not simulated because there are many different ways this can happen and there is not much data available to compare results with.

Table 1. Overview of detecting methods on the model

		Game situations		
		incoming ball	acceleration	chasing a ball
Detecting method	Peak current [A]	1.42	0.29	0.9
	Max derivative [A/s]	131	4	82
	Average derivative [A/s]	10.14	0.454	6.42
	Integrated current [A.s]	0.145	0.050	0.092

On a first observation all of the applied methods could be used to detect the ball and distinguish the different situations. But when applying the detecting methods to the robots there are some technical limitations and problems that require a modification of the detection methods that are detailed in next section.

5 Embedded Implementation of the Sensing Methods

The kicker sensing control architecture presented in figure 9 is characterized by two hierarchical levels of action. The lower level is implemented in a dedicated embedded hardware responsable for acting in the following tasks:

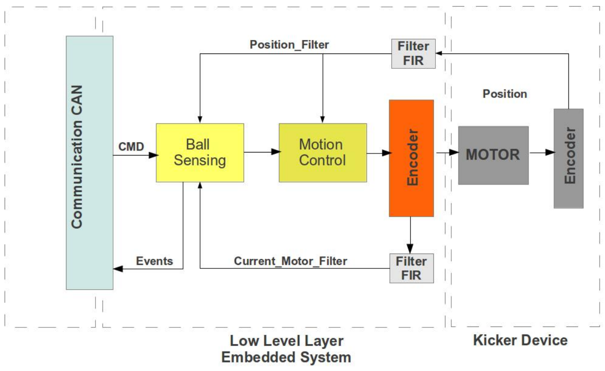


Fig. 9. Kicker Sensing implemented Architecture

- **Motion Control:** this task will perform control in position of the kicker through a proportional-integral-derivative controller based on command messages (KICK, RCV_BALL, KEEP_BALL, PREPARE_RCV_BALL) received by the CAN protocol or by the ball sensing task at low level.
- **Ball Sensing:** based on the information received from the motor current after being filtered and the position of the kicker this task processes the ball sensing methods by sending to the higher level (via CAN) all relevant information (continuous values and discrete events) and defining actions to the motion control task.

5.1 Current Filtering

When starting to monitor the current in the different situations one of the first things that came clear was the poor quality of the current signal. Therefore it was necessary to do some filtering on the incoming current signal.

Since the sampling frequency of the motor current can be much higher than the frequency of the current variations by the movement of the kicker, and the frequency of some noise sources is also higher, therefore a low-pass filter can reduce considerable the noise amplitude. A FIR (Finit Impulse Response) filter was chosen, since it does not require to calculate the filter output for every input like in an IIR filter (Infinite Impulse Filter). The output of this kind of filter is the sum of the current and previous inputs multiplied by filter coefficients. The filter structure:

$$[h]y(n) = \sum_{i=0}^{M} y_i x_{(n-i)} \tag{6}$$

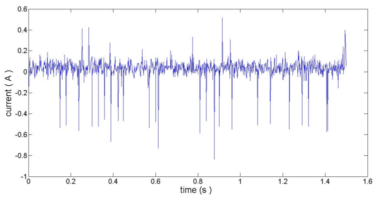


Fig. 10. Unfiltered signal

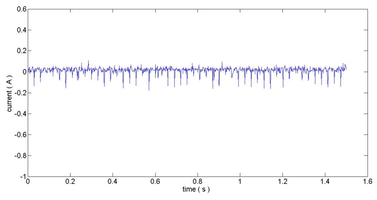


Fig. 11. Filtered signal

A downside for filtering is the delay on the filtered signal. This delay is a result of working with previous values. The delay depends on the sampling frequency Fs and filter order M:

$$delay = \frac{M-1}{2 \times Fs} \tag{7}$$

To design the filter the Matlab Signal Processing Tool (sptool)[2] was used. With this tool the filter coefficients for a determined filter specification were calculated.

Initially the current sampling and ball sensing happened in the same interrupt that was called every 1.4 ms. Considering the slope of a soft ball hitting the kicker has a typical duration of approximately 30ms and the detection had to happen as fast as possible, there was not much room for delay. Originally the sampling of the current happened at the same interrupt as the algorithm for the kick detecting. This interrupt is called every 1.4ms. So the sampling frequency is 714Hz. The first filters developed sampled and calculated at this speed. To make a filter with the required attenuation of the noise, the order was too high and so was the delay. To obtain a high order filter without much delay in the filtered signal the sampling of the current was put in another interrupt that is called every 0.14ms so the sampling frequency is 7.14KHz. The calculating of the filter still happens at the original speed (714Hz). It is not necessary to calculate it more frequently than it is used and it would consume too much time from the cpu. In this way a 10 order filter is obtained without the cost of lots of cpu time and delay. In Fig.11 is depicted the result of the 10th order filter that is used. The amplitude of the noise is reduced more than 5 times, and this makes the signal much more suitable for processing.

5.2 Applying the Detection Methods in the Embedded Control System

When applying the detection methods in the embedded control system to the robot some issues popped up immediately that led to some changes in the methods.

In the maximum derivative method, the numeric differentiation of the current signal is not adequate to the noise level in the current measurements. Although it work well with the developed kicker model in simulation, it does not behave well in the noisy measurements in the robot. This method caused a lot of false detection, even after the current filtering, and was not used in the tests with the robot.

The integrated current method had some problems as well. When the kicker was at its receiving position, the noise in the current signal made the integrated value reach fairly high values. To prevent this, reset conditions for the integrated value were programmed. A first reset condition applied when the current signal is lower than a threshold value (0.2A), is when the sign of the derivative changes. So it may be the start of a new slope which means that a new measurement should be started, so the integrated value is reset to zero. A second reset condition is used when the current has a value bigger or equal to 0.2A . In this case the derivative of the kicker angle is calculated and compared to the previous calculated value. If the sign of the derivative changes the integrated value is reset to zero. These two reset conditions make it certain that the integrated value is only from the last slope of the current.

6 Results

The methods previous presented were implemented in the embedded controller and applied to the different game situations and the table 2 was obtained. When comparing the values in table 2 there should be taken in consideration that these values are typical values for a qualitative analysis.

Table 2. Overview of detecting with real data P=4

		Game situations			
		incoming ball	acceleration	collision	chasing a ball
Detecting method	Peak current[A]	1.9	1.5	1.5	1.4
	Differentiation[A/s]	50	40	60	50
	Integrated current[A.s]	0.058	0.058	0.046	0.115

Based on this table we can draw some conclusions about the detection methods. The differentiation method prove to be useless in reality: all the values are in the same range of magnitude. With the peak current method only an incoming ball can be detected. The integrated current method is sensitive enough to sense a ball even in the chased scenario. This is a major improvement with regards to previous implementations, so this method is the preferred one to be further refined and explored.

To improve the detection, the PID settings of the controller that keeps the kicker in place had to be adjusted. To better distinguish the incoming ball situation from the acceleration situation the P setting of the controller was adjusted from P=4 to P=15. The idea behind this is that the torque generated by the inertia of the kicker is not influenced by the PID settings , it only depends on the acceleration. When the P action of the controller is set firmer it will bring the kicker faster in a balanced position when accelerating so the integrated value is reset quicker. The current will be the same as the torque generated by the acceleration is constant. For the incoming ball scenario a higher P-action of the controller means it will be decelerated faster when it hits the kicker so there will be a bigger torque generated. In theory, the integrated value should stay the same because it resembles the energy that the motor has to put into the kicker to stop the ball. In both cases the ball has the same speed and therefore the value should be the same. The detection methods applied to current logs of the robot with the P=15 are given in table 3.

Table 3. Overview of detection with real data P=4

		Game situations			
		incoming ball	acceleration	collision	chasing a ball
Detecting method	Peak current [A]	4	1.4	2.5	1.8
	Differentiation [A/s]	140	20	110	40
	Integrated current [A.s]	0.092	0.035	0.058	0.069

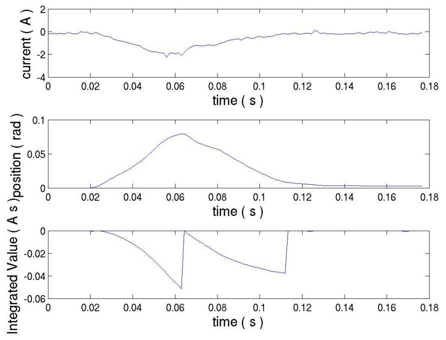


Fig. 12. Receiving soft ball

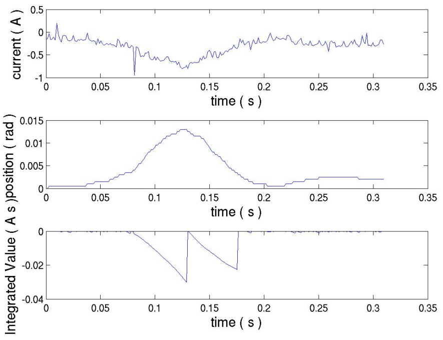


Fig. 13. Acceleration

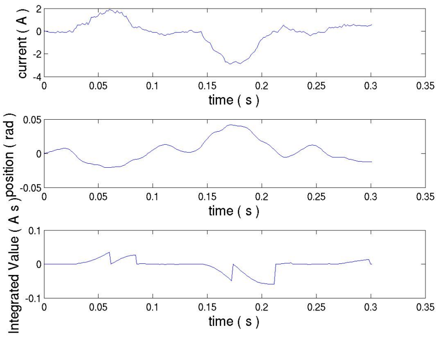


Fig. 14. collision with an opponent

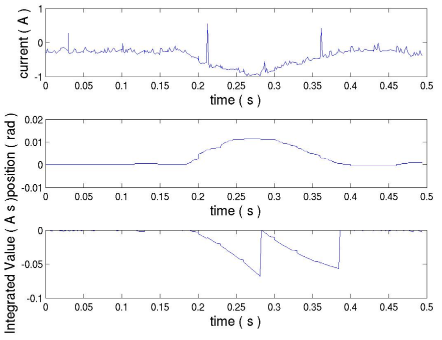


Fig. 15. Chasing a ball

Here in you can see that the integrated value of the ball hitting the kicker increase when the P component of a PID control is higher. If we analyze the values in this table it's clear that the same conclusions can drawn for the peak current and for the differentiation method as the former previous PID configuration. The big difference is the integration method. For the new P both incoming ball and chasing-a-ball game situations are well distinguish from the other two non ball situations. Only a hard collision could be misleading and therefore confused with a chasing of a ball situation. Some logs of the different situations with the hard PID-settings are given in Fig.12 to Fig.15. In these figures you can see the efective action of the two current integration reset conditions in the different game situations.

7 Conclusions and Future Work

The paper has presented the development and validation of ball sensing and detection method that uses high rate and low latency measurements of the torque and energy in motor as well as the angular position of the kicker leg. This is integrated in the embedded motor control of the kicker device allowing it to detect the ball before it starts to move away from the kicker, so that the kick instant can be optimized. A set of game situations that must be distinguished by the system were hereby identified. Several detection methods were proposed and tested both in simulation, with a kicker model built for that propose, and with the robot. The current integration method, that provides an energy like measurement, shown to be superior to the other tested methods, is the only method that has the ability to detect an incoming ball when the robot is standing still and when the robot chases a ball. With the higher P-value in PID the detection value for chasing a ball is close to the one of a collision, and that could lead to some wrong detection of the ball. For that we propose some solutions. The first one is to use different PID settings for different situations. In the receiving ball we can use a more hard settings and for chasing a ball use a more soft setting. Another orthogonal solution would be to integrate information from the accelerometer of the robot in the detection. When the robot has a collision with something this would result in a big acceleration or deceleration. The accelerometer module is connected to the same CAN bus where the embedded kicker control system is also connected to. When a big deceleration or acceleration is detected the kick sensing mechanism could be temporally shutoff and reactivated when the robot is stable again. To achieve optimal settings more testing should be done with data collected during a game. There is still some room for improvement with the kick itself. Presently, when the ball is detected a kick is immediately performed. It would be better that the kicker keeps moving back after detection and performs the kick when its position is at its maximum without losing contact with the ball. This way the kicker would have an increased contact with the ball during a kick, resulting in a harder kick. Another application is to develop a reception control mode that once the ball is detected it decelerates it.

Acknowledgment. The authors acknowledge the major support given by the ISEP-IPP Institution, by the INESC TEC, to this project. This work is financed by the ERDF - European Regional Development Fund through the COMPETE Programme and by National Funds through the FCT - Portuguese Foundation for Science and Technology within project FCOMP-01-0124-FEDER-022701 and under Perception-Driven Coordinated Multi-Robot Motion Control Project with reference PTDC/EEA-CRO/100692/2008.

References

1. Mathworks - simulink,
   `http://www.mathworks.com/products/simulink/`
2. Mathworks - sptool,
   `http://www.mathworks.com/help/toolbox/signal/ug/f0-320.html`

3. Robocup homepage, `http://www.robocup.org/`
4. Warmerdam, T.P.H., Peijnenburg, A.T.A.: Philips cft robocup team description. In: RoboCup 2002: Robot Soccer World Cup VI (2002)
5. Cameron, D., Jahshan, D., George, D.: Mu-wallabies 2003 team description. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003: Robot Soccer World Cup VII on CD, vol. 3020, XVI 767 p. Springer, Berlim (2003, 2004) ISBN: 3-540-22443-2
6. de Best, J., van de Molengraft, R., Steinbuch, M.: A novel ball handling mechanism for the robocup middle size league. Mechatronics 21(2), 469–478 (2011)
7. Dirkx, B.: Philips cft robocup team description. In: Nardi., et al. (eds.) RoboCup 2004: Robot Soccer World Cup VIII. LNCS (LNAI), vol. 3276. Springer, Berlim (2004)
8. Monteiro, J., Moutinho, I., Silva, P., Silva, V., Ribeiro, F., Braga, P.: Minho robot football team for 2003. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003: Robot Soccer World Cup VII on CD, vol. 3020, XVI 767 p. Springer, Berlim (2003, 2004) ISBN: 3-540-22443-2
9. Janssen, R.J.M., Meessen, K.J., de Best, J.J.T.H., Bruijnen, D.J.H., Naus, G.J.L., Aangenent, W.H.T.M., van den Berg, R.B.M., van de Loo, H.C.T., Heldens, G.M., Vugts, R.P.A., Harkema, G.A., van Brakel, P.E.J., Bukkums, B.H.M., Soetens, R.P.T., Merry, R.J.E., van de Molengraft, M.J.G., Kanters, F.M.W., Hoogendijk, R.: Tech united eindhoven team description 2011 (2011),
`www.techunited.nl/media/files/teamdescriptionpaper2011.pdf`
10. John, M., Miller, F.P., Vandome, A.F.: Dribbling. VDM Verlag Dr. Mueller e.K. (2011)
11. Hoogendijk, R.: Design of a ball handling mechanism for robocup. Master's thesis, Technische Universiteit Eindhoven (2007)
12. Dias, A., Silva, H., Almeida, C., Dias, N., Lima, L., Santos, T., Costa, I., Almeida, J., Martins, A., Silva, E.: Iseporto robotic soccer team for robocup 2010: Improving defence and dynamic passing. In: RoboCup, Istanbul (2011)
13. Nassiraei, A.A.F., Takemura, Y., Sanada, A., Kitazumi, Y., Ogawa, Y., Godler, I., Ishii, K., Miyamoto, H., Ghaderi, A.: Concept of mechatronics modular design for an autonomous mobile soccer robot. In: Proc. Int. Symp. Computational Intelligence in Robotics and Automation, CIRA 2007, pp. 178–183 (2007)
14. Pinheiro, P., Costelha, H., Neto, G., Pires, V., Arroz, M., Vecht, B., Lima, P., Custodio, L.: Isocrob 2004: Team description paper. In: Robocup (2004)
15. Bukkems, B.H.M., Kanters, F.M.W., Meessen, K.J., Willems, J.J.P.A., Merry, R.J.E., van de Molengraft, M.J.G., Aangenent, W.H.T.M., de Best, J.J.T.H.: Tech united eindhoven team description 2009 (2009),
`http://www.techunited.nl/media/`
`files/teamdescriptionpaper2009.pdf`

Cooperative Global Tracking Using Multiple Sensors

Roman Marchant, Pablo Guerrero, and Javier Ruiz-del-Solar

Department of Electrical Engineering,
Universidad de Chile,
Avenida Tupper 2007, Casilla 412-3, Santiago, Chile
`{romarcha,pguerrer,jruizd}@ing.uchile.cl`
`http://www.die.uchile.cl/`

Abstract. Multi-robot systems are increasingly present in nowadays applications. In order to allow an effective decision making, a reliable world representation is required. In a team of robots performing a given task, it is beneficial to share information about the world. In this work, a multi-object, multi-sensor and cooperative tracking method is proposed for the Robocup Standard Platform League (SPL), where two teams of humanoid robots play soccer against each other. Each robot is equipped with two low-cost, noisy, and narrowed-field-of-view cameras and two noisy sonar sensors. In addition, they are endowed with a wireless communication hardware. The on-board computer is a low capacity processing unit (x86 500[Mhz]). The proposed tracking system uses all these hardware elements and it is distributed, in the sense that it is executed in every robot. The proposed tracking system is validated in simulations and in real experiments. Main results show an important improvement on simulated and real results when tracking mobile-objects.

Keywords: Multi-robot tracking, Kalman Filter.

1 Introduction

Modeling a dynamic environment, i.e. determining the spacial location of the objects of interest, is one of the central challenges in mobile robotics. The existence of errors in the model of the environment may lead to mistaken actions. Therefore, the reliability of this model is of vital importance for making correct decisions.

Multi-robot systems are increasingly present in real applications. For instance, mining, agriculture and human-care require that multiple robots share information in order to succeed. Due to the small portion of the environment perceived by every robot, sharing sensory information may allow to expand largely the accessible portion of the environment.

The here-addressed problem is the multi-robot estimation of the state of multiple objects in a dynamic environment. The focus of this work is to define a cooperative tracking methodology and a matching algorithm. The complexity of this problem depends strongly on the application being addressed, because the environment and the robot hardware determine how hard the tracking of multiple objects in a particular scene is. The most significant aspects of the environment that determine the complexity of the problem are the number of objects, their velocity variability and the size of the arena. While the robot hardware variables that affect are the range, precision and field-of-view of the robot sensory hardware, as well as the computational capabilities of the robot. In addition, the precision of the robot movements are important for odometry calculation.

In this work, a multi-object, multi-sensor and cooperative tracking method is proposed for a Robocup *Standard Platform League* (SPL) environment. The estimation of the state of mutiple objects (in this case the poses of the robots and the ball) is of central importance in the SPL since it allows the existence of complex behaviors such as making passes that consider the positions of the partners and opponents, avoiding robots that are not being currently perceived. The proposed tracking system is distributed, in the sense that it is executed in every robot.

The main contribution of this paper is that it proposes a cooperative multi-object, multi-sensor and high-rate tracking methodology for a real application, were the robots have a low capacity processing unit.

This paper is organized as follows: First, in section 2, some related work is reviewed. Then, the proposed methodology for tracking mobile objects is described on section 3. Section 4 presents experimental results on a simulated and a real robot. Finally, section 5 draws some conclusions and recommends future work.

2 Related Work

In the past decades, vast works have addressed cooperative state estimation in multi-robot systems. Those works have been implemented on a wide variety of applications, such as UAVs [1, 2], robot soccer [3–9], water vehicles [10] and general purpose robotics [11–13].

Regarding cooperative state estimation, it is possible to differentiate between two main areas of research. The first one is the cooperative tracking of one or more interesting objects in a scene [1–5, 7–9, 11–13], and the second area is focused on improving the localization of a mobile robot by adding other robots perceptions as inputs to the localization module [3, 6, 10].

Our work is very similar to [9], which explores deeply the areas of coordination and cooperation in multi-robot systems. Particularly on the mobile-object cooperative-tracking area, a classical Kalman filter approach is used. Problems such as measurement delay, distributed implementation, clock synchronization and how external information influence local estimations are addressed in [9]. The main limitation of that system is that it is designed and tested for an omni-directional perception which greatly simplifies the matching problem.

A similar approach is addressed by [4] and [8]. In these works, robots and obstacles are detected using a laser range finder. On the other hand, the ball is detected using visual perceptions. These works present the idea of multi-object tracking using a *Kalman Filter* for every object on the scene, although results are only focused on the ball tracking and no results are presented for robot tracking. The main drawback is the use of a central computer that makes an estimation of every robot and then sends the fused information back to all team members.

The main difference of the here-proposed global tracking methodology is that it is designed and implemented in an humanoid robot soccer platform, were narrowed field-of-view cameras are used, which reduces the number of perceptions to any object. Furthermore, it merges the information from sonars, cameras and other robots in a distributed manner. Another important difference is that in this work an evaluation of the proposed methodologies, using a laser-based ground-truth system, is presented. Moreover, a quantitative comparison between different tracking approaches is shown.

3 Global Tracking Methodology

3.1 Framework

The methodology is designed to operate in a robot soccer environment, although it may be generalized to other environments. Teams of soccer robots have a group goal, therefore the chosen application exploits the need of cooperation between robots. The teams in the SPL use Nao humanoid robots [14], which are equipped with two noisy sonar sensors and two low cost cameras. Narrowed-field-of-view cameras perceive objects with low frequency, therefore the sonar sensors are used as a supporting feature when no camera perceptions are available for an object.

In robot soccer as in most common situations, a global static origin, $\boldsymbol{O}$, may be defined. Several objects are present and some of them may be used as landmarks to infer the auto-localization, such as goals, lines and corners among others.

Objects may be classified depending on their pose behavior through time relative to a fixed coordinate system [15]. *Fixed objects* are those whose kinematic state (KS) is constant through time (e.g. goals, lines, corners), whereas *mobile objects* have a variable KS through time. *Mobile objects* may be classified into *passive* or *active* depending on the source that determines their KS variations. *Passive objects* change their KS only due to actions executed by other objects (e.g. ball). On the other hand, *active-objects* have KS changes determined by their own actions. Finally, *active objects* may be classified into *partner* or *non-partner*. Partner objects share information (e.g. teammates) while non-partner robots do not (e.g. opponents). In the robot soccer application, objects move on a two dimensional plane, and the kinematic state (KS) of an object is defined as a vector that contains its pose, $\boldsymbol{k} = (x, y, \theta)^T$. (x, y) is the position of the object relative to $\boldsymbol{O}$ and θ is the orientation of the object relative to $\boldsymbol{O}$.

The control software in each robot must solve a complex problem and run in real time. Given the scarce perceptual information, the low availability of computational resources, and the requirement of a minimum frame rate (30fps), the image processing algorithms cannot be as complex as the state of the art suggests. In order to achieve a high-level goal, several issues must be previously solved. Therefore, a software architecture consisting of four modules is implemented in each robot (Actuation, Decision Making, Perception and World Modelling). A detailed description of the here-relevant modules, Perception and World Modelling, is presented in the following paragraphs, while the connections between this two modules are illustrated in Fig. 1.

Perception

This module processes information coming from the sensors of the robot. The visual-perception sub-module classifies color pixels and then groups them into color blobs. Then, the objects are detected based on a set of rules applied to the blobs. For every visually detected object, a Gaussian PDF, $\boldsymbol{z}$, is generated for the pose of every object (containing the orientation only when necessary) relative to the observer robot. The mean of $\boldsymbol{z}$, $\overline{\boldsymbol{z}}$, is the calculated position using geometrical characteristics of the blobs and the 3D-pose of the camera. The covariance matrix Σ is calculated using previously obtained statistics. The perceived objects PDFs $\{\boldsymbol{Z}\}$ are partitioned into *fixed* $\{\boldsymbol{z}_f\}$, *passive* $\{\boldsymbol{z}_p\}$ and *active* $\{\boldsymbol{z}_a\}$ perceived objects.

The sonar-perception sub-module receives a set of up to nine measurements per sonar. Each value represents the Euclidean distance, d_i^r, between the observer robot and an i-th unknown object candidate. An unidimensional PDF, w, is generated for each measurement. The mean, $\overline{w}$, is the measured distance d_i^r, and the variance, δ, is previously obtained by an off-line statistic analysis. Additionally, the sonar ID w^{id} is included for each PDF ($w^{id} = 0$ for right sonar and $w^{id} = 1$ for left).

Finally, $\{\boldsymbol{z}_f\}, \{\boldsymbol{z}_p\}, \{\boldsymbol{z}_a\}, \{w\}$ are transmitted to the World-Modelling module, as shown in Fig. 1.

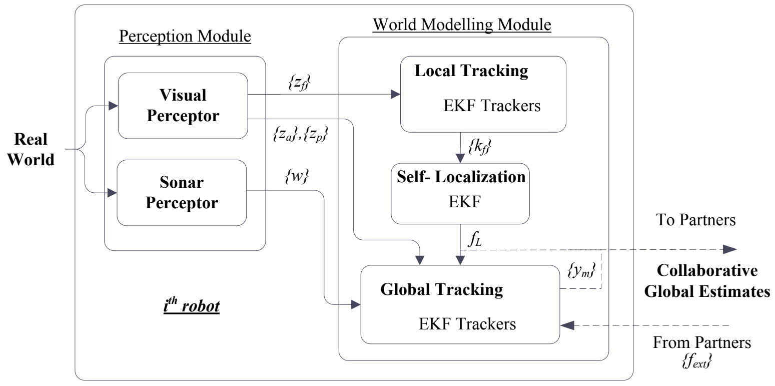


Fig. 1. Perception and World Modelling connection details

World Modelling

This module estimates the KS of the objects in the environment using information provided every time-step by the Perception module. Due to the noise and possible outliers generated by the Perception module, the World-Modelling module implements filtering and processing stages in order to achieve a precise representation of the KS of the surrounding objects.

The KS of an object is the mean of a Gaussian PDF, which is estimated for every object using a classical *Extended Kalman Filter* (EKF) as a tracker. The state variable $\boldsymbol{x}$ is the KS $\boldsymbol{k}$ of the object. The process model, f, its variance, R, the observational model, h, and its variance, Q, are defined for each tracker depending on the object type and observation source.

The World-Modelling module processes each received perception in a fashion that depends on its type. The $\{\boldsymbol{z}_f\}$ PDFs are delivered to the *local tracking* sub-module that implements a tracker for each object present in an a-priori-known map. The predictive stage for each tracker uses the odometry of the robot to move the estimated PDFs. The corrective stage uses the mean of $\boldsymbol{z}_f$ as an observation and Σ as the variance for a zero mean Gaussian PDF of the observational model.

The Self-Localization sub-module uses the KS estimations of the *fixed objects*, $\{\boldsymbol{k}_f\}$ (goals, lines and corners among others), to estimate recursively a Gaussian PDF ($\boldsymbol{f}_L$) of the current KS of the robot using an EKF-based self-localization algorithm. This localization estimation is relative to $\boldsymbol{O}$.

The global tracking sub-module uses $\{\boldsymbol{z}_a\}$, $\{\boldsymbol{z}_p\}$, $\{w\}$, $\boldsymbol{f}_L$ and the external estimated PDFs of the other robots mobile objects, $\{\boldsymbol{f}_{ext}\}$, as information sources.

Let us define the estimated PDFs of all mobile objects generated by the global tracking methodology as $\{\boldsymbol{y}_m\}$. In addition, an expression is defined for active, $\{\boldsymbol{y}_a\}$, and passive, $\{\boldsymbol{y}_p\}$, objects. These estimations are relative to $\boldsymbol{O}$.

Finally, the World-Modelling module transmits the estimated PDFs, relative to $\boldsymbol{O}$, of all mobile objects $\{\boldsymbol{y}_m\} = \{\boldsymbol{y}_a\} \cup \{\boldsymbol{y}_p\} \cup \{\boldsymbol{f}_L\}$ to the Decision-Making module and other robot's World-Modelling module.

The tracked objects KS is estimated using the methodology detailed in the following section.

3.2 Cooperative Global Tracking Methodology

The here-described methodology allows the tracking of multiple objects using multi-sensor ($\{\boldsymbol{z}_p\}$, $\{\boldsymbol{z}_a\}$, $\{w\}$) and cooperative $\{f_{ext}\}$ information.

A filtered PDF, y_m, referenced to O, is estimated for each mobile object using an EKF as a tracker. The maximum number of trackers, N, is previously determined according to a-priori knowledge of objects on the scene. In this particular framework, there are two types of mobile objects: N_p partner and N_o opponent robots, therefore $N = N_p + N_o$. Initially, all trackers are deactivated and linked to a particular object with an unique ID. They are activated or deactivated on each time-step depending on a covariance value threshold. In every time-step, a predictive stage is executed for each active *global-tracker*, the predictive stage has no inputs since odometries are not directly received from other robots. However, $Q \neq 0$, therefore a fixed amount of uncertainty is added over y_m. Q is approximated as a function of the tipical average velocity of the *mobile objects*. If any perceptual PDF is matched to a tracker, a corrective stage is executed (The details of the matching procedure are presented on section 3.3). But $\{\boldsymbol{z}_p\}$, $\{\boldsymbol{z}_a\}$, $\{w\}$ are initially referenced to the observer robot, therefore, as the information needs to be coherent with partner estimated PDFs, a reference system transformation to $\boldsymbol{O}$ is needed (see block diagram in Fig 2). This transformation is applied to each perception using f_L. The resulting PDFs $\{\boldsymbol{z}'_p\}$, $\{\boldsymbol{z}'_a\}$, $\{w'\}$ are referenced to O.

Depending on the type of perceptual PDF, a different observational model function must be used on the corrective stage of the EKF. This is mainly because each type of sensor delivers data with different dimensionality. The output of the proposed methodology is the estimated KS, $\{\boldsymbol{k}_m\}$, of present mobile objects. For each object, $\boldsymbol{k}_m = \overline{\boldsymbol{y}_m}$, where $\overline{\boldsymbol{y}_m}$ is the mean of the estimated PDF $\boldsymbol{y}_m$. Although sonar measurements are generated with a higher frequency than visual perceptions, the absence in the angle information may lead to incorrect estimations. This will affect the trajectory and probably increase error because after a corrective stage of the EKF which only uses sonar information, the state mean will only be corrected in its radial component. This effect is presented in Fig. 3.

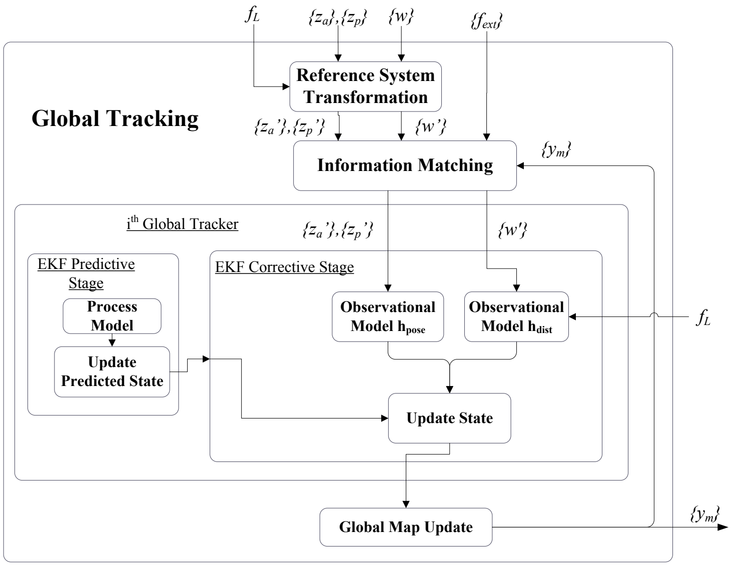


Fig. 2. Global Tracking Methodology Block-Diagram

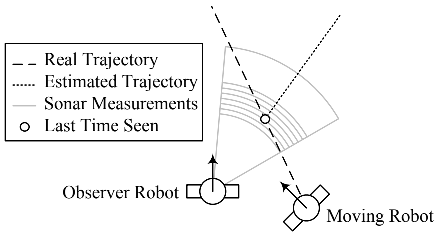


Fig. 3. Trajectory deformation effect produced by sonars

3.3 Matching Procedure

For every new perception, the tracker that is most likely to correspond to the received perception is selected in order to be updated. Perceptions are treated differently depending on their type. The output of this stage is the index of the *global-tracker* that best matches every input, or (-1) if the input is not associated to any tracker. All trackers are initially deactivated and they get activated when a visual perception or an external estimation cannot be associated to any active tracker.

Sonar Perception

When a *sonar-perception* PDF arrives, trackers whose state is outside the sonar range are not considered. For those active trackers whose state is inside the sonar range, an Euclidean distance measurement is used for deciding the index of the tracker associated to that particular input. If all trackers are deactivated or too far from the measurement (determined by the Euclidian distance threshold, γ), the sonar measurement is not associated to any tracker because there is no evidence that the measurement corresponds to an *active-mobile-object*. The matching algorithm for this type of perception is presented on Algorithm 1.

Algorithm 1. Matching Algorithm for Sonars

```
 1: Let $w$ be a sonar source PDF relative to $O$
 2: Let $\gamma$ be the Euclidian association threshold
 3: Trackers outside sonar range are filtered
 4: for active trackers in sonar range do
 5:    Calculate tracker relative pose $rP$
 6:    Calculate Euclidean distance $D_{ot}$ observer-tracker
 7: end for
 8: if $\min\{D_{ot}\} > \gamma$ then
 9:    return -1
10: else
11:    return Index of $\min D_{ot}$ tracker
12: end if
```

Visual Perception and External Estimations

This type of data are always associated to a tracker, except when all trackers are active and the Mahalanobis distance to each one of them is greater than the Mahalanobis distance threshold ξ. This is usually the case when an false detection is occasionally perceived. The matching algorithm for this type of perception is presented on Algorithm 2.

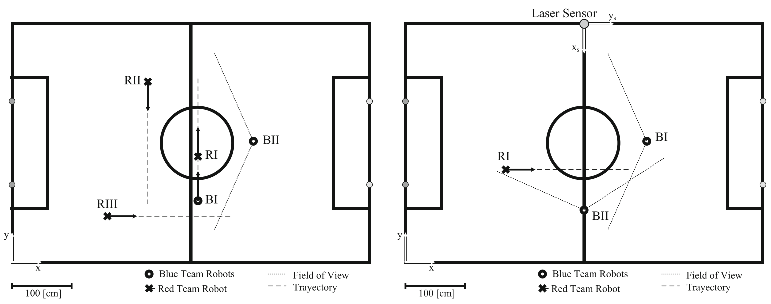
Algorithm 2. Matching Algorithm for Visual and External Estimates

```
1: Let $z_m$ be a *mobile-object* visual source PDF relative to $O$
2: Let $f_{ext}$ be an external source PDF relative to $O$
3: Let $\xi$ be the Mahalanobis association threshold
4: Trackers are filtered depending on object class
5: for active trackers of interesting class do
6:    if tracker is active then
7:       Calculate Mahalanobis distance $DM_{ot}$ observer-tracker
8:    end if
9: end for
10: if $\min\{DM_{ot}\} > \xi$ then
11:    if any tracker deactivated then
12:       return deactivated tracker index
13:    else
14:       return -1
15:    end if
16: else
17:    return Index of $\min DM_{ot}$ tracker
18: end if
```

4 Experiments

4.1 Experimental Setup

This section describes simulated and real experiments conducted on a SPL robot soccer environment. The first experiment uses a 2D simulator, which generates a time labelled database containing noisy robot perceptions, odometries and ground-truth data. The performance of the proposed system is evaluated on the same database but with different parameters. The experimental setup for the first experiment consists of five robots located on a simulated SPL soccer field. There are two teams of robots, the blue team and the red team. Three robots belong to the red team (RI, RII, and RIII) and two robots belong to the blue team (BI and BII). The initial position of the robots, their IDs and their ideal trajectories are illustrated in Fig. 4a. The idea of this experiment is that the blue robots BI and BII perceive the red team robots, and communicate their estimations between each other. The second experiment is executed using real robots on a SPL soccer field. The experimental setup is simpler than the simulated one due to operational complexity. In this experiment, the setup consists of two robots belonging to the blue team (BI and BII) and one robot belonging to the red team (RI). They are positioned on the field as illustrated in Fig. 4b. The ground-truth data is provided by a laser-based ground-truth system like the one proposed in [16], running on-line on an external computer. A time-labeled database is generated as in the simulation experiment.

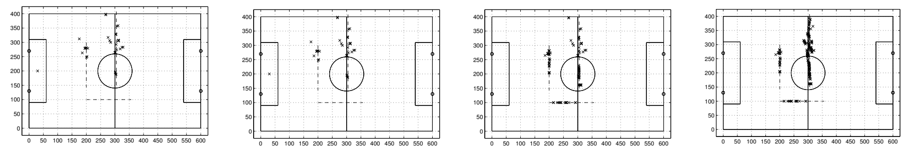


(a) Simulated Experiment (b) Real Experiment

Fig. 4. Experimental setup of robots in simulated and a real experiments

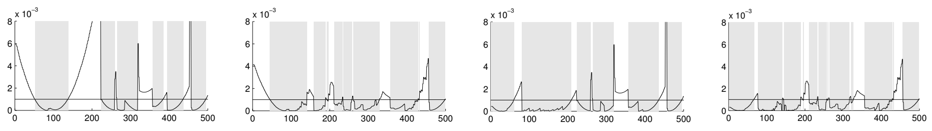
4.2 Results

The state-estimation error of the whole environment is calculated on both experiments every time-step for the moving robot BI, using four different configurations: (i) using only visual data, (ii) using visual and ultrasonic data, (iii) using cooperative estimations and visual data, and (iv) using cooperative estimations, visual and ultrasonic data. The estimated poses and expected trajectories of each tracker in all situations are presented in Fig. 5 for the simulated experiment and in Fig. 7 for the real experiment. The position error for tracker i at time k, denoted as e_k^i, is obtained by evaluating the difference between the estimated-PDF mean, $\boldsymbol{\mu}_k$, and the ground-truth position, $\boldsymbol{gt}_k$. The squared error ${e_k^i}^2$ is then calculated and normalized by the arena size $A_{size} = (600 * 400)[cm^2]$. The normalized squared error $\overline{{e_k^i}^2}$ for tracker III is plotted in Fig. 6 for the simulated experiment and in Fig. 8 for the real experiment. The normalized squared error is evaluated for the four different configurations (i to iv). The level of error at time k determines if the information is considered as useful or not. An arbitrary threshold ε, so-called decision-threshold, has been defined for this particular application with the value $(15[cm])^2, \overline{\varepsilon} = 0.001$ in the normalized space. The value of this threshold is the half of the max diameter of the tracked objects. A percentage of time when the information is considered as useful is calculated for each tracker and method. A table is generated with this values and is presented in Table 1 for the simulated experiment and in Table 2 for the real robot experiment. Furthermore, a gray background in Figures 6 and 8 represent visually when information is useful for robot decisions.Tables 1 and 2 show that the use of both ultrasonic and collaborative information is in all the analized cases the most reliable choice. Using these additional sources of information , the percentage of time in which the error is below $\overline{\varepsilon}$ grows between 20% and 40% in the simulated experiment and around 25% in the real experiment, from the case with only visual information. Additionally, the use of colaborative information is always positive, almost to the extent of the best case. The utility of the sonar information is not clear. While in table 1 it always appears to help, in table 2 it makes he results even worse than the case with only visual information.



(a) Only visual data (b) Visual and sonar data (c) Cooperative and visual data (d) Cooperative, visual and sonar data

Fig. 5. Results of the simulated experiment on a SPL reference system. Segmented lines indicate the ground-truth trajectory for each red-team robot. Crosses determine an estimated position for each time-step.

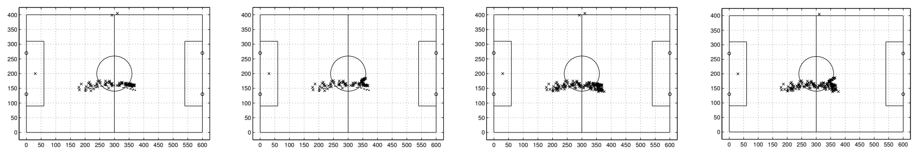


(a) Only visual data (b) Visual and sonar data (c) Cooperative and visual data (d) Cooperative, visual and sonar data

Fig. 6. Evaluation of the simulated experiment. Normalized error of the pose estimation of robot RI using different approaches (see subsection 4.2). Horizontal line represents decision-threshold $\overline{\varepsilon}$, and gray areas show when error is lower than $\overline{\varepsilon}$

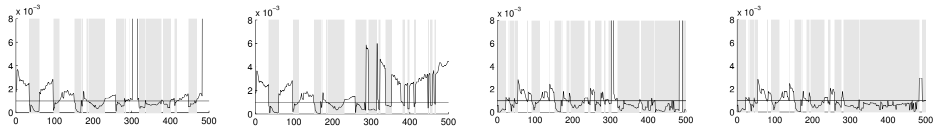
Table 1. Percentage of time when error is below $\overline{\varepsilon}$ for the simulated experiment. The different approaches (i), (ii), (iii) and (iv) are described in subsection 4.2

	(i)	(ii)	(iii)	(iv)
Opponent 1	18%	18%	62%	62%
Opponent 2	-	-	64%	64%
Opponent 3	48%	61%	67%	68%
$\overline{X}$	22%	26%	64%	65%



(a) Only visual data (b) Visual and sonar data (c) Cooperative and visual data (d) Cooperative, visual and sonar data

Fig. 7. Results of the real experiment on a SPL reference system. Segmented lines indicate the ground-truth trajectory for moving robot RI. Crosses determine an estimated position for each time-step.



(a) Only visual data (b) Visual and sonar data (c) Cooperative and visual data (d) Cooperative, visual and sonar data

Fig. 8. Evaluation of the real experiment. Normalized error of the pose estimation of robot RI using different approaches (see subsection 4.2). Horizontal line represents decision-threshold $\overline{\varepsilon}$, and gray areas show when error is lower than $\overline{\varepsilon}$.

Table 2. Percentage of time when error is below $\overline{\varepsilon}$ for the real robots experiment. The different approaches (i), (ii), (iii) and (iv) are described in subsection 4.2.

	(i)	(ii)	(iii)	(iv)
Opponent 1	45%	37%	63%	69%

5 Conclusion

A method for cooperatively tracking multiple objects using multi-sensory information was described and tested in a multi-robot application. The results show that the error of the kinematic state estimation of *mobile objects* decreases importantly when using information provided by cooperative robots, and decreases even more when using multi-sensorial information. In addition, the percentage of time when the estimation error is lower than an acceptable decision-threshold increases significantly (between 20% and 40% in the simulated experiment and around 25% in the real experiment) when using cooperative and sensor fusion techniques. As a future work, we expect to include velocity estimations to reduce sonar trajectory distortions and manage multiple tracker hypothesis for each object allowing multi-modal distributions.

Acknowledgment. This research was partially supported by FONDECYT (Chile) under Project Number 1090250.

References

1. Wheeler, M., Schrick, B., Whitacre, W., Campbell, M., Rysdyk, R., Wise, R.: Cooperative tracking of moving targets by a team of autonomous uavs. In: 2006 IEEE/AIAA 25th Digital Avionics Systems Conference, pp. 1–9 (October 2006)
2. Campbell, M., Whitacre, W.: Cooperative tracking using vision measurements on seascan uavs. IEEE Transactions on Control Systems Technology 15(4), 613–626 (2007)
3. Cánovas, J.P., LeBlanc, K., Saffiotti, A.: Robust multi-robot object localization using fuzzy logic. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 247–261. Springer, Heidelberg (2005)

4. Dietl, M., Gutmann, J.-S., Nebel, B.: CS freiburg: Global view by cooperative sensing. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 133–143. Springer, Heidelberg (2002)
5. Karol, A., Williams, M.-A.: Distributed sensor fusion for object tracking. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 504–511. Springer, Heidelberg (2006)
6. Liu, Z., Zhao, M., Shi, Z., Xu, W.: Multi-robot cooperative localization through collaborative visual object tracking. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007. LNCS (LNAI), vol. 5001, pp. 41–52. Springer, Heidelberg (2008)
7. Nisticò, W., Hebbel, M., Kerkhof, T., Zarges, C.: Cooperative visual tracking in a team of autonomous mobile robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 146–157. Springer, Heidelberg (2007)
8. Silva, J., Lau, N., Rodrigues, J., Azevedo, J., Neves, A.: Sensor and information fusion applied to a robotic soccer team. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 366–377. Springer, Heidelberg (2010)
9. Pagello, E., D'Angelo, A., Menegatti, E.: Cooperation issues and distributed sensing for multirobot systems. Proceedings of the IEEE 94(7), 1370–1383 (2006)
10. Moratuwage, M., Wijesoma, W., Kalyan, B., Patrikalakis, N., Moghadam, P.: Collaborative multi-vehicle localization and mapping in high clutter environments. In: 2010 11th International Conference on Control Automation Robotics Vision (ICARCV), pp. 1422–1427 (2010)
11. Schmitt, T., Hanek, R., Beetz, M., Buck, S., Radig, B.: Cooperative probabilistic state estimation for vision-based autonomous mobile robots. IEEE Transactions on Robotics and Automation 18(5), 670–684 (2002)
12. Dietl, M., Gutmann, J.-S., Nebel, B.: Cooperative sensing in dynamic environments. In: Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1706–1713 (2001)
13. Shuqin, L., Le, Z., Xiaohua, Y.: Design and implementation of multi-robot cooperative tracking. In: 2010 14th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 490–494 (April 2010)
14. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of nao humanoid. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 769–774 (2009)
15. Coltin, B., Liemhetcharat, S., Meriç Andli, C., Tay, J., Veloso, M.: Multi-humanoid world modeling in standard platform robot soccer. In: 2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 424–429 (2010)
16. Marchant, R., Guerrero, P., del Solar, J.R.: A portable ground-truth system based on a laser sensor. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 234–245. Springer, Heidelberg (2012)

Implementing a Real-Time Hough Transform on a Mobile Robot

John Morrison, Eric Chown, and Bill Silver

Bowdoin College

Abstract. Robotic vision is a challenging problem due both to the uncertain nature of real-world environments and the computational constraints of mobile platforms. Many standard computer vision algorithms have high computational requirements and are often seen as unsuitable for use in an embedded system such as the Aldebaran Nao. Many current approaches use fragile algorithms and reduced resolutions to achieve a low processing latency. We implement the Hough Transform, a standard line detection algorithm, for real-time use in the RoboCup Standard Platform League. Using assembly language instructions to expose the processor's full potential, this project implements a real-time Hough transform for line detection on 320x240 pixel images.

1 Introduction

The primary source of perceptual information for the robots in the RoboCup Standard Platform League (SPL) [6] is their camera, forcing real-time vision processing to become a topic of intense research and development. The SPL environment and platform offer a particular set of constraints which challenge any potential vision system, such as a relatively slow (by desktop standards) processor, a narrow field of view, unpredictable images, partially obscured objects, and limited processing time per frame. The output of the vision system is the foundation for higher level cognition and skilled soccer playing.

The current state of the art in RoboCup vision systems, mostly based on color segmentation and color run-length encoding, is ad-hoc and tied tightly to the RoboCup environment. The methods described by the top competitors in the league [3,9,8] for identifying object shapes in the image are largely collections of hand coded heuristics for what defines a "goal post" or a "line" as a human would describe it. Domain information about the specific objects in question is, of course, required for reliable detection, but the goal of this project is to show that general pattern recognition techniques are a viable option for SPL vision systems.

Computer vision research has produced many pattern recognition algorithms which are more reliable, more robust, and more general than these hand designed region building heuristics. Unforunately, there has not been sufficient research yet to optimize these algorithms for the SPL. This is why teams in have been hesitant to use them in competition.

To put the performance requirements of a real-time vision system in context, consider that with a 640x480 YUV 422 image, there are 614,400 values to be processed. So, allocating 20 ms for vision processing there are only 32 nanoseconds of processing time available per value. With a 500 MHz processor, like the Nao's Geode, that means there are only 16 machine cycles available per value. For comparison, a single integer divide can take 24 cycles, and one floating point arctangent can take up to 354 cycles [1]. It is with these constraints in mind that this project was undertaken.

The first step of the new vision processing is edge detection. Edge detection provides the points of interest which will be used in the Hough transform for line detection. After the edge detection, the edge points are processed with the Hough transform. The Hough transform works through a voting mechanism. It tallies the "votes" of each edge point for a particular set of possible lines in the image. The voting mechanism makes the algorithm robust, yet straightforward.

This project aims to establish the Hough transform as a reliable and applicable vision algorithm for the embedded RoboCup Nao platform. The Hough transform has been used infrequently [8, 128] in work by other teams, but when it was, it was often in a limited or prohibitively slow manner. The Hough transform is an algorithm which suits the highly uncertain RoboCup environment well, and, when properly implemented, is also acceptable for full image processing on an embedded platform.

We seek a general algorithm for pragmatic reasons. General vision algorithms are robust in various circumstances and can be re-purposed and reused in new situations. Many current SPL vision systems tightly integrate the steps of object detection. When the rules change, or the environment is altered, the ad-hoc system can require extensive revising to be adapted. A good general algorithm will be easily adaptable. The current SPL vision solutions maintain high frame rates by skipping lots of pixels, sampling the images down to paltry 160x120 or below. This loss of resolution degrades accuracy and can cause mis-detections because of insufficient visual evidence.

2 Assembly Language

In order to achieve the desired real-time efficiency of this project, we wrote the majority of the code for this project in x86 assembly language. Assembly language can sometimes be the only way to truly squeeze every bit of efficiency from a CPU. It lets the programmer use the ideal instruction mix for the task, a job a compiler can sometimes struggle to do.

In addition, modern architectures have vector processing instructions which have no easily accessible and embeddable C/C++ equivalent. The vector processing units, known as Single Instruction Multiple Data (SIMD) units, are capable of operating simultaneously on multiple pieces of data, often with special instructions capable of performing complex operations with a single instruction.

The Geode LX processor can operate the x86 MMX instruction set which uses 64-bit registers to operate on multiple 1, 2, or 4 byte values with a single instruction. We make extensive use of vector instructions throughout this project.

3 Edge Detection

Edge detection, as applied here, is the process of finding points of significant and rapid change in the intensity of one or more channels of an image. The physical boundaries between objects on the field cause brightness discontinuities which are captured in an image and are largely independent of lighting, viewpoint, and camera settings. This makes edge points a more reliable measure for object detection than the brightness values themselves.

To strike a balance between computational requirements and accuracy, we chose to use the Sobel operator [5] in the edge detection process. It requires only a single pass over the image, but performs very well as an edge detector. The Sobel Operator is applied to the image's Y channel. Edges in the Y channel provide good boundaries of field lines across lighting conditions.

3.1 Sobel Operator

To detect the edges in images, we first calculate the gradient at every pixel in an image by convolving the image's Y channel with the Sobel operator [5, 147]. The Sobel operator is a pair of kernels as depicted in Table 1, which approximate the (x, y) components of gradient. The magnitudes in x and y are used to compute a magnitude and direction of the gradient vector at each pixel.

To run the Sobel edge detector in real-time with the computational constraints of the Nao, the operation needs to be optimized significantly. The Sobel operator is an ideal candidate for SIMD instructions. G_x and G_y are calculated independently at each pixel. By hand coding in x86 assembly language and using the MMX instructions, we were able to calculate four pixels' gradients in parallel, resulting in a significant speed increase.

Table 1. x,y Sobel Kernels

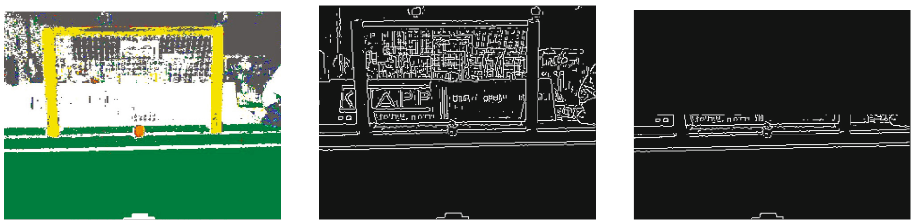
(a) G_x

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

(b) G_y

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

As processor speeds increase, memory accesses and memory bandwidth become a limiting factor in high performance computations. The 3DNow! instruction set provides the prefetch instruction to help lower the memory bottleneck. By bringing soon to be accessed memory into the fastest cache, the programmer can remove some waiting for I/O. By prefetching two rows below the current pixel during the Sobel operator execution, we reduced cache miss penalties. When a new row is accessed, it is already in the L1 cache because the 64 KB L1 cache is large enough for more than four rows of pixels.



(a) The color segmented image (b) Edge detection on the entire image (c) Edge detection only performed below the field horizon

Fig. 1. Reducing edges with the field horizon

3.2 Edge Peak Detection

Once the gradient magnitude and direction is calculated at each pixel, the gradient values need to be compared to each other to find the actual edge points in the image. The edges are found by locating the peaks in the gradient map, the points of greatest change, which are the desired output of the edge detector. The goal of the peak detection phase is to also ensure that only one edge point is found for each real edge point in the image.

Using the gradient direction, an asymmetric peak test is then applied in the gradient direction to all remaining candidate points. The peak test determines if the candidate edge point's gradient magnitude is greater than that of the pixel in the opposite direction of the candidate's gradient, and greater than or equal to the magnitude of the pixel in its gradient direction.

Once a pixel passes the asymmetric peak test, it is labeled an edge peak and its location in the image, along with its gradient direction, are appended to the list of previously found edge peaks. The final list of edge peaks is then sent to the Hough transform to be processed further.

A significant limiting factor in peak detection computation time is the angle calculation. Floating-point calculations are slow and provide more accuracy than needed for our purposes. To remove this bottleneck, we chose to replace the floating point angles with eight-bit binary angles. An eight-bit binary number represents the range 0-255, so a radian angle t can be computed as an eight-bit binary angle with the formula $t_{int} = t_{radian} * \frac{128}{\pi}$. These binary angles are accurate to 1.41 degrees.

The machine level floating point arctangent function is also too slow, taking up to 354 machine cycles [1]. By precomputing an arctangent lookup table, we removed the costly arctangent function call. The table was indexed by the result of $\frac{G_y}{G_x}$. According to the Geode LX Databook, an integer divide of 2 byte-long values can take up to 16 machine cycles, while an integer multiply of similar length values only takes 3 cycles [1]. To take advantage of this, we precomputed another lookup table of $\frac{1}{G_x}$, indexed on G_x.

The power of using assembly language comes in the ability of the programmer to exploit every corner of a machine's instruction set. The x86 instruction set is large and has evolved over time to accommodate many different needs and applications. The advantage of the x86 instruction set's breadth is that it has many optimization opportunities, but they can be hard for a compiler to use. The author of the code has the best idea what the code is intended to do, and the circumstances it will encounter, so he can choose the best instruction mix to accomplish that task. That power has great rewards in the edge detection phase of this project.

4 Hough Transform

The Hough transform was patented in 1962 as a "method and means for recognizing complex patterns in photographs" [4], specifically straight lines, from a set of points. The transform has also since been generalized to find any parameterizable general pattern, [2] but we will only use it here for line finding.

The Hough transform has many properties which make it suitable for line finding in RoboCup. The field lines in RoboCup are uniform, straight lines but are often occluded. The Hough transform deals well with occlusion through the voting mechanism [5]. A line which is partially covered by a ball, or a robot, will still be recognized as crossing through the body of the robot. This is necessary for RoboCup vision as other robots are often obstructing lines.

Without specific attention paid to its implementation, the Hough transform can be memory and processor intensive [8,5] which has limited its adoption in the league in the past. Specific attention was paid during this project to memory access patterns and computation ordering during the Hough voting procedure.

4.1 Voting

The first step in the Hough transform is marking the Hough accumulator space for every peak found during the edge detection. Looping through the list of edge peaks, we increment the $\{r, \theta\}$ accumulator bins which correspond to each possible line which could pass through that edge peak.

A basic implementation of the transform would increment the bin for every line that could possibly pass through that line. For a single edge peak, over 256 bins would need to be incremented, at least one per angle. Filling so many bins would quickly overwhelm a real-time system. An image with 2,000 edge peaks would have to fill many more than 512,000 (and experimentally, likely more than 1,000,000) bins.

However, the gradient angle provides sufficient information to limit that search to within a small error range. The gradient direction approximation computed in previous steps is approximately equal, within an error margin, to the angle of the line, so we can limit the marking in the Hough space to angles which are within a predetermined error margin of the gradient angle. The error margin needed is small, ± 5 was found sufficient for this project. Thus, instead of incrementing bins for 256 angles, only bins corresponding to 10 different angles, representing a span of approximately 14 degrees, are incremented.

While limiting the span of Hough bins to mark is a significant improvement in running time, it is not significant enough, and there is still more time to be recovered. Thus far all the algorithms and implementations have operated sequentially in a single pipeline using only basic integer instructions. The Geode actually has two hardware pipelines, an integer unit (IU) and a floating point unit (FPU). The FPU handles MMX instructions, as well, reusing its 64-bit registers for SIMD commands. With its dual pipelines, the Geode is capable of executing an integer instruction and an MMX instruction every clock tick.

With this in mind, we redesigned the Hough voting procedure to intersperse MMX and integer instructions. By injecting a mix of instructions into the processor, we are able to interleave computations and significantly decrease the necessary run time of the routine.

We set up two pipelines which run simultaneously and communicate using a pair of ping-pong buffers:

1. **MMX/FP Unit:** The MMX registers compute the locations in memory to be incremented.
2. **Integer Unit:** The IU increments the Hough bins at the locations computed by the MMX pipeline.

Table 2. 2x2 Boxcar Kernel

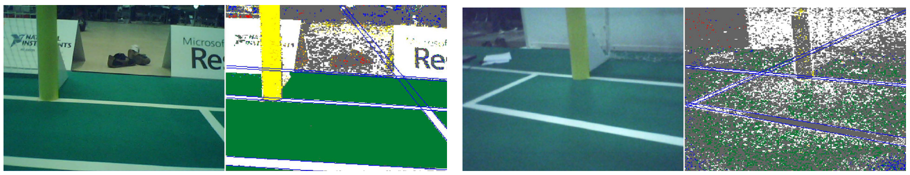
$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

After the voting is complete, we smooth the Hough Space to reduce the noise. The smoothing is completed with a 2x2 "boxcar" kernel, as seen in Table 2. The kernel is applied to each bin in the accumulator by averaging each bin with the bins to the right, to the right and below, and below it. This helps to smooth noise in the Hough space due to errant edge peaks. The boxcar is simple, fast, in place, and can be implemented effeciently using MMX instructions.

After the Hough space has been filled, the bins with the highest values must be found. First the Hough space is scanned for bins with a non-zero value. During the smoothing step, we subtracted a fixed noise threshold and clamped the value at zero. Once a bin with a positive value is found, the eight bins surrounding it are compared with the current bin. If the current bin has a value greater than its eight surrounding bins, it is marked as a peak and added to the end of a list of peaks. Its r and t indices, along with its "score," the value of its accumulator bin, are recorded.

5 Results

The Hough transform based line finder presented here is a step forward from the region and heuristic based approaches that are the current state of the art in the SPL. From edge detection through line pairing, this field line detection system is straight forward and effective. The entire system has only three parameters: the edge detection noise threshold, the Hough angle spread parameter, and the Hough noise threshold. Compared with the ad-hoc color based systems, which contain dozens of different parameters, the new system is more robust and does not require significant calibration.



(a) Hough line detection with a good color table (b) Hough line detection with a poor color table

Fig. 2. New line detection does not depend on color calibration

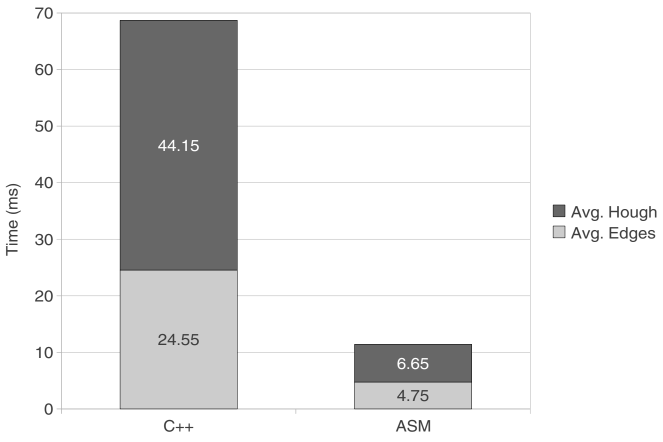


Fig. 3. Average Full Image Line Detection Run Time

The new field line detection system using assembly language coded routines is fast. Compared with a C++ implementation of the same algorithm, it will run significantly faster while producing similar results. Differences between the two implementations come from floating point rounding and a small difference in gradient angle calculations between the implementations. When compared across a test set of 15 images with varying composition, the assembly version averages a factor of six improvement over the C++ version. Figure 3 compares the average run times over entire images (without the field horizon limiting mentioned in

Section 3.1), breaking each version down into the edge detection and Hough transform steps. The C++ version averages a run time of 68.7 milliseconds per image, whereas the assembly version averages 11.4 milliseconds.

This project's goal is to build a Hough transform suitable for use in a robotic system running at 30 frames per second. At 30 frames per second, all the processing for each frame must occur in under 33 milliseconds, including not only vision, but also motion and behavior processing. A line finding system running at more than 15 milliseconds was decided to be unacceptable, as it left too little time for remaining processing. As such, the 11.4 millisecond average run time is acceptable for use in a real-time RoboCup SPL vision system.

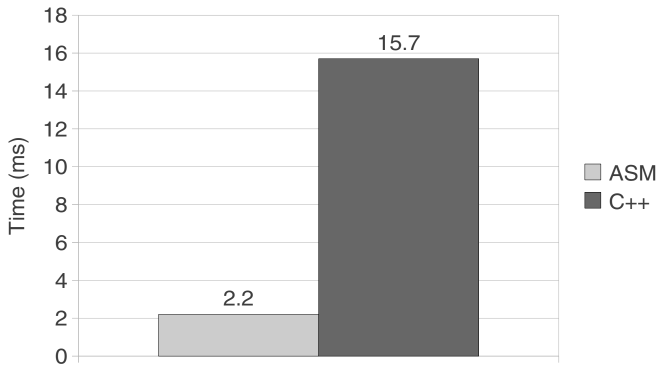


Fig. 4. Average Sobel Operator Run Times

5.1 Edge Detection Performance

The edge detection step benefits the most from the transition to assembly language using the MMX instructions. A speedup factor of seven is achieved in the switch. As the Sobel operator is applied to every pixel in the image, the C++ and assembly raw timing values can be compared.

The combination of SIMD instructions and memory prefetching to improve cache hits yields a significant improvement. Memory access time is effectively cut to zero, leaving only computational time as a constraint. The SIMD instructions help to reduce even that by operating on multiple values simultaneously.

5.2 Hough Transform Performance

The Hough voting routine, as measured in these experiments, is shown to be an acceptable real-time vision algorithm for RoboCup. The C++ version is indeed too slow for real-time use, as the busier images cause it to take over 100 milliseconds to complete. This is over three times the desired frame rate, so it is clearly

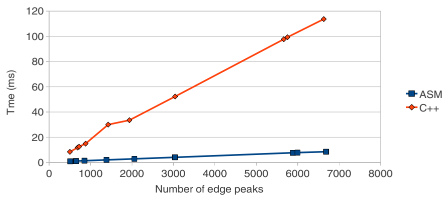


Fig. 5. Average Hough Voting Run Time

not acceptable. The assembly version, however, takes under nine milliseconds, or one tenth of the time to complete on a similar number of edge points. This is a considerable speed up, given that there is no loss of accuracy between the two versions.

The Hough space smoothing routine is a simple operation involving four sums and a subtraction. The assembly version developed here runs over twice as fast as the C++ version. The output of GCC for this code, even using `-mmmx`, which allows the compiler to use MMX instructions, does not use MMX instructions and is thus limited to operating on one bin at a time. Efficient compiler generation of SIMD code is difficult [7] and programmers cannot rely on a compiler to generate any code in particular.

6 Conclusion and Future Work

In the future, the generality of edge detection and shape detection could be extended to the RoboCup goal posts and the ball. These objects are currently detected with color blobbing and present similar opportunities for improvements. A circular Hough transform would be possible with the ball, removing the dependency on color calibration to see the most important object in RoboCup vision. The orange ball presents a strong contrast to the green carpet in the V channel and presents a good opportunity to move slowly away from color calibrated information. The SPL plans to switch away from the orange ball to other styles of ball where the predetermined color is not important. A more general approach to ball detection would be helpful here.

The argument can be made that with faster processors there will be no need to implement solutions like this, but that argument falls apart when other changes are considered. For instance, doubling the dimensions of an image quadruples the pixel count. Higher resolution cameras can quickly overwhelm faster processors. A faster frame rate, higher accuracy demands, or even just more complex objects demand more processing efficiency and power than a faster processor can provide alone. Speed and necessity will always be an issue, so in order to move away from engineered solutions, efficient implementations of general purpose algorithms are necessary.

The transition to general image processing algorithms will improve vision processing robustness in changing environments, and is a necessary transition for the advancement of the sport of robot soccer. By replacing the color segmented and run-length encoding based system with a system using the Hough transform to identify field lines, we have taken a first step towards gaining independence from calibration and manual tuning. This will lead to increased productivity for the roboticist, as they do not have to spend so long calibrating the robot to new situations, and increased performance on the robot, as it can deal with fluctuations within its environment.

We have shown that general vision algorithms can still perform in this real-time environment and are a suitable replacement for the engineered solutions the SPL has typically used. Commonly held beliefs that the Hough transform is too slow for RoboCup are here shown to be baseless. A well designed system can be both robust and fast.

Also, computer scientists often shun hand written assembly as adding needless complexity and non-portability when optimizing compilers are available. As the results here show, an experienced programmer designing a complex system can still beat a compiler by making full use of the processor's architecture and instruction set in ways a compiler may not be able. The compiler's non-use of SIMD instructions is a great example. It is hard in a high level language such as C or C++ to instruct a compiler that it can make use of vector instructions for a certain complex task. By writing it oneself, a programmer can achieve optimal performance from the application.

References

1. Advanced Micro Devices, Inc., One AMD Place, Sunnyvale, CA 94088. AMD Geode^{TM}LX Processors Data Book
2. Ballard, D.H.: Generalizing the hough transform to detect arbitrary shapes*. Pattern Recognition 13(2), 111–122 (1981)
3. Hermans, T., Strom, J., Slavov, G., Morrison, J., Lawrence, A., Krob, E., Chown, E.: Northern Bites 2009 Team Report (2009)
4. Hough, P.: Method and Means for Recognizing Complex Patterns (1962)
5. Jain, R., Kasturi, R., Schunck, B.G.: Machine Vision. McGraw-Hill Science/Engineering/Math. (1995)
6. RoboCup Standard Platform League, `http://www.tzi.de/spl`
7. Leupers, R.: Code selection for media processors with SIMD instructions. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2000, pp. 4–8. ACM, New York (2000); ACM ID: 343679
8. Ratter, A., Claridge, D., Hengst, B., Hall, B., White, B., Vance, B., Nguyen, H., Ashar, J., Robinson, S., Zhu, Y.: rUNSWift Team Report 2010 (2010)
9. Röfer, T., Müller, J., Laue, T.: B-Human Team Report and Code Release 2010 (October 2010)

Extending Virtual Robots towards RoboCup Soccer Simulation and @Home

Sander van Noort and Arnoud Visser

Universiteit van Amsterdam, Science Park 904, Amsterdam, The Netherlands

Abstract. The RoboCup is an initiative to promote the development of robotics in a social relevant way. The competition consists of several leagues and it would be beneficial if developments in one league could be reused in other leagues. This paper describes the development of a simulation model for a humanoid robot inside USARSim, which could be the basis of synergy between the Rescue Simulation, Soccer Simulation and @Home League. USARSim is an existing 3D simulator based on the Unreal Engine, which provides facilities for good quality rendering, physics simulation, networking, a highly versatile scripting language and a powerful visual editor. This simulator is now extended with the dynamics of a walking robot and validated for the humanoid robot Nao. On this basis many other robotic applications as benchmarked in the RoboCup initiative become possible.

Keywords: simulation, multiple kinematic chains, dynamics.

1 Introduction

Robotic simulation is essential in developing control and perception algorithms for robotics applications. Simulation creates the environment with known circumstances, which allows rapid prototyping of applications, behaviors, scenarios, and many other high-level tasks. Robot simulators have been always used in developing complex applications, and the choice of a simulator depends on the specific tasks we are interested in simulating. Yet, the level of realism of a simulator is also important in this choice.

A 3D simulator for mobile robots must also correctly simulate the dynamics of the robots and of the objects in the environment, thus allowing for a correct evaluation of robot behaviors in the environment. Moreover, real-time simulation is important in order to correctly model interactions among the robots and between the robots and the environment. Since simulation accuracy is computationally demanding, it is often necessarily an approximation to obtain real-time performance [1].

In this paper the focus is on the humanoid Nao robot, which is selected by the RoboCup organization as the standard platform for the Soccer competition. In addition, this robot is also used in the @Home competition [2,3] (see Fig. 1).

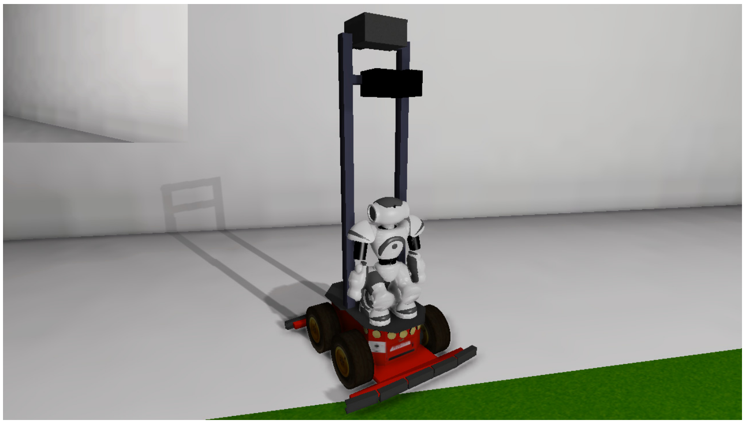


Fig. 1. Configuration of a humanoid robot on a wheeled platform in USARSim, as used in the RoboCup @Home [2]

This robot is widely used in many research institutes around the globe. The Nao contains several kinematic chains (legs, arms, head), which means that its model can be the basis of other robots with multiple kinematic chains.

A model is described to replicate the dynamics of the Nao robot in USARSim [4]; an existing 3D simulator based on the Unreal Engine. Inside USARSim robots are simulated on the sensor and actuator level, making a transparent migration of code between real robots and their simulated counterparts possible. USARSim is an open source project, available on sourceforge[1]. It includes a powerful editor to create worlds and allows experiments, benchmarks and competition scenarios to be set up easily.

2 Related Work

There are many robotic simulator platforms available. The first legged robot developed inside USARSim was the Aibo [5]. The first humanoid robot developed inside USARSim was the Robovie-M [6], developed by the Artisti Humanoid team. Both models were developed on basis of the Unreal Engine 2 / Karma Physics engine. With this engine four Aibo's or two Robovie-M could be simulated before the framerate dropped below an acceptable level. Currently USARSim is based on Unreal Engine 3 / NVidia PhysX. The latter physics engine is more focused on parallelization to make optimal usage of modern cpu's.

Inside the RoboCup @Home League simulation are sparsely used [2,3,7,8,9,10]. Teams typically use older simulation environments, such as Gazebo [7,8] or Carmen [9]. Another possibility is to use a commercial package like Webbots [10]. Essential for this League is to be able to use innovative robot and sensor

[1] `http://usarsim.sourceforge.net`

combinations, a rich environment with a wide variety of shapes and textures, natural lighting, support of the Kinect and preferably a ROS interface. USARSim fulfills all those prerequisites [11].

SimSpark[2] is the official 3D RoboCup simulator and is primarily made for this goal. The simulator is open source and freely available. It uses a client-server architecture, where agents (i.e. robot controllers) are the clients that communicate with the simulation server. A limited number of robots (mainly the Nao) are supported, although it is made easy to add new robots with `rsg` files that describe the physical representation of a robot.

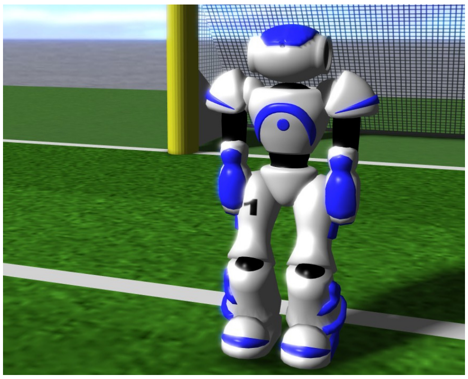


Fig. 2. Screenshot of SimSpark, the simulator used in the Soccer Simulation League

SimSpark always starts a football simulation, including a soccer field, game states and referee. The robots are controlled using a custom protocol, not the native interface of the Nao.

3 Simulation Model

The RoboCup version of the Nao (H21 model) has 21 joints, resulting in 21 degrees of freedom (DOF). There is also an academic version with 25 degrees of freedom, which has 2 additional DOF in each hand.

The movement of each joint can be described by a rigid body equation[12]. The first step is to definition of unconstrained motion as described in equation (1). This equation contains four vectors, it takes both the spatial information $x(t)$, $R(t)$ and the linear and angular momentum $P(t), L(t)$ into account. $F(t)$ and $\tau(t)$ are external forces and the input to solve this equation. The linear and angular speed $v(t)$, $\omega(t)$ can be derived from the linear and angular momentum when the total mass M and the inertia tensor $I(t)$ of a rigid body is known.

[2] `http://simspark.sourceforge.net`

$$\frac{d}{dt}Y(t) = \frac{d}{dt}\begin{bmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \omega(t)^{*}R(t) \\ F(t) \\ \tau(t) \end{bmatrix} \tag{1}$$

The inertia tensor $I(t)$ is time dependent, but can be calculated from the inertia tensor I_{body} in body space, which is a fixed property, by taking the orientation of the body into account $I(t) = R(t)I_{body}R(t)^T$.

$$\left[v(t) = \tfrac{P(t)}{M}, \omega(t) = I(t)L(t) \right]^T \tag{2}$$

The next step is to take contacts into account. When the rigid body encounters a contact it imposes a constraint on the movement.

Two different types of contacts can be distinguished. The first is a contact caused by bumping into another rigid body or into the world. The other type of contact is caused by having a joint defined between two rigid bodies which are part of the robot.

3.1 Joint Definition and Convention

As said in the previous section, a joint connects two rigid bodies and limits the movement in some way. The type of movement limitation results in different types of joints, like a rotational joint, translational joint (also called prismatic joint), spherical joint, screw joint, etc.

A rotational joint, also called *revolute joint*, is as the name suggests capable of rotating around an axis. This type of joint allows one degree of freedom (DOF) between the two rigid bodies, namely the range of motion around the specified axis. In case of this type of joint the motion is usually also limited to a specified range around the axis.

It is important how the relative position and orientation of the frames is characterized. A commonly used convention to describe this is the *Denavit Hartenberg* (DH) notation. This convention uses homogeneous transformation matrices to describe the relative positions of the frames (coordinate systems). This convention is used in USARSim. A full description can found in the book Robotics, chapter 2.2.10, by K.S Fu *et al.*[13].

The Denavit Hartenberg representation is visualized in figure 3. Red lines show the z axes (motion axis) of the joints, while the yellow and green lines show respectively the y and x axes of the joints. The middle blue line shows the start z axis. The other blue lines represent the end points of the five joint chains. Each transformation is represented by a translation / rotation matrix.

3.2 Shape Definition

The shape of the robot is needed to detect collisions between parts of the robot. To define the shape of each part use can be made of the representations of the

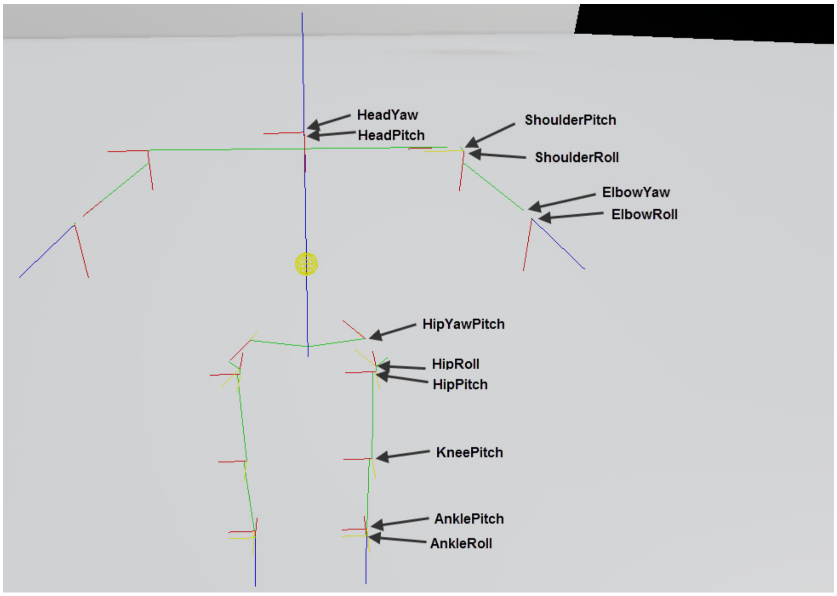


Fig. 3. Visualization of joints according to the Denavit Hartenberg convention. Red lines show the z axes, yellow the y axes and green the x axes of the joints.

Unreal Engine. Two collision representations are relevant for simulations of robot in USARSim. The first collision representation is intended for *static meshes* in Unreal Engine. Static meshes are a type of meshes that are not dynamic. This name does not imply they cannot move or interact with the world. The advanced option for static meshes is to check collisions per polygon against the static mesh 3D model itself and is potentially expensive to use. There is also a (simplified) collision hull option, but this option is not used for robots inside USARSim. Additionally there is a collision representation which is intended for *skeletal meshes* in the Unreal Engine. Skeletal meshes are used for game characters, not for USARSim robots.

The second collision representation is intended for PhysX and is created in the same way as the advanced static mesh version. The PhysX collision model is used in the physics simulation. However sensors will usually involve collision detection with the first representation. For example a simulated sonar sensor uses Unreal Engine tracing to detect objects in the world, which uses the Unreal Engine collision model. Both representations are needed for a realistic simulation of a robot. Care has been taken (as can be seen in Fig. 4) to keep both representations equivalent for the Nao robot. Both the link and shape representation are described in more detail in [1].

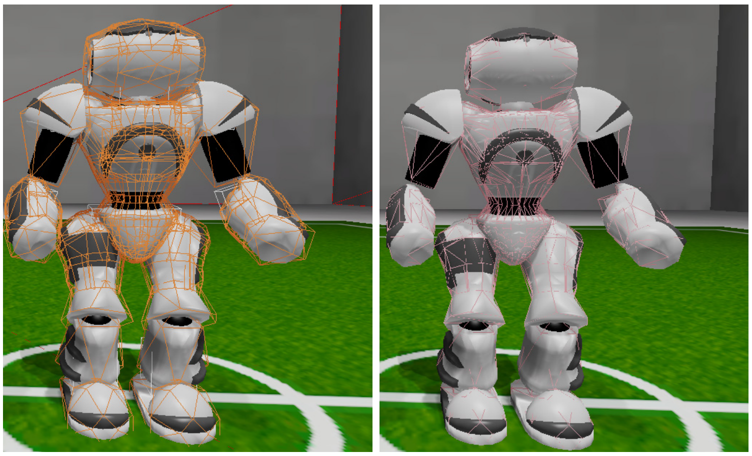


Fig. 4. The left picture shows the PhysX collision model, the right picture the Unreal Engine collision model

4 Experiments

The experiments are divided into two categories; experiments which check general properties for constrained rigid body motion and experiments that are directly related to the proposed Nao model. The basic experiments for constrained rigid body motion are described in an earlier paper [1]. Here we concentrate on the possible applications.

4.1 Advanced Experiments

In this section experiments are done with the simulated and real Nao. The results of these experiments are compared to see how close they resemble each other. The experiments all consist of the combined movement of multiple joints. A more simple version of this experiment would be the movement of a single joint (for instance turning the head). Such simple experiments are performed and show close correspondence. The more advanced experiments are more interesting in the sense that they show sometimes unexpected results due to the interaction of the constraints in between joints. Alternative advanced experiments would the Tai Chi balance (demonstrated in [1]) and collisions between two robots (demonstrated in [5]).

Walking. Realistic walking comparable to the walking behavior of the real Nao is crucial in a humanoid simulation. During a RoboCup match a robot will have to walk a large part of the time.

For this experiment several walking and turning tests were done for the simulated and real Nao using the included walk engine of the Nao provided by

Aldebaran. This walk engine uses a simple dynamic model inspired by work of Kajita *et al.*[14] and is solved using Quadratic programming [15]. When walking at full speed it can reach a velocity of $9.52cm/s$ and $42deg/s$ when turning.

In this test the Nao was set to do a single full step with the left leg. The joint angles of the real and simulated Nao were recorded and compared.

Fig. 5 shows the average joint angles of the LKneePitch joint (i.e. the left knee) with standard deviation over ten recordings of the real and simulated Nao. The standard deviation for the real Nao is lower than the simulated Nao. The same behavior is also seen for the standard deviations of the other joints.

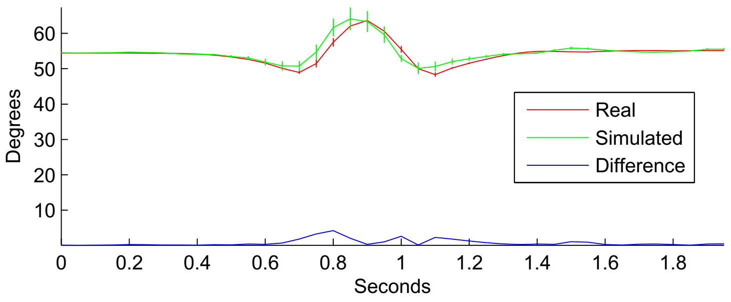


Fig. 5. Average joint angles with standard deviation of the LKneePitch joint while executing a single step. Joint angles were averaged over ten runs for the real (red) and simulated (green) Nao. The blue line shows the difference between the joint angles trajectories.

More walking experiments (including walking multiple steps straight and in circles) are described in [1].

Kicking. Another motion is a kick of a ball with the right leg. The motion was performed ten times for both the real as simulated Nao robot. The recorded joint angles were averaged and the standard deviation was computed.

Figure 6 shows the average joint angles of the RAnkleRoll joint with the standard deviation and the difference between the average joint angles. This joint is interesting because the joint angles trajectory is not the same. During the kick motion the RAnkleRoll joint is told to move to 10 degrees in half a second and stay at 10 degrees for the remaining part of the motion.

The angles trajectory shows not much difference in moving towards 10 degrees, although the standard deviation of the real Nao angles is higher than the simulated Nao angles. However when staying at 10 degrees the real Nao joint is not able to maintain this angle around 1.5 second. In this case the Nao fails to reproduce the behavior of the real joints because we did not include the restrictions of the collision hull of this particular joint in our model[3]. The joint angle range of this joint is limited by the movements of the AnklePitch joint. Around

[3] `http://www.aldebaran-robotics.com/documentation/nao/hardware/kinematics/nao-joints-33.html`

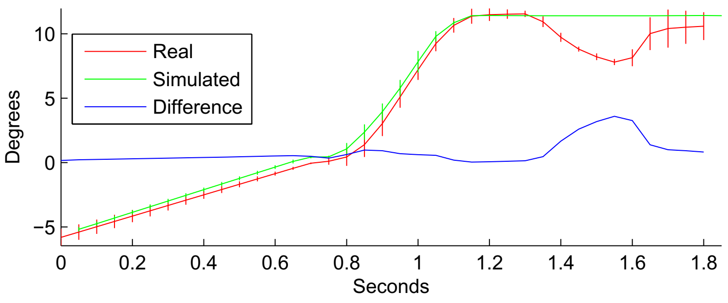


Fig. 6. Joint angles and standard deviation of the RAnkleRoll joint while executing a kick motion. Results were averaged over ten runs. The red line shows the angles trajectory of the real Nao, while the green line shows the same for the simulated Nao. The blue line shows the difference between the two angle trajectories.

1.5 second the RAnklePitch joint moves from around -30 degrees to -60 degrees. At this joint angle the RAnkleRoll becomes limited to a range of between -6 and 3 degrees.

5 Full Application Experiment

To test how well the performance is for real applications, the source code of the Dutch Nao Team[16] has been tested with USARSim.

This application not only involves walking around, but also perception of the ball and dedicated behaviors like kicks and standing up.

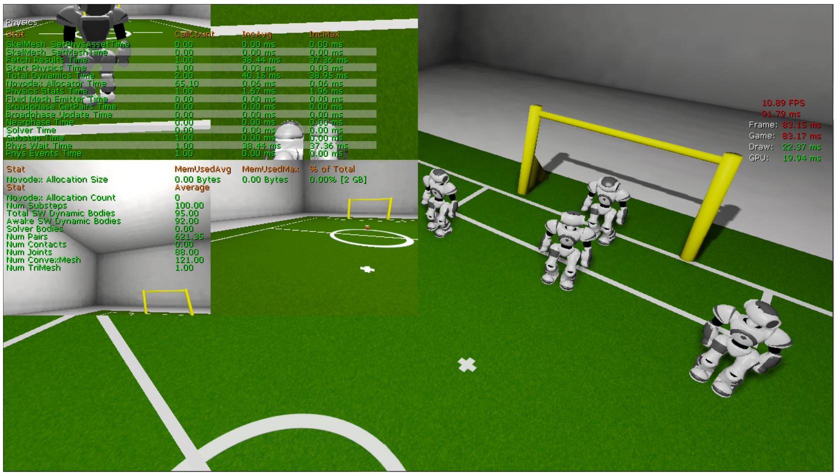
To test real applications an intermediate program has been created, UsarNaoQi, which works as a proxy server, converting NaoQi messages in USARSim messages and vice versa. NaoQi is the framework provided by Aldebaran and allows the user to control the Nao in various programming languages (C++, Python, C# or Urbi).

The source code of the Dutch Nao Team is written in Python, and could be directly applied. The code was fully functional, the robots could standup, position themselves on the field, locate the ball and kick the ball. The only observed difference is in the approach of the ball; the Dutch Nao Team code makes a number of small steps to get in a good position behind the ball. In simulation those steps are too small; the Nao needs too much time to position itself.

The experiment was performed by putting a number of Nao robots in the simulated RoboCup environment. The average frames per second (FPS) was recorded for two different scenarios. In the first scenario the Nao is simply standing and doing nothing. In the second scenario we executed the Nao with robot controller from the Dutch Nao Team. The controller was set in *play* mode. In this mode the Nao will walk around scanning for the ball.

The experiment was performed on a computer with an Intel iCore 7 920 processor and an AMD Radeon HD 6850 graphics card. USARSim was used in combination with the UDK December build 2011. UsarNaoQi was set to use a time step of 10ms; the Naos in USARSim sent status updates at a rate of 100 times per seconds (joint angle updates).

Without any Naos the scene is rendered at a FPS of 320. With one and two Naos the FPS drops to around 110 and 55 respectively, which is enough for running a decent simulation. With three Naos the FPS drops to 30, which is still acceptable. With four Naos the simulation frame rate drops to 10 FPS, resulting in incorrect movements.


```
Physics
Stat                           CallCount  IncAvg     IncMax
SkelMesh_SetPhysAssetTime      0.00       0.00 ms    0.00 ms
SkelMesh_SetMeshTime           0.00       0.00 ms    0.00 ms
Fetch Results Time             1.00       38.44 ms   37.36 ms
Start Physics Time             1.00       0.03 ms    0.03 ms
Total Dynamics Time            2.00       40.15 ms   38.95 ms
Novodex Allocator Time         65.10      0.06 ms    0.06 ms
Physics Stats Time             1.00       1.67 ms    1.96 ms
Fluid Mesh Emitter Time        0.00       0.00 ms    0.00 ms
Broadphase GetPairs Time       0.00       0.00 ms    0.00 ms
Broadphase Update Time         0.00       0.00 ms    0.00 ms
Nearphase Time                 0.00       0.00 ms    0.00 ms
Solver Time                    0.00       0.00 ms    0.00 ms
Substep Time                   1.00       0.00 ms    0.00 ms
Phys Wait Time                 1.00       38.44 ms   37.36 ms
Phys Events Time               1.00       0.00 ms    0.00 ms
Stat                           MemUsedAvg  MemUsedMax  % of Total
Novodex Allocation Size        0.00 Bytes  0.00 Bytes  0.00% [2 GB]
Stat                           Average
Novodex Allocation Count       0
Num Substeps                   100.00
Total SW Dynamic Bodies        95.00
Awake SW Dynamic Bodies        92.00
Solver Bodies                  0.00
Num Pairs                      621.35
Num Contacts                   0.00
Num Joints                     88.00
Num ConvexMesh                 121.00
Num TriMesh                    1.00
```

```
10.89 FPS
91.79 ms
Frame: 83.15 ms
Game:  83.17 ms
Draw:  22.37 ms
GPU:   19.94 ms
```

Fig. 7. Four Naos in action with the physics statistics displayed

To find the performance bottlenecks in the simulation various profiler tools provided by UDK are used (PhysX statistics and UnrealScript code profiler). Using these tools reveals that when simulating four Naos half of the frame time is spent in the physics. The remaining part of the time goes to the sonar sensor (tracing), receiving and processing messages in the bot connection with the controller, sending the current status to the controller (joint angles) and updating the current joint angles.

6 Discussion

Sensors. The experiments are limited to the motion of the simulated Nao caused by the movement of the joints. However the Nao is equipped with a wide range of sensors (as discussed in the introduction). The different sensors like the cameras, bumpers, sonars and inertial unit also contribute to the behavior of the Nao. More research is needed specifically aimed at these sensors. For example the Nao is equipped with two cameras. Although the camera sensors obviously function, it is not possible to say much about the correct working of these cameras without validation. Figure 8 shows an example of the problems you encounter when simulating a camera. The sensitivity of the camera of the different Nao versions results in a different camera image. Such differences would need to be modeled to simulate a camera properly. Although Unreal Engine already offers excellent rendering options, the current implementation in USARSim limits the simulation of the camera to simply capturing the image and sending it without modification.

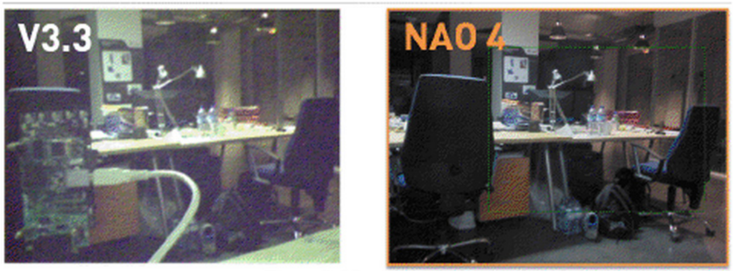


Fig. 8. Camera image of Nao 3.3 vs 4.0 (Courtesy Aldebaran Robotics)

Servo motor. Another interesting research option is to extend the simulator with a more realistic servo motor and gears simulation, as used in the MoToFlex simulator[17]. In a physics engine a common way to control the joints of a robot is to set the desired joint angles and leave it to the physics engine to satisfy the constraints between the links (as described in this paper). This approach is not the most realistic way to drive a joint and the method seen in the MoToFlex simulator[17] could improve the behavior of the joints.

Scaling issues. Due several design choices it is not possible to scale up to high number of simulated Nao robots. Scaling up the number of Naos is important for simulating a RoboCup scenario. The goal of the RoboCup competition is to play a real football match with 22 Naos. Part of the reason why the simulation cannot scale up to this number of Naos is due the choice of the collision model, the physics timing step and possible overhead of message parsing and other sonar sensor. Scaling of the number of Naos could be improved by simplifying the collision model by using more simple shapes (spheres, boxes) and lowering the Physics timestep settings. The same could be applied to the message parsing (moving the code from Unreal Script to C for example) and the sonar sensor (using less traces to determine the sonar distance).

Extending to other legged robots. One of the goals was to make a generic model that could be applied to other limb typed robots, like the ABB Frida or a spider like robot. The collision tools of Unreal Engine allows to quickly create varies collision shapes, varying from simple models (boxes, spheres) to complex convex models, possible based on the visual shape of the robot.

This model can easily be applied to other robots with multiple kinematic chains as shown in figure 9.

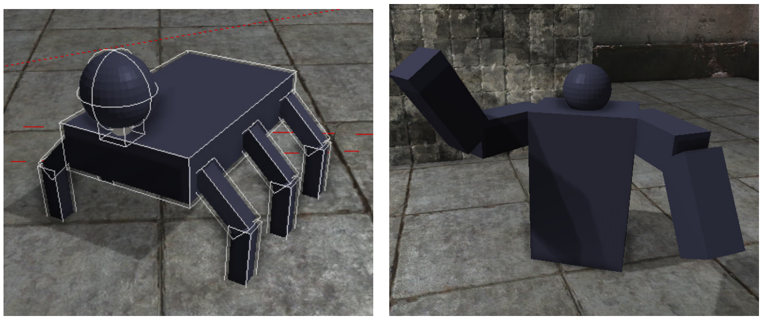

(a) A spiderlike robot (b) A two arm manupulator robot

Fig. 9. Example of robots with multiple kinematic chains

7 Conclusion

In this paper we demonstrated that the simulation of the Nao in USARSim resembles reality quite closely. Our current model is usable in practice on the condition that one keeps in mind the limits of the method; like the walking behavior and the scaling issues with the number of Naos. The combination of Unreal/USARSim provides several advantages over other robot simulators. The simulation is at such a level that transparent migration of code between real robots and their simulated counterparts is possible. In this paper this is demonstrated with an intermediate program, UsarNaoQi, which enables access to the simulated robot with its native interface. Using this interface several experiments have been performed with both the real and simulated robot. The model developed for this humanoid robot demonstrates that robots with complex dynamics could be realistically modeled inside USARSim, which could be the basis of the introduction of other models of complex robots into USARSim like two-arm manipulators and/or service robots.

References

1. van Noort, S., Visser, A.: Validation of the dynamics of an humanoid robot in usarsim. In: Proceedings of the Performance Metrics for Intelligent Systems Workshop (PerMIS 2012) (March 2012)
2. van Elteren, T., Neculoiu, P., Oost, C., Shantia, A., Snijders, R., van der Wal, E., van der Zant, T.: Borg - the robocup@home team of the university of groningen - team description paper. In: Proc. CD of the 15th RoboCup International Symposium (July 2011)

3. Dessimoz, J.D., Gauthey, P.F.: Rh6-y toward a cooperating robot for home applications. In: Proc. CD of the 15th RoboCup International Symposium (July 2011)
4. Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: Usarsim: a robot simulator for research and education. In: Proceedings of the International Conference on Robotics and Automation (ICRA 2007), pp. 1400–1405 (2007)
5. Zaratti, M., Fratarcangeli, M., Iocchi, L.: A 3D simulator of multiple legged robots based on uSARSim. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 13–24. Springer, Heidelberg (2007)
6. Greggio, N., Menegatti, E., Silvestri, G., Pagello, E.: Simulation of Small Humanoid Robots for Soccer Domain. Journal of the Franklin Institute 346(5), 500–519 (2009)
7. Lunenburg, J., Clephas, T., Dirkx, N., Willems, B., Elfring, J., Sandee, J., van de Molengraft, M.: Tech united eindhoven team description 2011. In: Proc. CD of the 15th RoboCup International Symposium (July 2011)
8. Alenyà, G., Tellez, R.: The reem@iri 2012 robocup@home team description. In: Proc. CD of the 16th RoboCup International Symposium (June 2012)
9. del Solar, J.R., Correa, M., Lee-Ferng, J., Hevia-Koch, P., Parra, I., Mascar, M.: Uchile homebreakers 2010 team description paper. In: Proc. CD of the 14th RoboCup International Symposium (June 2010)
10. Lallee, S., Lise Jouen, A., Petit, M., Madden, C., Boucher, J.D., Weitzenfeld, A., Dominey, P.F.: Cooperative human robot interaction with the nao humanoid: Technical description paper for the radical dudes. In: Proc. CD of the 15th RoboCup International Symposium (July 2011)
11. Balakirsky, S., Kootbally, Z.: USARSim/ROS: a combined framework for robot control and simulation. In: Proceedings of the ASME 2012 International Symposium On Flexible Automation (ISFA 2012) (June 2012)
12. Baraff, D.: An introduction to physically based modeling: rigid body simulation I - unconstrained rigid body dynamics. SIGGRAPH Course Notes (1997)
13. Fu, K., Gonzalez, R., Lee, C.: Robotics: control, sensing, vision, and intelligence. McGraw-Hill (1987)
14. Kajita, S., Tani, K.: Experimental study of biped dynamic walking. IEEE Control Systems Magazine 16(1), 13–19 (1996)
15. Wieber, P.: Trajectory free linear model predictive control for stable walking in the presence of strong perturbations. In: Proceedings of the International Conference on Humanoid Robots, pp. 137–142 (2006)
16. Verschoor, C., et al.: Dutch nao team - code release 2011 and technical report 2011, Universiteit van Amsterdam (October 2011) (published online)
17. Urbann, O., Kerner, S., Tasse, S.: Rigid and soft body simulation featuring realistic walk behaviour. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 126–136. Springer, Heidelberg (2012)

A Survey about Faults of Robots Used in RoboCup

Gerald Steinbauer

Institute for Software Technology
Graz University of Technology, Graz, Austria
steinbauer@ist.tugraz.at

Abstract. Faults that occur in an autonomous robot system negatively affect its dependability. The aim of truly dependable and autonomous systems requires that one has to deal with these faults in some way. In order to be able to do this efficiently one has to have information on the nature of these faults. Very few studies on this topic have been conducted so far. In this paper we present results of a survey on faults of autonomous robots we conducted in the context of RoboCup. The major contribution of this paper is twofold. First we present an adapted fault taxonomy suitable for autonomous robots. Second we give information on the nature, the relevance and impact of faults in robot systems that are beneficial for researcher dealing with fault mitigation and management in autonomous systems.

1 Introduction

A major goal of research in Robotics and Artificial Intelligence is to develop truly autonomous robots that are able to assist humans in exhausting or repetitive tasks or in dangerous environments. Examples include factories, mining, search and rescue missions and space exploration. In such application domains dependability plays a major role because autonomous robots must not endanger people sharing common space, face unexpected situations or allow no human intervention in the case of problem. In the context of autonomous robots dependability is strongly coupled with fault mitigation.

Autonomous robots in academia and industry frequently show faults or undesired behaviors. This is caused by physical faults in the robot's hardware, bugs in the robot's software, unreliable algorithms or unexpected situations not considered during development. A lot of research has been conducted to develop automated methods for fault-tolerance, diagnosis and repair to be able to cope with these problems. Such methods improved the reliability of robots but still we see a large fraction of robots used in academia or industry fail due to the reasons mentioned above.

Although, the topics of dependability are very important surprisingly there are few founded publications and studies about the reliability of autonomous robots and the relation between faults, counter-measures and dependability. Such work either qualitative or quantitative seems yet to be extremely interesting for all autonomous robots researchers and developers. So far we have only few information about what the common types of faults and problems are. Therefore, no answer can be given to the question what issue tackled had the most positive impact on robot system faults. It might be nice if one has a method that allows a robot to automatically cope with a particular type of fault. If this type of fault, though, occurs once a year in one particular robot the methods has very limited impact on the general reliability of autonomous robots.

Based on the above observations we investigated the question: *What are the common, probable and most critical faults and malfunctions of autonomous robots?* In order to give a first founded answer RoboCup is an ideal environment. RoboCup is one of the leading initiatives to foster research and development in Artificial Intelligence and Robotics [1]. In RoboCup a large number of different research robots are used that demonstrate a wide variety of system architectures and properties. Moreover, the tasks and environments used in the RoboCup competitions are dynamic and complex and ask for non-trivial solutions. In order to be able to identify the common and therefore interesting faults to deal with we initiated a survey on robot faults in RoboCup. In the survey groups participating in different RoboCup leagues had been asked to describe their robot systems, the type and frequency of faults their systems suffer from and the impact of these faults on the mission success.

This primarily qualitative survey provides information about the properties of faults of robots used in RoboCup. It can be used to derive common activities and processes to improve the robot's reliability and to guide research into promising directions. For sure this survey is also interesting for researchers and developers outside the RoboCup community as the robots and techniques used in RoboCup are state of the art and similar robots are used in other fields of research and industry as well.

2 Definitions and Related Work

In this section we give basic definitions that are used throughout the remainder of the paper.

A *failure* is an event that occurs when the delivered service deviates from correct service. An *error* is that part of the system state that can cause a subsequent failure. A *fault* is the adjudged or hypothesized cause of an error.

For the current survey on faults of robots used in RoboCup we adapt the fault taxonomy of [2]. Our taxonomy shown in Figure 1 is more suitable for this particular domain. The category *hardware* comprises all faults related to physical faults of the

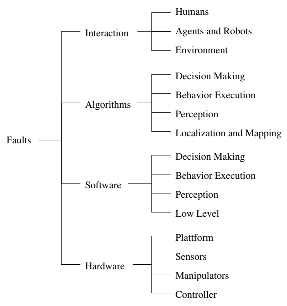


Fig. 1. Fault taxonomy used in current survey

robot's equipment. Design and implementation faults of the robot's software systems fall in the group *software*. The category *algorithms* aggregates problems of methods and algorithms based on its nature. Here we distinguish between a faulty implementation of an algorithm and shortcomings of the algorithm itself. In the category *interaction* we group problems that arise from uncertainties in the interaction with the environment (perception and acting), other agents and humans.

Furthermore, we use several attributes to classify faults and their properties. Some of them are reused from the literature others are newly defined or clarified. The used attributes are: (1) *relevance* of a fault for different robot systems, (2) relation to *condition* and events in the environment, (3) relation to *symptoms* (failures), (4) *impact* to the mission's success and (5) *frequency* of the occurrence of a fault.

We choose the fault categories and attributes based on our experience in research for fault diagnosis and repair [3,4]. Basically we found out that classical faults in hardware and software are not sufficient explanations for undesired behavior of an autonomous robot, e.g., the non-determinism of actions in the real world has a big influence on dependability. Moreover, much researcher work on advanced solutions to handle particular faults but ignore the question whether the fault is very much relevant. Therefore, information on the properties of faults such as impact is important.

As mentioned in the introduction there is few literature and studies on faults of autonomous robot systems. One of the few examples is [2] where the authors did a qualitative and quantitative study on faults of robots and their impact in the search and rescue domain. Moreover, probabilities were given for how likely it is that parts fail. In [5] the authors presented an autonomous tour guide robot and described its continuous operation for five month. The authors reported a Mean Time Between Failures (MTBF) of around 4.6 hours of the robot system and noted that most of the failures were caused by software components. This is one of the few long-term studies.

Intuitively one may assume that robot systems having abilities to automatically recover from faults will show a higher dependability. The authors of [6] showed that a team of robots with cooperative repair capabilities has a better chance to complete a cooperative mission. Research on automated recovery from faults in the context of autonomous robots systems is a very active field. The techniques comprise automated detection, identification and repair of faults. If the former two techniques are combined we refer to it as diagnosis. Diagnosis and repair techniques are quite heterogeneous in their properties (qualitative, quantitative or hybrid) and are able to cope with a wide range of faults (hardware, software or knowledge). In [7] the authors showed how particle filter can be used to detect faults in the mobility unit of a planetary rover. Many approaches like [8] specifically addressed the diagnosis of sensing and/or actuator faults. In [3] the authors presented a framework for detection and repair of faults in the control system of autonomous robots. The framework was based on qualitative model-based diagnosis.

To improve the dependability of autonomous systems a number of approaches have been proposed to handle unexpected situations caused by the environment. For instance there is a lot of work in the area of planning and execution monitoring to handle execution faults and unexpected external events. Example architectures are LAAS [9], CLARAty [10] and the Livingstone architecture by Williams and colleagues [11].

3 Investigated RoboCup Leagues

The robot platforms, the methods and the task used in RoboCup resemble to a high extent what is used and done around the world in Robotics and AI labs. In order to have access to a wide variety of robots we included the following leagues in our study: Middle Size, Rescue Robot and RoboCup@Home.

In the *Middle Size League* (MSL) two teams of up to six autonomous robots each play soccer against each other [12]. This league is the one that approximates real soccer best in terms of speed of the game and size of field (18 m width). The used robots have only some size and weight restrictions and move with a speed of up to 5 m/s. No external sensors or interventions are allowed. Research challenges in this league comprise methods for controlling a robot at high speed, localization and navigation with sparse landmarks or machine learning for behaviors like dribbling and team coordination.

The *Rescue Robot League* (RRL) [13] provides a testbed for robots in the urban search and rescue domain. The goal is to develop robots that are able to explore hazard sites after disasters like earth quakes or chemical accidents without endangering first responders. In the competition teams have to explore a given arena built of standard test modules such as ramps, step fields or stairs. The two major tasks are to localize trapped victims and to build a map of the arena. Research challenges comprise mechanical design of robust agile robot platforms, simultaneous localization and mapping (SLAM), exploration strategies or object recognition for victims and hazard material signs.

In the RoboCup@Home League (@Home) [14] service robots for office and home environments are developed. The test bed is an apartment with common parts such as a kitchen, a bed room or a living room. During the competition the robots have to autonomously perform several predefined tests such as to guide a guest to a particular place, find a given object or recognize persons. These resemble simple useful tasks in daily life. Moreover, teams can show individual achievements like newly integrated sensors or special capabilities of their robot in a free challenge. Research challenges comprise safe navigation in indoor environments, grasping and manipulation of objects, human-robot interaction (HRI) or object and face recognition.

4 Survey and Data Collection

4.1 Information about Fault and the Questionnaire

The aim of our work is to get a qualitative estimation which parts of robot systems are affected by faults, what the root causes are, what the symptoms (failures) caused are and what their impact and frequency is. These parameters very much matters for the design and implementation of fault diagnosis and repair approaches.

In order to collect the data needed to answer the above questions we divide the survey into nine main categories:[1]

1. general information on the groups and their hardware and software
2. the robot platform and related faults

[1] The full questionnaire, the full data set of the survey and further interpretations of the data are publicly available at `http:\\www.ist.tugraz.at/rfs`.

3. the robot's sensors and related faults
4. the robot's manipulators and related faults
5. the robot's control hardware and related faults
6. the robot's software and related faults
7. the used algorithms and related faults
8. off-line and on-line measures to deal with faults
9. research focus and general feedback

The questions in categories 1 and 9 provide general context information about the used robots and the research groups involved. Category 8 collects information about fault mitigation strategies already used in the field. The aim of the questions in the categories 2 to 7 is to collect broad feedback on possible faults in various robot subsystems and their impact. Therefore, these categories are further divided in five standard blocks:

1. how much are the related parts affected by faults
2. what are the dominant root causes for faults
3. how much fault are correlated with root causes
4. what are dominant symptoms (failures) caused
5. what is the impact of faults on the mission success rate
6. what is the frequency of particular faults

For instance the related parts of the robot's platform comprise among others wheels, tracks, motors, motor drivers or batteries. Root causes comprise physical problems such as wear or damage, connection or configuration problems or environmental conditions. Symptoms related to faults comprise immobility, power failures, unpredictable behavior or missing data. The selection has been done based on our experience in the construction of autonomous robots, the participation in RoboCup and the daily lab work as well.

The impact of faults is categorized as *not critical, repairable/compensable or terminal*. The frequency of faults are categorized as *never, sporadic, regularly or frequently*. These categories are related to the schedule of the RoboCup competition. *Repairable* denotes faults that can be repaired during a mission or game while *terminal* denotes faults that lead to an abortion of a mission or robots not able to take part in a game anymore. *frequently* means that a fault occurs in each mission or game at least once.

The questions about the relevance of faults and how much parts are affected by faults had to be answered by integers from 0 (not relevant or affected) to 5 (very much relevant or affected). For the correlation of faults to causes matrices were used where the entries represented a particular correlation and could be checked during answering.

4.2 Data Collection Process

The data collection process was conducted as on-line survey using a renowned commercial survey platform. We invited 68 research groups from 21 countries that participated in the last years in the RoboCup world championship or a major regional competition to complete the survey. 16 of these groups participated in Middle Size, 22 in Rescue Robot and 30 in RoboCup@Home. The survey was available on-line from August to November 2010. We received 25 responses to the invitation. This represents a good number of 36,7 % of the invited groups. Due to incomplete submissions we incorporate only 17 responses (25 % of the invited groups) in our survey. 6 responses came from the Middle Size, 5 from the Rescue Robot and 6 from the RoboCup@Home league. A response rate of 25 % of useful responses is above the usual rates in surveys and can be explained by the compact group of researchers participating in RoboCup and the interest in empirical data.

5 Results and Interpretation

5.1 General Information

We start with general information about the used robot systems. Table 1 left shows an overview of the used robot platforms. Nearly 80 % of the research groups use custom-built robot. This fact, though, is not equally true for all application areas. While groups in MSL and RRL mainly use custom-built robot platforms, @Home groups use commercially available platforms to a high extent. For the locomotion of the robot platform mainly differential drives (35 % of the groups), omni-directional drives (35 %), skid drives (25 %) or tracks and flippers (18 %) are used. Please note that some groups use more than one platform or locomotion type.

Table 1. Robot platforms (left) and sensors (right) used by teams in the different RoboCup leagues. Some teams use multiple platform types. Number of teams.

Platform	MSL		RRL		@Home		Overall	
	#	%	#	%	#	%	#	%
Pioneer/GuiaBot	0	0	1	20,0	2	33,33	3	17,65
Telemax	0	0	0,33	6,7	0	0	1	5,88
Madilda	0	0	0,33	6,7	0	0	1	5,88
Volksbot	0	0	0	0	1	16,67	1	5,88
other commercial	0	0	0	0	2	33,33	2	11,76
custom-built	6	100	4,34	86,6	1	16,67	12	70,59
sum	6	100	5	100	6	100	17	100

Sensor	MSL		RRL		@Home		Overall	
	#	%	#	%	#	%	#	%
Odometry/Encoder	6	100	5	100	5	83,3	16	94,1
Directed Camera	3	50	4	80	4	66,7	11	64,7
Hokuyo LRF	0	0	4	80	5	83,3	9	52,9
Omni-Camera	6	100	1	20	0	0	7	41,2
Sick LRF	0	0	0	0	5	83,3	5	29,4
Compass	2	33,3	3	60	0	0	5	29,4
IMU	1	16,7	3	60	0	0	4	23,5

Table 1 right shows an overview of sensors used on the robot systems. Some of them as odometry and directed cameras are commonly used in all domains. Others are used frequently in particular domains such as the light-weighted Hokuyo Laser Range Finder (LRF) (only RRL and @Home) or omni-directional cameras (mainly MSL). Specific sensors such as gas sensors are used very seldom for special purposes only.

Further used hardware components are manipulators. All groups except one from RRL and @Home use such devices. The most common types are the Katana robotics arm and custom-built solutions.

Notebooks are used as central control unit by over 60 % of the groups. Industrial PCs follow with 23,5 %. The major interfaces to hardware used are USB (by 88,2 % of the groups), RS232 and Ethernet (64,7 % respectively) and IEEE139a (52,9 %).

5.2 Robot Platform

We continue with results about faults of the robot platforms. Table 2 left depicts the feedback to which extent the various parts are affected by faults. Columns 2 to 7 shows

Table 2. Ranking of parts of the robot platform affected by faults (left) and the relevance of causes (right)

Platform Part	Ranking (0 .. not affected. 5 .. much affected)						
	0	1	2	3	4	5	avg.
	%						
Batteries	29,4	11,8	41,2	5,9	11,8	0	1,59
Motor Drivers	33,3	20,0	26,7	6,7	13,3	0	1,47
Controller Boards	29,4	41,2	11,8	5,9	11,8	0	1,29
Tracks	62,5	0	12,5	12,5	0	12,5	1,25
Motors	46,7	33,3	0	0	20,0	0	1,13
Gears	66,7	6,7	13,3	0	13,3	0	0,87
Wheels	68,8	6,3	12,5	6,3	6,3	0	0,75
Flipper	75,0	12,5	0	0	0	12,5	0,75
Chassis	75,0	18,8	0	0	0	6,3	0,5

Causes	Ranking (0 .. not relevant. 5 .. much relevant)						
	0	1	2	3	4	5	avg.
	%						
Connectors	11,8	5,9	23,5	17,6	23,5	17,6	2,88
Communication	17,6	17,6	23,5	17,6	11,8	11,8	2,24
Physical Impact	31,5	12,5	25,0	12,5	6,3	12,5	1,88
Wear	35,3	5,9	29,4	17,6	5,9	5,9	1,71
Vibration	29,4	17,6	23,5	17,6	11,8	0	1,65
Damage	23,5	41,2	11,8	11,8	5,9	5,9	1,53
Configuration	35,3	23,5	23,5	5,9	5,9	5,9	1,41
Temperature (Overheat)	52,9	5,9	11,8	11,8	11,8	5,9	1,41
Short Circuits	37,5	25,0	18,8	12,5	0	6,3	1,31
Environmental Conditions	76,5	5,9	11,8	5,9	0	0	0,47

the fraction of groups that assigned the corresponding scores (row 2) to various faults (column 1). Column 8 depicts the average score. It shows that batteries and motor driver are the primarily affected parts. An average score of 1,59 and 1,47 shows that these parts are moderately affected. Anyhow, some parts like tracks or flippers got individual high scores showing that some groups have serious problems with faults in these parts.

Furthermore, we were interested in the primary causes for faults in the robot platforms. Table 2 right depicts the relevance of different causes. The major causes for faults of robot platforms are problems with connectors (average score of 2,88) and communication problems (e.g., protocol or transmission, score 2,24). Moreover, we asked for the relation of faults and causes. According to the response connector problems are major causes for faults of batteries (40 %), controller boards (43 %) and motor drivers (42 %). The numbers reflects the fraction of groups that see a relation between causes and faults. Another prominent cause is wear with relation to faults of gears (75 %) and batteries (50 %).

Another interesting aspect is to which symptoms (failures) the different faults lead. Usually the correlation between symptoms and root cause is not obvious. With the same relevance score of 0 to 5 the major symptoms are immobility (average score of 3,12), unpredictable behavior (2,67) and reduced controllability (2,56).

Finally, we asked for feedback on the impact and frequency of faults of parts of robot platforms. Faults of batteries, motors and controller boards are the main reason for a termination of missions. Most of the other faults were mainly classified as repairable. Faults of batteries and motor drivers occur sporadic to regularly. The other faults occur never to sporadic.

5.3 Sensors

We used the same set of questions and scores for feedback on sensors used by the groups. According to their feedback the major sensors affected by faults are Hokuyo LRF (average score of 1,78), Swiss rangers (1,25), directed cameras (1,23) and sonars (1,2). IMUs (score 0,6) and odometry (0,65) are the most reliable sensors.

The most relevant causes for sensors faults are connectors (average score of 2,73), configuration problems (2,54) and communication problems (2,5). The most significant relation between sensor faults and causes exist between the Hokuyo LFR and configuration problems (43 %) as well as physical impact (43 %). More than 50 % of the groups mentioned a relation between most of the sensors and connector problems.

The major symptoms (failures) caused by sensor faults are that the sensor delivers no data (average score of 3,76) and wrong or corrupted data (2,47). An unstable data rate is a minor symptom (0,75). According to the responses faults of Sick LRF are terminal to the mission while most of the other faults are repairable. Faults of directed cameras and sonar occur regularly while most other faults are classified as sporadic.

5.4 Manipulators

Manipulators seem to be sensitive parts. They received a high average score of 3,09 for being affected by faults. The leading causes for faults are physical impact (average score of 3,7), damage (3,42), communication problems (2,17) and connectors (2,0). Major symptoms (failures) are problems with the kinematics (average score of 2,88), the precision (2,55) and the payload (2,33). Faults of manipulators have also high impact on the mission success as 45,4 % of the groups classified them as terminal. Nevertheless, over 70 % of the groups classified manipulator problems as sporadic.

5.5 Control System

This section only concerns control system's hardware such as computer and notebooks. The major causes for faults of the control system are connectors (average score of 2,24 out of 5,0), configuration problems (2,0), communication problems (2,0) and vibrations (1,36). The most prominent symptom of control system faults is that peripheral functions (e.g. particular ports) are missing or non-functional (average score of 2,88). This symptom is followed by the fact that the system does not boot (1,71), hangs (1,6) or crashes (1,6). Most of the symptoms such as the system does not boot, reboots or hangs were classified by over 50 % of the groups as terminal. The responses show that most problems with the control hardware are serious but occur only sporadically. But one third of the groups replied that missing functionality occurs regularly.

5.6 Robot Software

This section concerns software engineering aspects of the robot control software. We asked for faults of 13 major software parts such as computer vision, self-localization, low-level drivers, decision making, inter-process communication or behavior execution.

According to the feedback computer vision (average score of 2,06), behavior execution (2,0), inter-robot communication (1,85) and low-level device drivers (OS dependent, e.g., USB stack, 1,76) are the most affected software parts.

Table 3 left shows the relevance of causes for faults in the software. The major causes are configuration problems (average score of 2,56), performance leaks (i.e., miss of deadlines, 1,88), memory leaks (1,31) and access violations (1,19). Mutual exclusions (score of 0,64) and overflows (0,57) play only a minor role as causes.

Of particular interest is the relation of causes to faults within the software. About 30 to 70 % of the groups report a relation between configuration problems and failures in different parts of the software. Performance leaks are named as cause for faults in object tracking (60 %), computer vision (54 %) and inter-robot communication (50 %).

Table 3. Ranking the relevance of causes for robot software faults (left) and algorithms used affected by faults (right)

Causes	Ranking (0 .. not relevant. 5 denotes much relevant.)						
	0	1	2	3	4	5	avg.
	%						
Configuration	12,5	12,5	18,8	37,5	0	18,8	2,56
Performance Leaks	25,0	18,8	31,5	0	18,8	6,31	1,88
Memory Leaks	31,3	25,0	25,0	18,8	0	0	1,31
Access Violation	18,8	50,0	2,0	6,3	0	0	1,19
Race Conditions	60,0	26,7	6,7	0	6,7	0	0,67
Mutual Exclusions	78,8	0	14,3 0	0	0	7,1	0,64
Overflows	64,3	21,4	7,1	7,1	0	0	0,57

Algorithm	Ranking (0 .. not affected. 5 .. much affected)						
	0	1	2	3	4	5	avg.
	%						
Decision Making - State Machine	0	33,3	33,3	0	0	33,3	2,67
Object Recognition 2D	0	28,6	21,4	28,6	14,3	7,1	2,50
Feature Extraction	7,7	23,1	30,8	23,1	7,7	7,7	2,23
Mapping 3D	0	60,0	0	20,0	20,0	0	2,00
Mapping 2D	0	69,2	0	15,4	0	15,4	1,92
Classification	0	55,6	11,1	22,2	11,1	0	1,89
Path Planning	7,7	53,8	23,1	0	7,7	7,7	1,69
Path Execution	0	66,7	8,3	16,7	8,3	0	1,67
Self-Localization - Sample Based	21,4	28,6	21,4	21,4	7,1	0	1,64
Object Recognition 3D	0	25,0	25,0	25,0	0	25,0	2,75
Reasoning/Planning - Logic Based	0	33,3	33,3	0	33,3	0	2,33
Reasoning/Planning - Probability Based	0	50,0	50,0	0	0	0	1,50
On-Line Machine Learning	33,3	0	66,7	0	0	0	1,33
Self-Localization - Classical Filter	33,3	33,3	33,3	0	0	0	1,00
Decision Making - Fuzzy Logic	0	100,0	0	0	0	0	1,00
Knowledge Base	50,0	25,0	25,0	0	0	0	0,75

The feedback on the relevance of symptoms (failures) caused by software faults shows that the most prominent symptom is unpredicted behavior (average score of 3,00) followed by limited functionality (2,69) and crashes (1,94). While most faults of the robot software are classified as repairable (by 40 to 70 % of the groups) problems with OS-related low-level drivers (50,0 %), embedded software in actuators and sensors (46,2 %) and self-localization (37,5 %) are classified mainly as terminal to missions. Moreover, the occurrence of most faults is classified as never and sporadic. Only object tracking (by 30,8 % of the groups), low-level drivers (28,6 %) and computer vision (21,4 %) are classified as regularly.

5.7 Algorithms

Table 3 right shows the feedback on how much various algorithms are affected by faults. Please note that the table is divided into two parts. The algorithms of the upper section are commonly used and we thus got a significant number of replies. The algorithms in the lower section are less common and we thus got only a few replies (2 to 4). Please note that this question regards the algorithm itself rather than its implementation.

The most affected commonly used algorithms are decision making with state machines (average score of 2,67), object recognition in 2D (2,50) and feature extraction (2,00). The upper section of the table clearly shows that these algorithms get higher average score. Even the lowest score of 1,64 for sample-based self-localization is significantly higher than the lower scores for the hardware. These observations lead to the interpretation that algorithms are in general more affected by faults than hardware.

The most relevant cause for faults in algorithms is high computational demands (i.e., missing deadlines). It got an average score of 2,8 out of 5. It is followed by uncertain estimations (2,43), false positives (2,33) and wrong estimations (2,21).

Counting how often groups report causal relations between symptoms (failures) and a particular algorithm sample-based self-localization was named as the most affected algorithm (34 reported relations to some cause) followed by 2D mapping (25) and 2D object recognition (23). Please note that one group may report several causes for a problem of a single algorithm. Configuration problems were most often reported as cause for failures (36 reported relations to some algorithm) followed by wrong estimations (34) and missed computation deadlines (27).

Most of the problems of algorithms were mainly classified as repairable. But problems with decision making using state machines were classified by 50 % of the groups as terminal to the mission. It is followed by plan execution (36,4 %) and 2D mapping (33,3 %). Most of the faults of algorithms occur sporadically. Only 2D object recognition was reported by 45,5 % of the groups as regular problem. Decision making with state machine was reported by 12,5 % of the groups as frequent problem. Please note that classifications of algorithms founded only on 1 to 3 responses were omitted.

5.8 Fault Mitigation Techniques

For this part of the survey we asked about techniques the groups already use to mitigate faults. The questions were divided into techniques that are used off-line prior to the mission or on-line during the mission and into techniques for hardware or software. This section is quite important as it returns information about the state of the art in applied fault mitigation that in turn is close related to dependability.

Table 4 right shows the numbers of groups using particular off-line techniques to mitigate faults in the hardware. The most used technique is testing (used by 76,5 % of the groups) followed by preventive maintaining used by 52,8 % of the groups. Advanced techniques such as finite element method are not used.

Table 4. Groups using off-line techniques to handle hardware faults (left) and on-line techniques to mitigate software faults (right)

Technique	groups use it	
	#	%
Test Process	13	76,5
Preventive Maintenance	9	52,9
Simulation	5	29,4
Special Design Process	4	23,5
Special Implementation Process	4	23,5
Redundant Design	3	17,6
Iterative Design Process	1	5,9
None	1	5,9
Finite Element Method (FEM)	0	0

Technique	groups use it	
	#	%
Process Monitoring	13	76,5
Watchdog	10	58,8
Activity Monitoring	10	58,8
Plausibility Check of Results	7	41,2
Automated Diagnosis	5	29,54
Automated Repair	4	23,5
Automated Reconfiguration	2	11,8
Fault Detection and Isolation (FDI)	1	5,9
Automated Degradation/Adaptation	0	0
None	0	0

Major technique used on-line to deal with hardware faults are watchdogs used by 76,5 % of the groups. It is followed by automated diagnosis used by 29,4 % of the groups. Major techniques used off-line to mitigate software faults are testing (88,2 % of the groups) followed by simulation (76,5 %) and special design processes (47,1 %).

Table 4 left shows the numbers of groups using particular on-line techniques to mitigate faults in the software. It shows that mainly monitoring techniques such process monitoring (by 76,5 % of the groups), watchdogs (58,8 %) and activity monitoring (58,8 %) are used. An interesting aspect is that every group takes measures to deal with these on-line problems suggesting that there is a certain level of awareness to the problems.

Based on additional feedback of the groups using automated diagnosis most groups use some heuristics to check if particular sensors or actuators works correctly and simply reset the device by a command or even a power down/up cycle. One group uses additional sensors to validate results of the self-localization and reinitialize it in case. One group uses an logical framework to reason about undesired states of components or the high-level control and to issue repair actions, e.g., recalibration of an arm [15].

6 Conclusion and Future Work

In this paper we motivated, that in order to maintain a certain level of dependability of autonomous robot systems it is important to identify and to deal with faults of the system. In order to point out which faults of the system are more relevant in terms of their frequency and impact on mission success we adapted a fault taxonomy used for remote controlled robots towards autonomous robots. Moreover, we designed and conduct a survey about the nature of faults in the context of RoboCup. We sent the survey to 68 research groups around the world participating in RoboCup regularly. 17 responses were included in our study. The survey comprised the following parts: information about used hardware and software, hardware faults , software faults, and faults of algorithms as well as used counter-measures. The result of the survey is a database about the nature of faults occurring in autonomous robots. All data are publicly available on the survey website for use by other groups (see `http://www.ist.tugraz.at/rfs`).

We now summarize some of the major observation we can draw from the collected information. Faults in sensors have a similar frequency of occurrence as faults in the robot platforms but their negative impact on the success rate of the mission is much higher. Surprisingly, rather simple causes like connector problems causing hardware failures were reported very often. Therefore, one conclusion is that a high fraction of problems can already be mitigated by better engineering.

The involved groups reported that algorithms are in general more affected by faults than hardware. Basically the awareness of the research groups to the problem of dependability and faults is fortunately quite high. Almost all groups use some techniques to mitigate faults in the robot's hardware and software. Nevertheless, mostly straight forward techniques such as watchdogs or monitoring are used. But the fault properties and the interaction of symptoms and faults ask for advanced fault management.

Failures of algorithms caused by missed deadlines were reported by several of the involved groups. Therefore, any-time or at least predictable algorithms seem to be a promising research direction. Moreover, configuration seems to be a major problem and suggests further research in the direction of configuration management. Furthermore, faults related to the properties of algorithms received high relevance scores asking for more research into the direction of evaluation and validation of algorithms.

The survey provides a first qualitative overview on the nature of faults in the RoboCup context. We are convinced that these observations give information useful for developers and researchers in the area of autonomous robot systems in general. For instance researcher may use the information to concentrate on subsets of fault with high impact on dependability or use discovered symptom-cause relationships to improve their diagnosis models.

In future work we will enhance the survey as for example questions concerning fault mitigation techniques have to be more specific. We plan to conduct the revised survey again in RoboCup and other domains to collect more information to be able to provide significant quantitative results in the future. Moreover, the integration of data from industry is of particular interest. Our vision is that the survey will form a basis for a broader evaluation and classification of faults in robot systems.

Acknowledgment. We like to heartily thank all teams from the RoboCup Middle Size, Robot Rescue and RoboCup@Home leagues who actively took part of this survey. The work has been partly funded by the Austrian Science Fund (FWF) by grant P22690.

References

1. Visser, U., Burkard, H.: RoboCup: 10 Years of Achivements and Future Challenges. AI Magazine 28(2) (2007)
2. Carlson, J., Murphy, R.R.: How UGVs Physically Fail in the Field. IEEE Transactions on Robotics 21, 423–437 (2005)
3. Steinbauer, G., Mörth, M., Wotawa, F.: Real-time diagnosis and repair of faults of robot control software. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 13–23. Springer, Heidelberg (2006)
4. Brandstötter, M., Hofbaur, M., Steinbauer, G., Wotawa, F.: Model-based fault diagnosis and reconfiguration of robot drives. In: IEEE Conference on Intelligent Robots and Systems (IROS), San Diego, CA, USA (2007)
5. Tomatis, N., Terrien, G., Piguet, R., Burnier, D., Bouabdallah, S., Arras, K.O., Siegwart, R.: Designing a Secure and Robust Mobile Interacting Robot for the Long Term. In: Proc. IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan (2003)
6. Bererton, C., Khosla, P.: An analysis of cooperative repair capabilities in a team of robots. In: Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA), vol. 1, pp. 476–482 (May 2002)
7. Verma, V., Gordon, G., Simmons, R., Thrun, S.: Real-time fault diagnosis. Robotics & Automation Magazine 11(2), 56–66 (2004)
8. Long, M., Murphy, R., Parker, L.L.: Distributed multi-agent diagnosis and recovery from sensor failures. In: Int. Conference on Intelligent Robots and Systems (IROS) (2003)
9. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. Intenational Journal of Robotics Research 17, 315–337 (1998)
10. Volpe, R., Nesnas, I.A.D., Estlin, T., Mutz, D., Petras, R., Das, H.: CLARAty: Coupled Layer Architecture for Robotic Autonomy. Technical report, NASA - JPL (2000)
11. Muscettola, N., Nayak, P.P., Pell, B., Williams, B.C.: Remote Agent: to boldly go where no AI system has gone before. Artificial Intelligence 103(1-2), 5–47 (1998)
12. Lauer, M., Riedmiller, M.: Participating in Autonomous Robot Competitions: Experiences from a Robot Soccer team. In: IJCAI 2009 Workshop on Competitions in Artificial Intelligence and Robotics, Pasadena, CA, USA (2009)
13. Kleiner, A., Dornhege, C.: Real-time localization and elevation mapping within urban search and rescue scenarios: Field Reports. Journal of Field Robotics 24, 723–745 (2007)
14. Wisspeintner, T., van der Zant, T., Iocchi, L., Schiffer, S.: RoboCup@home: Scientific Competition and Benchmarking for Domestic Service Robots. Interaction Studies. Special Issue on Robots in the Wild 10(3), 392–426 (2009)
15. Schiffer, S., Wortmann, A., Lakemeyer, G.: Self-Maintenance for Autonomous Robots controlled by ReadyLog. In: Proceedings of the 7th IARP Workshop on Technical Challenges for Dependable Robots in Human Environments, Toulouse, France, pp. 101–107 (2010)

Real-Time Training of Team Soccer Behaviors

Keith Sullivan and Sean Luke

Department of Computer Science, George Mason University
4400 University Drive MSN 4A5, Fairfax, VA 22030 USA
{ksulliv2,sean}@cs.gmu.edu

Abstract. Training robot or agent behaviors by example is an attractive alternative to directly coding them. However training complex behaviors can be challenging, particularly when it involves interactive behaviors involving multiple agents. We present a novel hierarchical learning from demonstration system which can be used to train both single-agent and scalable cooperative multiagent behaviors. The methodology applies manual task decomposition to break the complex training problem into simpler parts, then solves the problem by iteratively training each part. We discuss our application of this method to multiagent problems in the humanoid RoboCup competition, and apply the technique to the keepaway soccer problem in the RoboCup Soccer Simulator.

1 Introduction

In this paper we describe a Learning from Demonstration (LfD) system called *Hierarchical Training of Agent Behaviors*, or HiTAB, and its application to problems in RoboCup. In LfD, an agent learns a behavior in real-time based on provided examples from a human demonstrator, usually through teleoperation of the agent. The goal of HiTAB is to learn complex stateful behaviors in the form of hierarchical finite-state automata (HFA), in real time, based on a small number of samples provided by a demonstrator. HiTAB can be applied both to single-agent training and to command hierarchies of arbitrarily large swarms of agents. We have used HiTAB to train humanoid robots, a team of differential-drive robots, and a variety of virtual agents, up to thousands of agents at a time, on many different problems.

The distinguishing feature of (single-agent) HiTAB is its approach to learning behaviors based on a small number of samples, which in turn enables rapid training in areas, such as behavior-based robotics, where samples are sparse. HiTAB achieves this through manual task decomposition, breaking a complex joint finite-state automaton into a hierarchy of much smaller automata to be iteratively learned and composed. Though HiTAB uses standard classification techniques to learn these automata, the resulting learned automata are often very simple, indeed trivial. This is exactly the goal: simple automata in turn define a low-dimensional space which can be learned with a small number of samples.

All machine learning methods combine some degree of automated machine induction and human domain knowledge. At the very least, a human is choosing an appropriate representation and bias. HiTAB lies at the far end of the induction/knowledge tradeoff. By manually decomposing the problem into a hierarchy of subproblems, the experimenter is defining the automaton's general architecture: HiTAB's machine learning is filling in the gaps. This puts HiTAB somewhere between machine learning and outright programming by demonstration.

In 2011 we applied HiTAB to train a humanoid kid-sized robot soccer behavior the night before the RoboCup competition, then fielded it in the competition alongside hardcoded robot behaviors. Our ultimate goal is to train *all* the top-level behaviors in our robot soccer team while at the competition.

In this paper we demonstrate another application of HiTAB to the Robocup domain: the keepaway problem in simulated soccer, using the RoboCup Soccer Simulator. In this problem, a group of *keepers* must collectively pass the ball amongst one another so as to prevent another team, the *takers*, from acquiring it. This problem requires the experimenter to train a homogeneous but interactive behavior among three agents.

The rest of this paper is organized as follows. We first discuss related work, then detail how HiTAB works in the single-agent case (for details on the multi-agent/swarm case, see [28]). We then discuss our prior and current attempts in the RoboCup Kid-Size Humanoid league. Then, we show how HiTAB may be applied to the keepaway problem in the RoboCup Soccer Simulator.

2 Related Work

Learning from Demonstration is a method to train agents by having a human demonstrator perform actions for the agent [1,2]. Since the agent is given the proper action to perform in a given situation, LfD is, broadly speaking, a supervised learning problem, though authors often use reinforcement learning, with a reward signal based on how closely a learned solution matches a trajectory shown by the demonstrator [7,19]. A variation of LfD, called *imitation learning*, attempts to mimic a demonstrator's actual actions (as a human) rather than observe the demonstrator teleoperate the robot [14,15].

The LfD literature may be divided into two categories: those which learn *plans* [22,31] and those which learn (usually stateless) *policies* [3,19] (for stateful examples see [8,13]). In most cases, the plan literature builds sparse machines describing occasional changes in behavior, whereas many, but not all, policy methods learn fine-grained changes in action, such as might be found in trajectory planning or control. The crucial difference between the two is that a plan learner may receive a new sample only when the user occasionally specifies a new behavior to perform; whereas trajectory policy learners may be inundated with samples with every slight modification or course correction. This in turn has an impact on the difficulty of learning: plan methods must often deal with an extreme sparsity in samples. Our work lies in the plan method category.

Like our own work discussed here, a number of other authors construct complex behaviors via scaffolding: breaking the task into smaller, easier to learn pieces and combining these smaller tasks to form complex behaviors [16,25,27].

Our approach requires manual decomposition and reassembly, but this is not the only approach. Instead of the demonstrator specifying how to combine simpler behaviors, the idea of *behavior fusion* has the agent learn how to automatically combine simple behaviors into more complex behaviors [20,21]. Closely related is the notion of automatic task decomposition which determines how to break complex behavior into simpler components [9,10].

Our work is distinguished in its application to both single- and multi-agent scenarios. Though in this paper we focus largely on single-agent learning, it is done in a collective environment. Multiagent learning from demonstration is a very difficult problem because of the gulf which exists between the desired emergent macrophenomena and the per-agent microbehaviors which give rise to them. This is particularly problematic for supervised learners, because in order to learn in a supervised fashion each agent must receive the *correct action* as a microbehavior: but the experimenter does not know what microbehaviors should be done to achieve the desired macrophenomenon, and parallel control of large numbers of agents is also difficult. As a result the vast majority of multiagent research has focused on reward-based techniques (reinforcement learning, evolutionary computation, etc.) rather than supervised learning [23]. Of those supervised learning methods used, most fall into the category of *agent modeling*, where agents learn about each other rather than a task given by a demonstrator. Still, there has been some work in multiagent LfD. Chernova et al. use confidence estimation to train multiple robots individually and rely on emergent multirobot behavior to accomplish the task [5,6]. A similar approach was used to train Sony AIBO robots to play soccer [4,11,12].

3 Hierarchical Training with a Single Agent

HiTAB's basic model consists of hierarchical finite-state automata. Each state in a HiTAB automaton corresponds to an agent behavior: and when in a given state, the agent performs the associated behavior. Behaviors may be either atomic behaviors hard-coded in the agent, or may themselves be other finite-state automata. Every automaton begins in its *Start* state, a blank state which immediately transitions to some other state. Automata may also have *flag states*, such as the *Done* state, which raises a flag indicating that the automaton believes it is done, then transitions to the *Start state.* Flag states allow parent automata to detect completion of sub-behaviors as if they were sensor features.

Transitions between states are controlled by *transition functions* which map the current state and feature vector to a new state. HiTAB's states are fixed (they are the current behaviors in its library), but it learns a transition function for every state in the automaton.

Learning the transition function is a classification task where the class labels are the individual states and attributes are the environmental features. While many classification algorithms are applicable, HiTAB at present uses a version of the C4.5 decision tree algorithm [24] with probabilistic leaf nodes. Decision trees nicely handle different types of data (e.g., continuous, toroidal, and categorical data), and do not require scaling of features relative to one another. Additionally, many agent tasks can be approximated by rectangular partitions of the feature space, which makes them a good target for decision trees. Leaf nodes in decision trees traditionally deterministically compute the class using the plurality of examples which reach that leaf. HiTAB instead uses a probability distribution over the classes appearing at a leaf node.

The motivation behind HiTAB was to develop a LfD system which could rapidly train complex, stateful agent behaviors in real time. As mentioned before, training complex agent behaviors typically requires many samples. HiTAB employs task decomposition to reduce the number of samples necessary to produce a detailed behavior. It does this in various ways:

– Behaviors (which take the form of finite-state automata) are organized into a hierarchy, allowing the operator to decompose a large joint behavior into many simpler behaviors which are trained independently, then reused in different situations by higher-level trained behaviors.
– Each behavior may be trained solely in the context of features and lower-level behaviors relevant to it. In contrast, training a single large behavior would require the joint of all basic behaviors and features, resulting in a much higher dimensional learning space. This results in dramatic savings: typically decomposition allows the dimensionality, and corresponding need for samples, to decrease from exponential to polynomial sizes.
– Behaviors and sensor features are parameterizable. Thus an operator may train a behavior such as *go to X*, and later reuse it as *go to the ball* or *go to the nearest wall*, etc.
– Incorrectly trained behaviors may be retrained without having to retrain the entire top-level joint behavior.

Running HiTAB. An automaton starts in its *Start* state. Each timestep, while in state S_t, the automaton first queries the transition function to determine the next state S_{t+1}, transitions to this new state, and if $S_t \neq S_{t+1}$, stops performing S_t's behavior and starts performing S_{t+1}'s behavior. It then performs one pulse of the state's underlying behavior: if the behavior is an atomic behavior such as "go forward", this might result in a single step forward. If the behavior is itself an HFA, this results in recursively performing the aforementioned transition and pulsing procedure on the underlying automaton.

Training with HiTAB. To begin training an automaton, the operator first selects the features to be used as attributes for the automaton's transition classifiers. Training then iterates between a *training mode* and a *testing mode.*

In the training mode, the demonstrator is in control. Each time the demonstrator directs the agent to perform a new behavior, the agent begins performing it, and also records a tuple $\langle S_t, \vec{f_t}, S_{t+1}\rangle$ which stores the current feature vector, along with the previous and new states. If state S_{t+1} has a behavior designed to be executed exactly once, then no additional examples are recorded. Otherwise, a useful *default example* is stored of the form $\langle S_{t+1}, \vec{f_t}, S_{t+1}\rangle$. This helps

HiTAB's classifier realize that S_{t+1} should be continuously performed unless, as indicated by a further example, the situation changes again.[1]

Ultimately the demonstrator switches to the *testing mode*, which causes the transition functions to be built from the collected examples. For a given state S_i, HiTAB reduces all examples of the form $\langle S_i, \vec{f_t}, S_j \rangle$ to samples of the form $\langle \vec{f_t}, S_j \rangle$ which are input to the classifier ($\vec{f_t}$ are the features and S_j are the labels). The resulting classifier defines the transition function for outgoing edges from S_i.

The agent then starts following the learned behavior autonomously. If the demonstrator observes the agent performing an incorrect behavior, he may step in and switch the agent back to training mode to collect additional examples.

Ultimately the trained behavior is saved to the behavior library. To do this, HiTAB first trims unused states and features. In addition, any parameterized behaviors and features are bound to a target (e.g., "nearest obstacle"), or to a parameter of the automaton itself. After saving to the library, the behavior may be used as a state in a higher-level automaton to be learned at a later time.

Formal Model. The HFA is at the heart of HiTAB. An automaton is a tuple $\langle S, B, F, T, G \rangle \in \mathcal{H}$ defined as follows:

- $S = \{S_1, \ldots, S_n\}$ is the set of *states* in the automaton. Included is one special state, the *Start* state S_0, and zero or more *flag states* (such as *Done*). Exactly one state is active at a time, designated S_t. The purpose of a flag state is simply to raise a flag in the automaton to indicate that the automaton believes that some condition is now true. Flags in an automaton appear as optional features in its *parent* automaton.
- $B = \{B_1, \ldots, B_k\}$ is the set of *basic behaviors*. Each state is associated with either a basic behavior or *another automaton* from $\mathcal{H}$, though recursion is not permitted.
- $F = \{f_1, \ldots, f_m\}$ is the set of observable *features* in the environment. At any given time each feature has a numerical value. The collective values of F at time t is the environment's *feature vector* $\vec{f_t} = \langle f_1, ..., f_m \rangle$.
- $T = \vec{f_t} \times S \rightarrow S$ is the *transition function* which maps the current state S_t and the current feature vector $\vec{f_t}$ to a new state S_{t+1}.
- Optional free variables (parameters) $G = \{G_1, \ldots, G_n\}$ for basic behaviors and features generalize the model: each behavior B_i and feature f_i are replaced as $B_i(G_1, \ldots, G_n)$ and $f_i(G_1, \ldots, G_n)$. The evaluation of the transition function and the execution of behaviors are based on ground instances of the free variables. For example, rather than have a behavior called *go to the ball*, we can create a behavior called *goTo(A)*, where A is left unspecified. Similarly, a feature might be defined not as *distance to the ball* but as

[1] Default examples are distinguished in HiTAB's decision tree mechanism: if the decision tree is choosing to place its pivot between a default example and a non-default example, the pivot is placed immediately adjacent to the non-default example. This differs from the normal case, where the pivot is placed exactly half-way between the two examples.

distanceTo(B). If such a behavior or feature is used in an automaton, either its parameter must be bound to a specific *target* (such as "the ball" or "the nearest obstacle"), or it must be bound to some higher-level parent of the automaton itself. Thus HFAs may themselves be parameterized.

4 Training Teams of Agents

We have applied HiTAB in three ways to train teams or swarms of agents to perform group behaviors:

1. A single agent behavior is trained in isolation, then distributed to multiple agents. The behavior does not require agent interaction and can be essentially done in parallel.
2. A homogeneous behavior is trained to be used by multiple coordinated agents. For example, the agents learn to form ranks, or work together to capture a prey. Because the behavior must interact with other agents, this kind of training can be challenging. In lieu of training multiple agents simultaneously, we have taken a new approach, which we call *behavior bootstrapping*. Here, we train an agent to perform a rudimentary version of the desired behavior in the context of do-nothing teammates. We then distribute this rudimentary behavior to the teammates, then train the agent on a slightly more capable behavior in the context of teammates performing the rudimentary behavior. We then distribute the slightly more capable behavior to the teammates, and train an even more capable behavior, and so on, until the desired sophisticated behavior is achieved. This approach is only really effective with a relatively small number of agents.
3. A collection of coordinated homogeneous behaviors are trained among a swarm of a (potentially very large) number of agents. The way this is done is by organizing the swarm into a command hierarchy: small groups of agents are assigned a commander (a virtual agent); then small groups of commanders are assigned a commander, and so on until the whole swarm is structured as a tree. We use HiTAB to train commanders in essentially the same way as real (leaf node) agents are trained. A commander's atomic behaviors correspond to the learned top-level behaviors of its subordinate agents, and when it begins to perform an atomic behavior it directs its subordinates to all begin performing the equivalent top-level behavior. The resulting hierarchical command structure strikes a mid-ground between a fully distributed swarm and a fully centralized one.

Examples of the third approach may be found in [28]. Because the number of agents is small (three teammates), in this paper we concentrate on the first two approaches, and particularly on the novel use of behavior bootstrapping to train three agents in concert.

We note that these methods, or at least the last two, fall under the multiagent learning subcategory defined in [23] as *team learning*, whereby a single learner is used at any particular time to train a team of agents. This is in contrast to *concurrent learning*, where multiple learners are simultaneously operating. Further, we note that the group behaviors described above are all homogeneous. However ultimately we aim to be able to train heterogeneous or mixed homogeneous and heterogeneous behaviors in large numbers of agents.

5 Team Robot Training of Humanoids at RoboCup

The goal of HiTAB is to allow real-time training of behaviors fast enough that it can be done in the field and on-the-fly by an operator. This has been demonstrated in previous work [18,29] for virtual agents, a single differential-drive robot, and a humanoid robot. But it had never been tested in a real-world challenge scenario. Thus as a proof of concept we fielded HiTAB-trained robots in RoboCup 2011.

Since 2009, we have competed in the RoboCup Kid-sized Humanoid League with the RoboPatriots [30]. Our humanoid robots have top-level behaviors in the form of hard-coded hierarchical finite-state automata. Such behaviors include locating the ball, servoing and approaching the ball, aligning with the goal, kicking and reattempting kicks, and so on.

On the soccer field the night before the 2011 competition, we deleted one of the hard-coded behaviors (servoing and approaching the ball) and trained a behavior in its place. We did this by directly teleoperating the humanoid on the field. We then saved out the trained behavior, and during the competition, the robots loaded this behavior from a file and used it in an interpreter along side the remaining hard-coded behaviors.

This behavior was not complex: it was meant as a proof of concept. However, the learned behavior worked perfectly. After discussions with colleagues at the competition, we have come to the conclusion that, to the best of our knowledge, this is the first time a team at RoboCup has used a behavior taught to the robots on the field at the competition itself.

For RoboCup 2012, our goal is to train most, if not all, of the top-level behaviors on the field at the competition. In essence, we will attempt to teach the team how to play soccer the night before the competition.

6 Team Robot Training of Keepaway Soccer

In preparation for the RoboCup 2012 humanoid goal, we applied HiTAB to the task of simulated soccer keepaway in the RoboCup Soccer Simulator. In the keepaway problem, a team of *keepers* tries to maintain possession for as long as possible from a team of *takers*. The two teams compete in a bounded area (in our case, a 20m $\times$ 20m box) within a regular soccer field in the RoboCup Soccer Simulator. In our version of keepaway, the agents have 360 degree and infinite view and cannot collide with the ball. We did not permit our keepers to dribble.

The keepaway problem presents several challenges. First, its limited inter-agent communication requires agents to learn independently, but the resulting

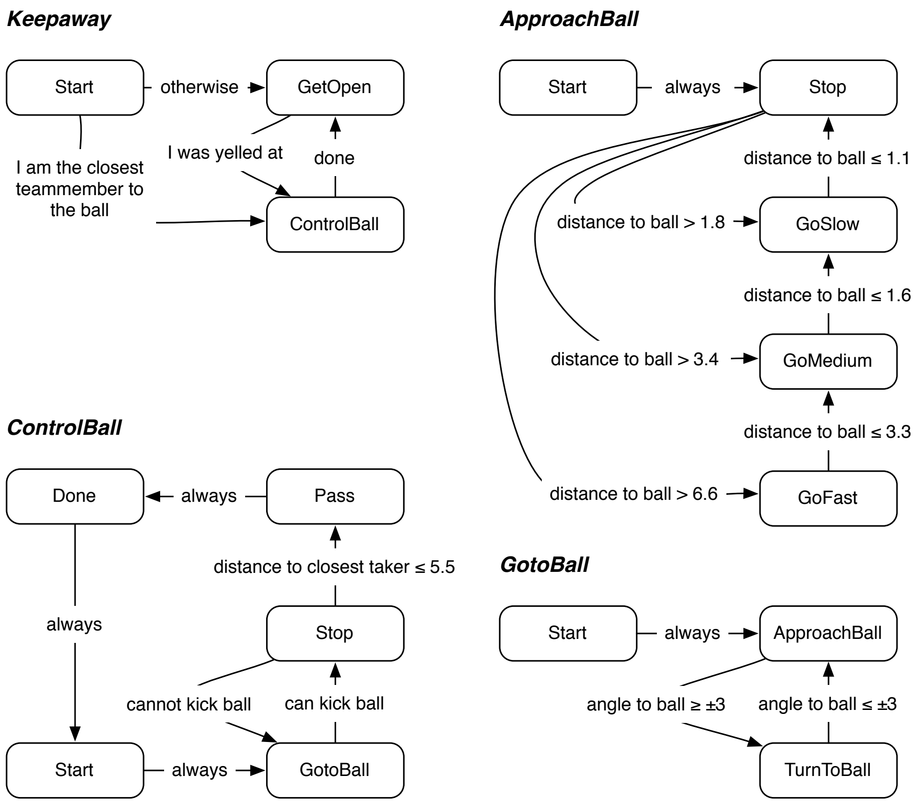


Fig. 1. Four automata trained for the Keepaway Problem. In each case, the automaton begins in *Start*. The *Done* behavior does nothing but raises a *done* flag in the automaton's parent, which is detected by the *done* feature (compare ControlBall with Keepaway). Real-valued numbers shown are the result of the training examples provided.

behaviors require coordination. Second, keepaway (and soccer in general) has a large state space. Third, the RoboCup Soccer Simulator injects random noise in agents' actions and sensors.

In this example we used HiTAB plus behavior bootstrapping to learn coordinated behaviors among the three agents. We first manually decomposed the keeper behaviors into a structure similar to Stone et al [26]. The keepers were provided with the following hard-coded behaviors: *GoFast, GoMedium, GoSlow, Stop, Pass, GetOpen,* and *TurnToBall.*

- The *GoFast, GoMedium,* and *GoSlow* functions moved the keeper straight ahead at velocities of 100, 90, and 75 respectively.
- The *Pass* function kicked the ball to the "most open" teammate. Openness was defined by determining the maximum angle subtended by the vector between the passer and receiver, and the vector between the passer and the closest taker. Kick strength accounted for friction and was proportional to

the distance between the passer and receiver[2]. The passer would then yell to the receiver to inform him of the incoming ball.
- *GetOpen* moved the keeper away from its teammates via a simple potential field, but constrained to be within 10 meters of the center of the box.
- *TurnToBall* rotated the keeper to directly face the ball.

The features we used were: *DistanceTo(X), DirectionTo(X), IWasYelledAt, BallIsKickable,* and *ATeammateIsCloserToBall*, where X could be set to either the ball, the closest keeper, or closest taker. The binary features *IWasYelledAt, BallIsKickable* and *ATeammateIsCloserToBall* were true when a yell message was received (from a passer), the ball was within kicking range, or another keeper was closer to the ball, respectively, and were false otherwise.

Given these basic behaviors and features, we trained four automata in the following order, as shown in Figure 1:

1. **ApproachBall:** A P-Controller in automaton form, based on *GoFast*, *GoMedium*, *GoSlow*, *Stop*, and *DistanceTo(ball).* This automaton attempted to move the agent until it was within kicking distance of the ball location.
2. **GotoBall:** Iterated between *ApproachBall* and *TurnToBall*, using the angle to ball. This automaton attempted to servo on the ball location, and did not require state.
3. **ControlBall:** Used the *GotoBall*, *Stop*, and *Pass* behaviors, the optional *Done* state, and the *DistanceTo(Closest Taker)* and *BallIsKickable* features. This automaton servoed on the ball, waited until a taker was sufficiently close, then passed the ball, plus some error handling.
4. **Keepaway:** The top-level automaton, used *GetOpen* and *ControlBall*, plus three features: *ATeammateIsCloserToBall*, *IWasYelledAt*, and *Done.* This automaton would initially either get open or take control of the ball depending on whether the agent was initially closest to the ball. It then iterated between the *GetOpen* and *ControlBall* behaviors depending on whether the agent believed it was in control of the ball at any given time.

Keepaway was notable in that it was trained via a simple form of behavioral bootstrapping. We began by training a single agent to either go to *ControlBall* or *GetOpen* when started. We then distributed this behavior to all the agents. We next restarted the game, which caused one agent to go to *ControlBall* while the others went to *GetOpen.* We further trained the ball-controlling agent to pass the ball and then get open, and then copied that behavior to all agents. After restarting again, we trained a single open agent to transition to *ControlBall* when yelled at, and distributed the final version of the behavior.

7 Experiments

We ran our learned keepaway behaviors for 200 episodes. An episode ended when the takers gained possession of the ball, or when the ball was kicked out of the

[2] The exact kick strength computation followed the U Texas Austin code used in `http://www.cs.utexas.edu/~AustinVilla/sim/keepaway/`.

Table 1. Number of data points to train the final behaviors, and an approximation of the total time to train the final behaviors.

Behavior	Number of Examples	Time to Train (minutes)
ApproachBall	18	10
GotoBall	10	10
ControlBall	11	45
Keepaway	10	90

keepaway box. The takers were from [26], but running at one quarter the speed of the original. All experiments were conducted using the MASON multiagent simulation package [17] (running HiTAB) plus the RoboCup Soccer Simulator.

Our trained keepers maintained possession for an average of 14.6 ± 0.87 seconds, and completed an average of 3.8 passes per episode. We were able to train the behaviors to play successfully: but obviously they will require more tweaking to keep the ball away from the takers for a longer duration.

We also wanted to examine how quickly behaviors could be trained using HiTAB. Table 1 shows the length of time spent actually training the agents (including collecting the samples and constructing the HFA), and the number of examples collected for the final trained model. Typically, the demonstrator required several iterations to train the final behavior due to demonstrator error or experimentation with different ways of achieving the desired behavior (and thus different automata structures). *Keepaway* took longer to train due to the behavioral bootstrapping involved. In particular, the majority of the time was spent determining how to manage the system such that two agents were in the correct configuration to collect appropriate data: inability to manipulate the agents was largely a GUI issue which can be remedied in the future.

We believe the experiments show HiTAB's ability to train a complex multiagent behavior in a reasonable timeframe, and without requiring a significant amount of data. Based on these results, we think our goal to train the RoboPatriot soccer behavior in Mexico City is viable.

8 Conclusions and Future Work

This paper demonstrated a supervised learning from demonstration system capable of training complex behaviors in a multiagent problem domain in real time. Our system, HiTAB, achieves this through manual behavior decomposition, per-sub-behavior feature reduction, and machine learning through classification. HiTAB's purpose is to do learning on a very small number of samples. Its use of behavior decomposition places us on the far end of what may be reasonably called machine learning, and very close to explicit programing by example. Multiagent supervised training (as opposed to user modeling) is unusual, and HiTAB is nearly unique in tackling this problem.

The primary difficulties we encountered in adapting HiTAB for the soccer keepaway problem centered on representation: HiTAB employs classification rather than regression, yet many of the behaviors in the robot soccer domain benefit from regression. For example, the *GetOpen* behavior computed the direction to go based on a potential field, which HiTAB could not easily do. Intelligent interception would also benefit from regression, as was originally demonstrated in [27]. It is reasonable to use HiTAB to train higher-level behaviors composed from lower-level behaviors which were either hard-coded or developed through another learning technique (such as a regression technique). Finally, our ultimate goal is to develop HiTAB towards heterogeneous multiagent behaviors. In the keepaway problem there is little need for heterogeneity: but in the Robocup Humanoid leagues it is plausible for all three robots to be heterogeneous, either by differences in capability (goalies) or simply behavior (a forward versus a midfielder).

References

1. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robotics and Autonomous Systems 57, 469–483 (2009)
2. Atkeson, C.G., Schaal, S.: Robot learning from demonstration. In: Fisher, D.H. (ed.) Proceedings of International Conference on Machine Learning (ICML), pp. 12–20. Morgan Kaufmann (1997)
3. Bentivegna, D.C., Atkeson, C.G., Cheng, G.: Learning tasks from observation and practice. Robotics and Autonomous Systems 47(2-3), 163–169 (2004)
4. Browning, B., Xu, L., Veloso, M.: Skill acquisition and use for a dynamically-balancing soccer robot. In: Proceedings of the American Association of Artificial Intelligence (AAAI), pp. 599–604 (2004)
5. Chernova, S.: Confidence-based Robot Policy Learning from Demonstration. Ph.D. thesis, Carnegie Mellon University (2009)
6. Chernova, S., Veloso, M.: Confidence-based multi-robot learning from demonstration. International Journal of Social Robotics 2, 195–215 (2010)
7. Coates, A., Abbeel, P., Ng, A.Y.: Apprenticeship learning for helicopter control. Communications of the ACM 52(7), 97–105 (2009)
8. Dixon, K., Khosla, P.K.: Learning by observation with mobile robots: A computational approach. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA) (2004)
9. Eyharabide, V., Amandi, A.: Automatic task model generation for interface agent development. Inteligencia Artificial 9(26), 49–57 (2005)
10. Garland, A.: Learning hierarchical task models by demonstration. Tech. Rep. TR-2001-03, Mitsubishi Electric Research Laboratories (2001)
11. Grollman, D., Jenkins, O.: Dogged learning for robots. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 2483–2488. IEEE (2007)
12. Grollman, D.H., Jenkins, O.C.: Can we learn finite state machine robot controllers from interactive demonstration? In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 407–430. Springer, Heidelberg (2010)
13. Hovland, G., Sikka, P., McCarragher, B.: Skill acquisition from human demonstration using a hidden markov model. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 2706–2711. IEEE (1996)

14. Ijspeert, A.J., Nakanishi, J., Schaal, S.: Movement imitation with nonlinear dynamical systems in humanoid robots. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pp. 1398–1403 (2002)
15. Jenkins, O., Mataric, M., Weber, S.: Primitive-based movement classification for humanoid imitation. In: Proceedings of the IEEE-RAS International Conference on Humanoid Robotics (Humanoids) (2000)
16. Lockerd, A., Breazeal, C.: Tutelage and socially guided robot learning. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), vol. 4, pp. 3475–3480. IEEE (2004)
17. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: A multiagent simulation environment. Simulation 81(7), 517–527 (2005)
18. Luke, S., Ziparo, V.: Learn to behave! rapid training of behavior automata. In: Grześ, M., Taylor, M. (eds.) Proceedings of Adaptive and Learning Agents Workshop at AAMAS 2010, pp. 61–68 (2010)
19. Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., Kawato, M.: Learning from demonstration and adaptation of biped locomotion. Robotics and Autonomous Systems 47(2-3), 79–91 (2004)
20. Nicolescu, M., Jenkins, O., Olenderski, A.: Behavior fusion estimation for robot learning from demonstration. In: Proceedings of Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications. IEEE Computer Society (2006)
21. Nicolescu, M., Jenkins, O., Stanhope, A.: Fusing robot behaviors for human-level tasks. In: Proceedings of the International Conference on Development and Learning (ICDL), pp. 76–81. IEEE (2007)
22. Nicolescu, M.N.: A Framework for Learning from Demonstration, Generalization and Practice in Human-Robot Domains. Ph.D. thesis, University of Southern California (2003)
23. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems 11(3), 387–434 (2005)
24. Quinlan, J.R.: C4.5: Programs for Machine Learning, 1st edn. Morgan Kaufmann Series in Machine Learning. Morgan Kaufmann (January 1993)
25. Saunders, J., Nehaniv, C., Dautenhahn, K.: Teaching robots by molding behavior and scaffolding the environment. In: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI) (2006)
26. Stone, P., Sutton, R.S., Kuhlmann, G.: Reinforcement learning for RoboCup-soccer keepaway. Adaptive Behavior 13(3), 165–188 (2005)
27. Stone, P., Veloso, M.: Layered learning and flexible teamwork in robocup simulation agents. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup 1999. LNCS (LNAI), vol. 1856, pp. 495–508. Springer, Heidelberg (2000)
28. Sullivan, K., Luke, S.: Learning from demonstration with swarm hierarchies. In: Proceedings of Autonomous Agents and Multi-Agent Systems (AAMAS) (2012)
29. Sullivan, K., Luke, S., Ziparo, V.A.: Hierarchical learning from demonstration on humanoid robots. In: Proceedings of the Humanoid Robots Learning from Interaction Workshop at Humanoids (2010)
30. Sullivan, K., Russell, K., Andrea, K., Stout, B., Luke, S.: RoboPatriots: George Mason University 2012 RoboCup team. In: Proceedings of the 2012 RoboCup Workshop (2012)
31. Veeraraghavan, H., Veloso, M.M.: Learning task specific plans through sound and visually interpretable demonstrations. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 2599–2604. IEEE (2008)

SLAM in the Dynamic Context of Robot Soccer Games

Stefan Tasse, Matthias Hofmann, and Oliver Urbann

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

Abstract. This paper evaluates the benefits of modeling the dynamic environment of robot soccer games as a SLAM problem. Moving objects such as other robots and the ball are not only tracked individually, but modeled in a full state and used for localization at the same time. This is described as an implementation of an efficient system capable of running in real time on limited platforms such as the humanoid robot Nao, and the system's benefit is evaluated using real world experiments.

1 Introduction

For an autonomous robot, knowing its location and the current state of the environment is essential as it represents the basis for planning and reactive behavior. Generally, this task may involve different aspects: Localization in known environments, mapping previously unknown environments, and tracking dynamic elements therein. In most common literature [1], those aspects are handled separately. Either a robot is only expected to localize in a known structured environment, leaving a choice among many existing solutions which mostly differ in their suitability for different kinds of uncertainties occurring during the robot's operation, or no prior knowledge exists at all as assumed in simultaneous localization and mapping (SLAM) scenarios, which represents the other extreme. Dynamic elements are generally ignored in both cases, either handled as noise or explicitly filtered out [2,3].

Real world applications, however, always consist of a mixture of all those aspects. Many features in the robot's area of operations will be known beforehand, either from floor plans for indoor, or aerial photographs for outdoor scenarios, or from previous mapping efforts or other given specifications. On the other hand, such a priori information rarely covers all features which are of interest for the localization task. Some might even be occluded due to recent changes in the environment. Incorporating new information into a robot's internal map can therefore often improve its ability to localize precisely. Similarly, it can be shown that explicitly incorporating dynamic features into the system improves the estimation quality, both for the tracking result and the localization [4].

In the course of this paper the modeling of the full state in dynamic situations will be covered. Section 2 gives an analysis and a brief overview over related work.

Section 3 describes the actual implementation of such a system with the practical approximations necessary to reduce the problem's complexity to a point where it is applicable in real-time on limited embedded platforms, specifically on the Nao V3.3 robot. An evaluation is given in section 4 in the robot soccer context of the RoboCup Standard Platform League (SPL), and the paper is concluded in Section 5.

2 Modeling

This section addresses the different aspects of the problem in the context of a team of cooperating autonomous agents acting in highly dynamic competitive environments, therefore settling certain design choices as the basis for the implementation of the system described in section 3.

2.1 Localization and Robot-Centric Tracking

The computational complexity of common filtering approaches naturally increases with the state's dimensionality. Separating the estimation of the robot's own position and that of the positions of other surrounding elements is therefore motivated from a performance standpoint. Moreover, tracking dynamic objects by stationary observers is a widely explored problem. Those are the main reasons why the tracking problem for autonomous robots is commonly done in a robot-centric local coordinate system.

The modeling of the localization and tracking aspects as a unified global estimation problem, however, has some advantages compared to simple tracking in robot-centric local coordinates. The latter necessitates the update of all tracked objects with odometry data, which e.g. for humanoid robots can be extremely unreliable. Those propagate nearly uncorrected into infrequently observed targets, leading to significant drift. At the same time, the separate robot pose estimation already corrected part of the odometry's errors and prevented the same drift in its estimate. This motivates the advantage of modeling dynamic objects in global reference systems. Additionally, information about tracked objects can be beneficial for the localization when modeled in a unified state. Even if the objects' motion uncertainty prevents their use for accurate localization, shared information about those objects among a team of agents might still resolve multi-modal or symmetrically ambiguous localization states.

This leads to a heterogeneous system which has some resemblance to the SLAM problem, since new features are mapped and used for localization at the same time. In previous literature pure localization and the full SLAM problem have been mostly separated. Only recent publications began exploring the intermediate between those extremes by incorporating a-priori information into systems otherwise formulated as SLAM, e.g. in [5] where a SLAM approach is augmented by a-priori information in form of aerial images. Moving objects are just considered to be obstructive in normal SLAM algorithms and either handled as noise [2] or tracked in separate model to be filtered out of the SLAM input, therefore without any direct positive effect on the SLAM output [3].

An alternative system providing the same characteristics as the one proposed here has already been published in [4]. It describes an adaptation of the Fast-SLAM concept to include a-priori information as well as newly mapped static and dynamic features with different degrees of uncertainty in their recognition processes and motion models. The approach presented here differs from the one in [4] in its vast use of Kalman filters in all different stages of the system, whereas the latter integrates the SLAM aspects by use of a Rao-Blackwellized particle filter. Notably, the system presented here can run in real-time on a Nao robot in parallel to all other modules necessary to participate in RoboCup SPL games.

2.2 Heterogeneous Information Sources

As stated so far, the proposed system should use information about previously known and previously unknown, static and dynamic features, and incorporate all those into a coherent estimate of both the robot's own positions as well as the potentially dynamic states of the other objects. This obviously implies various different, and more importantly, heterogeneous information sources.

Distinctions can be made according to the characteristics of each feature, whether it can be used for localization directly or needs to be mapped, too, either as a static but previously unknown feature or a dynamic one including motion updates. A features associated uncertainty can vary both with respect to the reliability and precision of its observation and the inherent predictability of its motion model. Simple in-animated objects for example may just follow physical equations of motion. Other autonomous agents on the other hand may change their intention and action unpredictably, while being harder to measure reliably due to their more complex shape, varying silhouette and changing backgrounds.

Each such distinction offers the possibility to apply approximations without losing too much precision in the estimation result. A more thorough analysis on the implication for those heterogeneous information sources and possible approximations can be found in [4]. In the implementation described in this paper, only a subset of those is employed.

The most relevant approximation in the context of this paper is the aggregation of measurements to build local short-term models of each observation type, thereby decreasing the uncertainty associated with the observed target and allowing to filter false positives. Once sufficiently recognized such a short-term model can be forwarded to the central estimation system as a meta-measurement and deleted from the temporary local model. The deletion of such models is important to preserve the independence assumption between consecutive measurements which is important for the Bayes filter concept. Insufficiently validated local hypotheses on the other hand can be pruned away without effecting a negative influence on the system's estimate. The short life span of those local models, e.g. below one second, prevents odometry errors to accumulate, but often allows the integration for example of a series of image processing results to obtain superior measurement quality.

2.3 Distributed Modeling

The sharing of information among a team of autonomous agents is especially desirable in cases where single robots have a very limited field of view and when occlusion frequently occurs. The distribution of information can be done with two conceptually different approaches. One approach can be classified as bottom-up and distributes measurements between robots, which are subsequently handled by common sensor fusion techniques. This is for example done in [6] and [7]. The top-down approach as applied in [8] and [9] consists of merging the individual robots' world maps. The prerequisite for such map merging is that all poses of participating robots need to be known, either in a consistent global coordinate frame or relative to each other. A common implementation in exploration scenarios is with uniquely identifiable robots which initiate map merging when observing each other, or when all robots are confidently localized in the global reference frame.

This latter approach would exclude poorly localized robots from map merging, however, those might also profit from the shared information, even specifically to resolve their poor localization in case of symmetries. If the measurements are distributed among robots, basically only each sending robot needs to be localized successfully in a global reference frame. An additional advantage is the computational and architectural simplicity of observation distribution compared to map merging, especially if observations are already aggregated in temporary local models as described in section 2.2, which in case of reliable localization can be distributed at the same time when used for the local integration into the global world model. Note that this approach does not guarantee a globally consistent model among all robots, since insufficiently localized robots do not send out information and therefore integrate more (but potentially also more unreliable) knowledge. The difference between the models of well localized robots however can be minimized by globally scheduling the exchange and integration phases of the observation distribution [4].

3 Implementation

The objective now is the realization of the demands specified in section 2, namely to model the robot's surrounding environment in one unified model using the information of a whole team of robots as input. This is hardly possible to implement as a real-time system on an embedded platform without applying measures to decrease the computational complexity. The presented approach consists of three stages, which will be covered in the following sections. The first stage handles the static map information to realize most (but not all) of the localization problem, and is based on an algorithm which can perform as a very efficient stand-alone localization [10]. Parallel to this runs a stage performing local percept aggregation according to the temporary short-term models described in section 2.2. Finally section 3.3 presents the integration of local and distributed perceptions into a consistent global world model.

In the following, each observation of a feature is represented by the two angles $z = (\alpha_1, \alpha_2)^T$ describing the direction in which the feature has been detected, and a third angle α_3 in case the feature has an identifiable orientation relative to the robot coordinate system. This is visualized in figure 1.

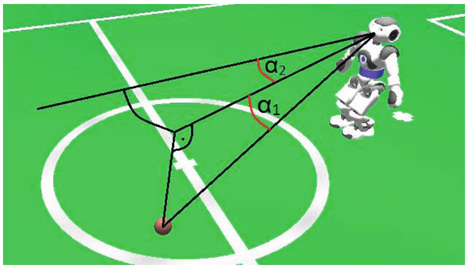


Fig. 1. Measurement of a feature expressed in horizon aligned observation angles

3.1 Multi-model Kalman Localization

In contrary to the system described in [4], which is based on a particle filter localization, the approach presented here bases on a multi-hypothesis Unscented Kalman Filter (UKF) localization which has been presented in [10]. This utilizes an approach to Gaussian mixture filtering which combines the accuracy of the Kalman filter and the robustness of particle filters without sacrificing computational efficiency. This is done by pointing out similarities to particle filtering with an extremely low number of particles, and bypassing critical approximations in common Gaussian mixture algorithms.

Applying known techniques from both fields in a new combination results in a multi-hypotheses Kalman filter which is superior to common Kalman filters in its ability of fast re-localization in kidnapped robot scenarios and its representation of multi-modal belief distributions, and which outperforms particle filters in localization accuracy and computational efficiency. The output of this system is a set of robot pose hypotheses with corresponding covariances and likelihood estimations. If this is used as a separate localization module, the most likely hypothesis can be considered as the localization's result, and used as an input for behavior decisions or further planning.

To use this in the context of a unified world model it is necessary to keep track of the history of each hypothesis' origin for fusion and spawning of new hypotheses, and the change of the likelihoods among the set of hypotheses, which corresponds to a re-localization event for example with a kidnapped robot or after temporary localization loss caused by extreme odometry errors or collisions. Otherwise each estimate's change can be considered as a pre-filtered input for the global estimation system. This input bears the characteristics, on the one hand, of partially corrected odometry data, and on the other hand that one of buffered and pre-processed sensor data. In addition to this, integration of further information, including the communicated observations of other robots, can affect the pose estimates, so those changes need to be fed back into the localization module. The following sections will address the integration into the global model and the stochastic soundness of this.

3.2 Local Percept Aggregation

When building upon the UKF localization described above, the full state can not be factorized as in FastSLAM, but needs to be expressed as a joint probability function, as in the EKF-SLAM solution. The increase of estimation complexity by the high-dimensional state is countered by aggregation of some of the image processing results into temporary percept-buffers as motivated in section 2.2 with the aforementioned advantages.

This is applied to full extend to the dynamic features, i.e. the ball and other robots in a robot soccer scenario. Measurements of robots or the ball consist of two angles $z = (\alpha_1, \alpha_2)^T$ as described above. The estimated state consists of a 2-dimensional spacial component and a corresponding velocity component: $\mu = (x, y, v_x, v_y)^T$. For the spacial component of the state $\mu' = (x, y)^T$ and the height of the robot's camera r, equations 1 and 2 can be used to calculate the sensor model in form of the observation matrix H as in equation 3. Note that the velocity is not observable by processing a single camera image.

$$h_{\alpha_1}(\mu) = \text{atan2}(r, |\mu'|) \tag{1}$$

$$h_{\alpha_2}(\mu) = \text{atan2}(y, x) \tag{2}$$

$$H = \begin{pmatrix} -\frac{rx}{|\mu'|^3+r^2|\mu'|} & -\frac{ry}{|\mu'|^3+r^2|\mu'|} & 0 & 0 \\ -\frac{y}{x^2+y^2} & \frac{x}{x^2+y^2} & 0 & 0 \end{pmatrix} \tag{3}$$

For objects which are simply governed by the physical laws of motion, instead of being motorized or controlled, the motion model for the control update consists of a continuous motion slowed down by a friction factor $k = \frac{F_{friction}}{m}$ as the force generated by the friction divided by the mass of the object. Since the state is modeled in local coordinates, the robot's own motion, given by the translational and rotational odometry $(\delta_x, \delta_y, \delta_\theta)$, also transforms the local estimate. This results in the following time update for the velocity vector:

$$v_t = \begin{cases} \overbrace{\left(1 + \frac{k\Delta t}{|v_{t-1}|}\right) \Omega(-\delta_\theta)}^{=:V} \, v_{t-1} \text{ for } |v_{t-1}| \geq |k\Delta t| \\ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} v_{t-1} \text{ else} . \end{cases} \tag{4}$$

where $\Omega(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$ is the rotation around α. The full time update therefore predicts the state μ_{t-1} according to equation 5.

$$\overline{\mu}_t = g(\mu_{t-1}) = \begin{pmatrix} \Omega(-\delta_\theta) & \Delta t \Omega(-\delta_\theta) \\ 0 & V \end{pmatrix} \mu_{t-1} - \begin{pmatrix} \delta_x \\ \delta_y \\ 0 \\ 0 \end{pmatrix} \tag{5}$$

This results in the Jacobi matrix G for the process update as the partial derivatives of x,y,v_x and v_y at $(x_{t-1}, y_{t-1}, v_{x,t-1}, v_{y,t-1})$:

$$G = \begin{pmatrix} \Omega(-\delta_\theta) & \Delta t \Omega(-\delta_\theta) \\ 0 & M \end{pmatrix} \tag{6}$$

$$M = \begin{cases} \begin{pmatrix} \frac{\partial g_{v_x}}{\partial v_x} & \frac{\partial g_{v_x}}{\partial v_y} \\ \frac{\partial g_{v_y}}{\partial v_x} & \frac{\partial g_{v_y}}{\partial v_y} \end{pmatrix} & \text{if } |v_{t-1}| \geq |k\Delta t| \\ \Omega(-\delta_\theta) & \text{else} \end{cases} \tag{7}$$

with

$$\frac{\partial g_{v_x}}{\partial v_x} = \left(1 + \frac{k\ \Delta t}{|v|}\right) \cos(-\delta_\theta) - \frac{k\ \Delta t\ v_x\,(\cos(-\delta_\theta) v_x - \sin(-\delta_\theta) v_y)}{|v|^3} \tag{8}$$

$$\frac{\partial g_{v_x}}{\partial v_y} = -\left(1 + \frac{k\ \Delta t}{|v|}\right) \sin(-\delta_\theta) - \frac{k\ \Delta t\ v_y\,(\cos(-\delta_\theta) v_x - \sin(-\delta_\theta) v_y)}{|v|^3} \tag{9}$$

$$\frac{\partial g_{v_y}}{\partial v_x} = \left(1 + \frac{k\ \Delta t}{|v|}\right) \sin(-\delta_\theta) - \frac{k\ \Delta t\ v_x\,(\sin(-\delta_\theta) v_x + \cos(-\delta_\theta) v_y)}{|v|^3} \tag{10}$$

$$\frac{\partial g_{v_y}}{\partial v_y} = \left(1 + \frac{k\ \Delta t}{|v|}\right) \cos(-\delta_\theta) - \frac{k\ \Delta t\ v_y\,(\sin(-\delta_\theta) v_x + \cos(-\delta_\theta) v_y)}{|v|^3}\,. \tag{11}$$

Thus local models of dynamic objects in the robot's environment can be modeled using separate Kalman filters. In case of the unpredictability of the motion of autonomous robots it is possible to neglect the estimation of their velocity and apply high process noise instead.

The separate localization module described in section 3.1, in itself also a buffer integrating information from static, known world features into a localization belief model, is used analogically to those percept-buffers, but the state is not deleted periodically after forwarding the belief to the SLAM part of the algorithm. This localization reflects part of the SLAM state, and changes to this part of the SLAM state are fed back into the localization module's state. Thus the virtual localization measurements used to update the SLAM state are basically the innovation introduced by new static feature observations. Therefore those measurements are still conditionally independent from previous measurements given the current belief state, so the Markov assumption is not violated.

3.3 Local and Distributed Knowledge Integration

The state of the full model of the robot's environment consists of its own pose $p_0 = (p_{0,x}, p_{0,y}, p_{0,\theta})^T$, the poses of all cooperating robots $(p_i = (p_{i,x}, p_{i,y}, p_{i,\theta})^T$

with $i \in 1, ..., n$), and the states of the dynamic objects. While only a small subset of cooperating robots or other elements are observed at the same time and modeled according to section 3.2 in each time interval, they remain part of the full model also during time intervals where these are not observed. It is possible to dynamically shrink or expand the state vector if new unknown robots are observed. Alternatively a separate mechanism could keep track of active and inactive slots in the state vector by using time-to-live counters. This latter approach has been chosen here to prevent frequent rescaling of both the state vector and its covariance matrix.

The integration of the locally accumulated and the distributed information into the model will be done in the process and sensor update. The own pose and those of cooperating robots can be updated with the pose changes propagated from the individual localization modules relative to the pose used for the last update. The ball is updated using a motion model similar to the one in equation 5, but without the odometry related rotations due to the local coordinate system. Other autonomous agents can either be updated according to the latest velocity estimations, or just using an identity and appropriately high process noise following the reasoning proposed in section 2.2.

The sensor update consists of two different kinds of observations. If a robot, either the local robot itself or any of the communicating robots in the team, has made observations of static world elements which have been used to update the separate localization estimate in the first stage (cp. section 3.1), then this absolute pose estimate is used as a direct measurement of the corresponding pose in the state vector, i.e. the measurement Jacobian is an identity in the corresponding submatrix.

The other case is the observation of a dynamic feature by one of the robots in the team. If the observed dynamic feature is a robot (without further identified characteristics such as team markers etc.), this dynamic object may either be any of other robots in the team, or one of a number of non-cooperating other robots in the environment. In this case, the maximum likelihood correspondence will be chosen to be updated, or a new model will be inserted or activated if the other choices are too unlikely. The corresponding expected observation is in a robot-relative euclidean coordinate system, since this is the format of the local models distributed as aggregated percepts. It is expressed as a function of the observed object's model $(m_x, m_y, m_{v_x}, m_{v_y})$ and its observer's pose p_i, with $i = 0$ for local observations and $i \in 1, ..., n$ communicated ones, which are otherwise not distinguished any further.

The observation model is given by equations 12 and 13

$$h_{m_x,m_y}(p_i) = \Omega(-p_{i,\theta}) \left[(m_x, m_y)^T - (p_{i,x}, p_{i,y})^T \right] \tag{12}$$

$$h_{m_{v_x},m_{v_y}}(p_i) = \Omega(-p_{i,\theta}) \left(m_{v_x}, m_{v_y} \right)^T \tag{13}$$

from which the corresponding entries in the measurement Jacobian can be calculated as in equation 14, with c_θ and s_θ short for $\cos p_{i,\theta}$ and $\sin p_{i,\theta}$, respectively.

$$\begin{pmatrix} \frac{\partial h_{m_x}}{\partial m_x} & \frac{\partial h_{m_x}}{\partial m_y} & \frac{\partial h_{m_x}}{\partial m_{v_x}} & \frac{\partial h_{m_x}}{\partial m_{v_y}} & \frac{\partial h_{m_x}}{\partial p_{i,x}} & \frac{\partial h_{m_x}}{\partial p_{i,y}} & \frac{\partial h_{m_x}}{\partial p_{i,\theta}} \\ \frac{\partial h_{m_y}}{\partial m_x} & \frac{\partial h_{m_y}}{\partial m_y} & \frac{\partial h_{m_y}}{\partial m_{v_x}} & \frac{\partial h_{m_y}}{\partial m_{v_y}} & \frac{\partial h_{m_y}}{\partial p_{i,x}} & \frac{\partial h_{m_y}}{\partial p_{i,y}} & \frac{\partial h_{m_y}}{\partial p_{i,\theta}} \\ \frac{\partial h_{m_{v_x}}}{\partial m_x} & \frac{\partial h_{m_{v_x}}}{\partial m_y} & \frac{\partial h_{m_{v_x}}}{\partial m_{v_x}} & \frac{\partial h_{m_{v_x}}}{\partial m_{v_y}} & \frac{\partial h_{m_{v_x}}}{\partial p_{i,x}} & \frac{\partial h_{m_{v_x}}}{\partial p_{i,y}} & \frac{\partial h_{m_{v_x}}}{\partial p_{i,\theta}} \\ \frac{\partial h_{m_{v_y}}}{\partial m_x} & \frac{\partial h_{m_{v_y}}}{\partial m_y} & \frac{\partial h_{m_{v_y}}}{\partial m_{v_x}} & \frac{\partial h_{m_{v_y}}}{\partial m_{v_y}} & \frac{\partial h_{m_{v_y}}}{\partial p_{i,x}} & \frac{\partial h_{m_{v_y}}}{\partial p_{i,y}} & \frac{\partial h_{m_{v_y}}}{\partial p_{i,\theta}} \end{pmatrix}$$
$$= \begin{pmatrix} c_\theta & s_\theta & 0 & 0 & -c_\theta & -s_\theta & -(m_x - p_{i,x}) \cdot s_\theta + (m_y - p_{i,y}) \cdot c_\theta \\ -s_\theta & c_\theta & 0 & 0 & s_\theta & -c_\theta & -(m_x - p_{i,x}) \cdot c_\theta - (m_y - p_{i,y}) \cdot s_\theta \\ 0 & 0 & c_\theta & s_\theta & 0 & 0 & -m_{v_x} \cdot s_\theta + m_{v_y} \cdot c_\theta \\ 0 & 0 & -s_\theta & c_\theta & 0 & 0 & -m_{v_x} \cdot c_\theta - m_{v_y} \cdot s_\theta \end{pmatrix} \tag{14}$$

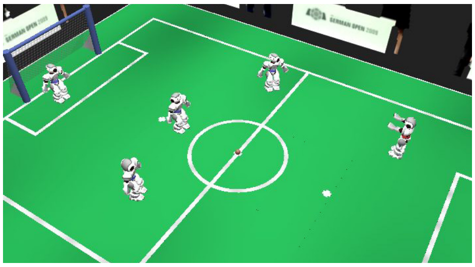
Re-localization events can be handled by resetting the corresponding state variables and removing the covariances, i.e. setting all entries in the covariance matrix in the rows and columns to zero. If such a previous mis-localization by a team member resulted in modeled false positives, those will stay as isolated features in the state for some time and will be deleted or inactivated after a certain time without observation. This serves as a self-repair routine to remove clutter from the environmental model, and to prevent the growth of the state by the accumulation of models of such elements. The same is done if two models of unknown features are decided to correspond to the same origin after a series of observations, so that the information needs to be fused into the first model and the seconds needs to be deactivated. Alternatively it would be possible to keep multiple environment models for each localization hypothesis, as done in [4].

4 Evaluation

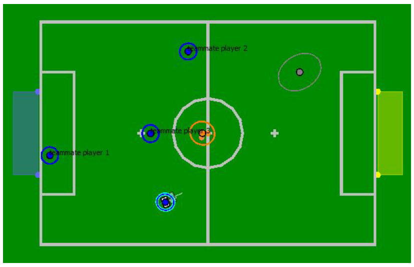
The modeling process is complex and incorporates a multitude of different information, so that a step by step illustration of the working principle is not practical. To evaluate the presented approach, a simulated situation first illustrates the theoretical possibilities and the qualitative effect in section 4.1, followed by a quantitative analysis in soccer games using experiments with real robots in section 4.2. Both setups use an SPL scenario as specified by the 2011 rules.

4.1 Qualitative Demonstration

Figure 2 illustrates a simple scenario in a simulated environment. The robots in a team share their information for distributed cooperative modeling. Figure 2(b) shows the resulting model with 2D covariance ellipses extracted from the full state. In the following, one robot looks down and does not see any static field features any more, and both he and the ball are teleported to another location on the field (see figure 3). The use of distributed percepts and the modeling of the own pose together with the ones of other robots and the ball position and velocity allows the robot to not only correct its position, but also its orientation.

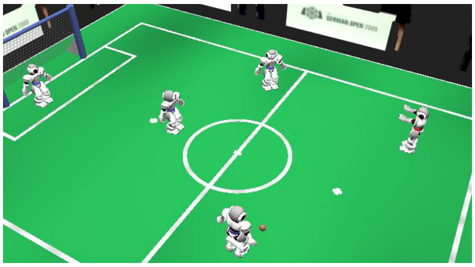


(a) Setup of the robots on the field.

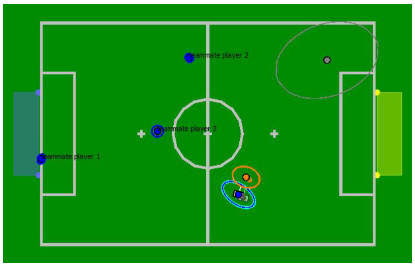


(b) World model generated from local and distributed information.

Fig. 2. Scenario with a team of robots looking around and sharing perception information to cooperatively model their environment



(a) Scenario after teleportation of ball and downwards-looking robot.



(b) World model generated from local and distributed information.

Fig. 3. Following the situation in figure 2, one robot looks down and only sees the ball but no landmarks, and he and the ball are teleported. The shared information however still allows for a correction of both position and orientation of the robot.

This simple experiment shows the potential usefulness of such a combined modeling of a robot's dynamic environment and its pose in it. RoboCup SPL games contain periods where robots are chasing the ball, approaching it for precise positioning to shoot at the goal, or even dribbling it. During those periods odometry errors are integrated into the robot's localization if not countered by frequently looking up at static field features to correct the robot's pose estimation. If looking at the ball also allows the correction of those odometry errors, especially the orientation, this is expected to be a clear advantage.

4.2 Quantitative Performance Evaluation

The artificial situation created in the previous section just serves as an example of how localization benefits may be gained. To allow a quantitative evaluation of the approach's performance, the perceptions of a robot have been recorded during normal game situations with real robots on a regular SPL field. Those perceptions include the proprioception, i.e. odometry, orientation and joint angle information, exteroception, i.e. perceptions of objects by means of image processing, as well as the distributed local models of other cooperating robots running the same code, and ground truth information provided by a camera system mounted above the field.

This set of input information is then processed by two different module configurations. One is the configuration described in section 3. The second uses the same localization but a simpler module for cooperative tracking of dynamic objects without any feedback into the localization, and has been used to win the second place at RoboCup 2011. This experiment is not set up to show that the localization works, since both solutions are based on the same competitive solution for the localization problem with all features described in [10], but to evaluate the additional benefit gained by unified modeling of the full state.

A first evaluation of several recorded situations did not show any conclusive results, meaning the positive and negative effects of the full state modeling equaled out most of the time, in a low percentage of cases the full system even showed a slightly decreased localization quality. Closer evaluation showed that the currently used visual robot recognition provides too much uncertainty or even uncorrected systematical errors, such as in the distance estimation, to be beneficial for the localization.

A second configuration of the system, which ignores the robot perceptions for the modeling of the robot's own position, but still uses the much more precise ball perceptions, showed the expected results. As can be seen in the representative extract visualized in figure 4, the proposed system provides beneficial information for the robot's own localization most of the time. A direct comparison of the localization quality of both systems shows that the robot pose translation errors for the full system model are below 25 cm in 83% of the time, and only 72% of the time for the unassisted underlying localization module, and the average errors over the whole experiment are 166 mm compared to 213 mm. However, note that with this second configuration of the system, in the teleportation experiment the robot's orientation could not be recovered as easily as described in section 4.1.

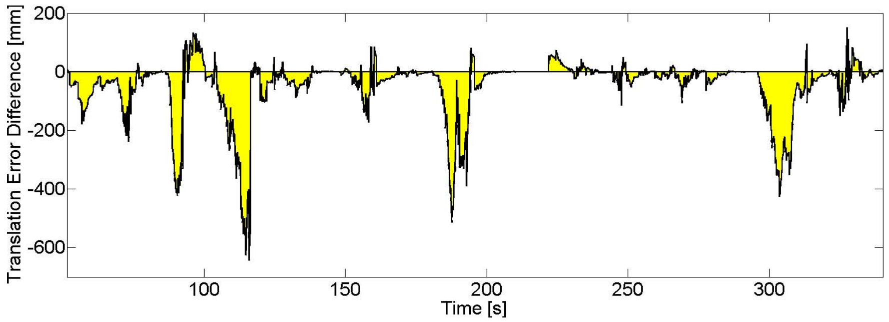


Fig. 4. Difference of translation errors of the two described systems. Negative values mean larger errors of the unassisted localization compared to modeling the full state.

5 Conclusion

This paper presents the advantages of modeling the full environment state estimation as compared to only localizing in said environment. A competitive stand-alone localization module is extended to perform as a full state model, and the additional gain in localization performance is evaluated both in a simulated situation as well as in several real world experiments with multiple robots and ground truth provided by an external camera system. While the robot perception in the current vision system is not good enough to benefit from using temporary opponent models as additional features for localization, usage of the ball as a dynamic feature significantly improves the localization quality.

An additional advantage of estimating the full state in a cooperative modeling approach is the existence of a single model which contains all information in a globally consistent way. This renders the switching between local tracking of the ball and a global team ball model obsolete, for example, and therefore simplifies behavior specification.

References

1. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). MIT Press (2005)
2. Hähnel, D., Schulz, D., Burgard, W.: Map building with mobile robots in populated environments. In: Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2002)
3. Schulz, D., Burgard, W., Fox, D., Cremers, A.B.: Tracking multiple moving objects with a mobile robot. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, p. 371 (2001)
4. Czarnetzki, S., Rohde, C.: Handling heterogeneous information sources for multi-robot sensor fusion. In: Proceedings of the 2010 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2010), Salt Lake City, Utah, pp. 133–138 (September 2010)
5. Kümmerle, R., Steder, B., Dornhege, C., Kleiner, A., Grisetti, G., Burgard, W.: Large scale graph-based SLAM using aerial images as prior information. In: Proceedings of Robotics: Science and Systems (RSS), Seattle, WA, USA (June 2009)
6. Stroupe, A., Matrin, M., Balch, T.: Distributed sensor fusion for object position estimation by multi-robot systems. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001) (2001)
7. Dietl, M., Gutmann, J.S., Nebel, B.: Cooperative Sensing in Dynamic Environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1706–1713 (2001)
8. Howard, A.: Multi-robot simultaneous localization and mapping using particle filters. The International Journal of Robotics Research 25(12), 1243–1256 (2006)
9. Zhou, X.S., Roumeliotis, S.I.: Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS), Beijing, China, pp. 1785–1792 (2006)
10. Jochmann, G., Kerner, S., Tasse, S., Urbann, O.: Efficient multi-hypotheses unscented kalman filtering for robust localization. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 222–233. Springer, Heidelberg (2012)

On Sensor Model Design Choices for Humanoid Robot Localization

Stefan Tasse, Matthias Hofmann, and Oliver Urbann

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

Abstract. The development of estimation systems based on Kalman filters requires several design choices. Among others, these are the methods used for linearization, coordinate systems for measurement representations, and approximations such as how to handle multiple simultaneous observations per time step. This paper evaluates these different choices with respect to their influence on the system's estimation quality and points out simple yet effective solutions. Camera-based localization for a humanoid robot is chosen as an example application and the localization benefit of different approaches is evaluated using real and simulated feature perceptions.

1 Introduction

Localization is essential for mobile robots. When facing the task of designing a localization algorithm for an autonomous robot, one may pick from a vast number of different approaches. While there exist many different strategies such as multi-angulation methods [1] or constraint based localization [2], most algorithms follow the concepts of recursive Bayesian filtering [3]. The two main representatives of this category are Kalman and particle filters. Gutmann and Fox state the common impression that "Markov localization is more robust than Kalman filtering while the latter can be more accurate than the former" in [4]. An additional argument for particle filtering is the easy representation of multimodal belief states. However, Gaussian mixtures allow the same for Kalman filters, and recently such multiple model Kalman filters have been applied with great success even on robot platforms with very limited resources [5,6], allowing superior localization quality and robustness to comparable particle filters, but at lower computational costs in case of [6].

Kalman filters have been applied in many tasks and are covered extensively in literature. Designing a Kalman filter for localization is therefore not a significant challenge. However, this filter will often not perform to its full potential. The implementation process presents several design choices, some of which are discussed frequently, while others are generally neglected or only mentioned briefly. Specialized references such as [7] as well as the most standard books [3,8] leave the impression that the most important decision is whether to address the system's non-linearity by Taylor series expansion such as in the Extended Kalman filter (EKF) or by use of the Unscented transform as in the Unscented Kalman filter (UKF). This will be addressed only briefly in Section 2. Other influences will be discussed in the course of this paper in Section 3 and 4. Those may seem trivial at first and their importance not obvious, but their significant influence on the outcome will be shown for the example of localization for a humanoid robot. As such, it is this paper's main contribution to point out simple design choices which will lead to significant localization quality improvement with minimal effort.

2 Addressing Non-linearity by Taylor Series Expansion or Unscented Transform

The Kalman filter in its original form is optimal for systems which fulfill a number of assumptions, such as only involvement of zero mean Gaussian noise and a known and linear system to model. This is rarely given for practical applications, since most systems of interest are non-linear in one aspect or another. The Kalman concept is popular and successful nonetheless, which is due to the possibility to linearize the non-linear models around the current estimate. This provides a decent enough approximation to allow the tracking.

In general, two different concepts are commonly used: the Extended and the Unscented Kalman filter. The Extended Kalman filter employs a Taylor series for linearization, which in effects means to simply substitute Jacobi matrices for the linear transformations in the original Kalman filter equations. This method is and has been widely used for the last four decades. See [3] for further details. Of course the linearization may result in different approximation qualities of the uncertainty propagation, depending on each individual use case. Further limitations of this approach arise in cases of discontinuous systems or such with singularities. Additionally, it is often perceived by developers that "calculating Jacobian matrices can be a very difficult and error-prone process" [7] due to the manual derivation of the Jacobians and possible translation errors in their subsequent conversion to code.

The Unscented Kalman filter offers a different approach to estimate the different expectations necessary to apply the standard Kalman equations, namely the predictions of state and observation and the cross-covariance between the two. This is done by deterministically sampling the state space around the current mean and covariance, applying the non-linear transformation to those sigma points, and then recovering the transformed mean and covariance. Significant improvements of applying the unscented transformation compared to analytical linearization have been shown in [7].

Those findings have led to the impression that choosing an Unscented Kalman filter instead of an Extended Kalman filter will be a major source of improvement in most systems, and that this will be the main design choice in developing an estimation system for a given application. In the course of this paper we will show that much easier alterations may have much bigger effects.

3 Measurement Coordinate System Choices

To illustrate the effect of measurement coordinate system choices, we assume as an example application the problem of estimating the localization of a humanoid robot, so the state to be estimated is the robot's pose $x = (p_x, p_y, p_\theta)^T$. The robot perceives point features on the ground around it, e.g. by means of processing images recorded by one or several cameras mounted in its head. Each point feature corresponds to a landmark with known global position $l = (l_x l_y)^T$. Those expected and actual perceptions, $\overline{z}$ and z respectively, can by expressed in different coordinate systems, each of which may be used to formulate the sensor model of the Kalman filter.

In the following, let

$$\Omega(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \tag{1}$$

be the rotation around α and (l_x, l_y) the global coordinates of a known landmark which is part of the robot's map of the environment. The time index t is omitted in all following equations for the sake of simplicity.

3.1 Measurements in Cartesian Coordinates

As the localization problem is expressed as an orientation and a position in global Cartesian coordinates, a first intuitive choice is to express a measurement on the ground around the robot in robot-centric Cartesian coordinates as shown in Figure 1. The sensor model to calculate the expected measurement $\overline{z} = (z_x, z_y)^T$ for the current robot pose $x = (p_x, p_y, p_\theta)^T$ and a correspondence to the landmark with known global position $l = (l_x l_y)^T$ is then given in Equation 2 and the corresponding Jacobi matrix in Equation 3.

$$\overline{z} = \begin{pmatrix} z_x \\ z_y \end{pmatrix} = h(x, l) = \Omega(-p_\theta) \cdot \left[\begin{pmatrix} l_x \\ l_y \end{pmatrix} - \begin{pmatrix} p_x \\ p_y \end{pmatrix} \right] \tag{2}$$

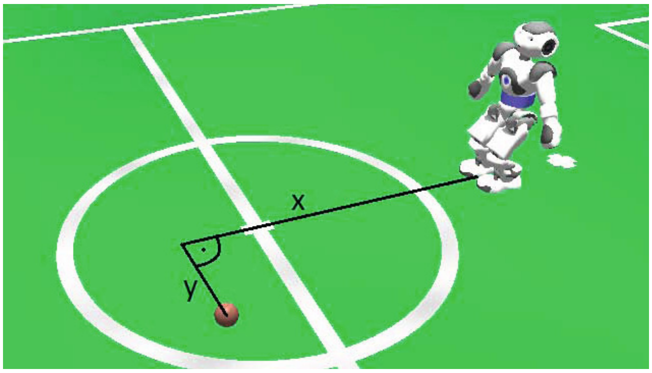


Fig. 1. Observation given in euclidean coordinates

$$H = \frac{\partial h(x,l)}{\partial x} = \begin{pmatrix} \frac{\partial z_x}{\partial p_x} & \frac{\partial z_x}{\partial p_y} & \frac{\partial z_x}{\partial p_\theta} \\ \frac{\partial z_y}{\partial p_x} & \frac{\partial z_y}{\partial p_y} & \frac{\partial z_y}{\partial p_\theta} \end{pmatrix} \tag{3}$$
$$= \begin{pmatrix} -\cos p_\theta & -\sin p_\theta & -(l_x - p_x)\sin p_\theta + (l_y - p_y)\cos p_\theta \\ sinp_\theta & -\cos p_\theta & -(l_x - p_x)\cos p_\theta - (l_y - p_y)\sin p_\theta \end{pmatrix}$$

3.2 Measurements in Cylindrical Coordinates

Measurements can also be expressed in cylindrical coordinates, i.e. range and bearing, to indicate the distance and direction of the observed feature (cf. Figure 2).

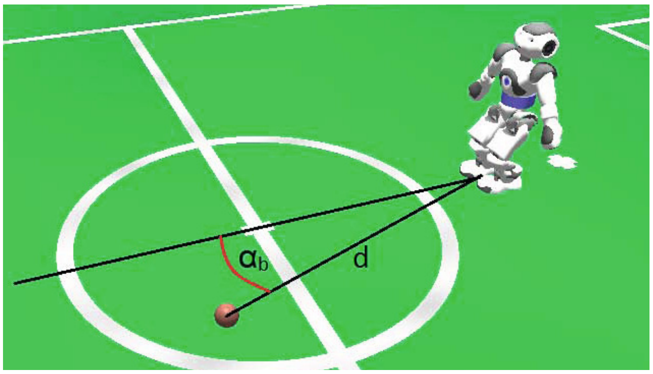


Fig. 2. Observation given as range and bearing

This is often the first choice of those familiar with laser scanners or developers of robot-centric path planning algorithms. In this case, the Sensor model function and Jacobi matrix are given by Equation 4 and 5, respectively, with the abbreviation $d^2 = (l_x - p_x)^2 + (l_y - p_y)^2$.

$$\overline{z} = \begin{pmatrix} z_r \\ z_b \end{pmatrix} = h(x,l) = \begin{pmatrix} \sqrt{(l_x - p_x)^2 + (l_y - p_y)^2} \\ \text{atan2}(l_y - p_y, l_x - p_x) - p_\theta \end{pmatrix} \tag{4}$$

$$H = \frac{\partial h(x,l)}{\partial x} = \begin{pmatrix} \frac{\partial z_r}{\partial p_x} & \frac{\partial z_r}{\partial p_y} & \frac{\partial z_r}{\partial p_\theta} \\ \frac{\partial z_b}{\partial p_x} & \frac{\partial z_b}{\partial p_y} & \frac{\partial z_b}{\partial p_\theta} \end{pmatrix} \tag{5}$$
$$= \begin{pmatrix} (-l_x + p_x)d^{-1} & (-l_y + p_y)d^{-1} & 0 \\ (l_y - p_y)d^{-2} & (-l_x + p_x)d^{-2} & -1 \end{pmatrix}$$

3.3 Measurements in Spherical Coordinates

A third coordinate system choice is given by using the vertical and horizontal angles α_1 and α_2 as indicated in Figure 3. While the meaning of the vertical

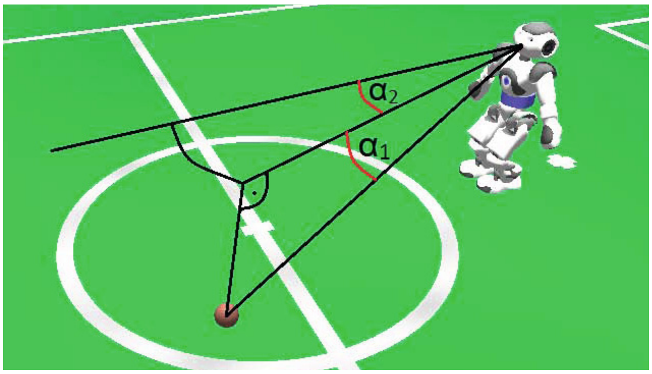


Fig. 3. Observation given in angular coordinates

angle may not be intuitive for any direct further use, this is the coordinate system which is closest to the actual perception process in this example. With the same abbreviation of $d^2 = (l_x - p_x)^2 + (l_y - p_y)^2$ as used above and the height of the camera h_{camera}, the sensor model function and Jacobi matrix are given in Equation 6 and 7.

$$\begin{aligned}\overline{z} &= \begin{pmatrix} z_{\alpha_1} \\ z_{\alpha_2} \end{pmatrix} = h(x, l, h_{camera}) \\ &= \begin{pmatrix} \operatorname{atan2}(h_{camera}, \sqrt{(l_x - p_x)^2 + (l_y - p_y)^2}) \\ \operatorname{atan2}(l_y - p_y, l_x - p_x) - p_\theta \end{pmatrix}\end{aligned} \tag{6}$$

$$\begin{aligned}H = \frac{\partial h(x,l)}{\partial x} &= \begin{pmatrix} \frac{\partial z_{\alpha_1}}{\partial p_x} & \frac{\partial z_{\alpha_1}}{\partial p_y} & \frac{\partial z_{\alpha_1}}{\partial p_\theta} \\ \frac{\partial z_{\alpha_2}}{\partial p_x} & \frac{\partial z_{\alpha_2}}{\partial p_y} & \frac{\partial z_{\alpha_2}}{\partial p_\theta} \end{pmatrix} \\ &= \begin{pmatrix} \frac{h_{camera}(l_x - p_x)}{d(h_{camera}^2 + d^2)} & \frac{h_{camera}(l_y - p_y)}{d(h_{camera}^2 + d^2)} & 0 \\ (l_y - p_y)d^{-2} & (-l_x + p_x)d^{-2} & -1 \end{pmatrix}\end{aligned} \tag{7}$$

3.4 Experimental Comparison

Two experiments are set up to compare the effects of the sensor model design choices described so far.

Simulated Perception. A simulation is set up to test the correctness of the implementation and the conformity with related work's results. Localization algorithms are used with the above mentioned different linearization and coordinate system choices and parametrized using fixed measurement covariances, which were chosen to be optimal for each approach separately. This simulation assumes a humanoid robot with noisy odometry and a perception process which measures randomly distributed landmarks with unique correspondences and contains errors mainly from the cameras unknown orientation, i.e. the errors originate from normally distributed noise in the spherical coordinate system. As expected, Figure 4 shows the localization to be best using this spherical representation. Furthermore, the classical example of transforming between spherical/cylindrical and Cartesian coordinates is handled much better by the UKF than by the EKF as predicted for example by [7].

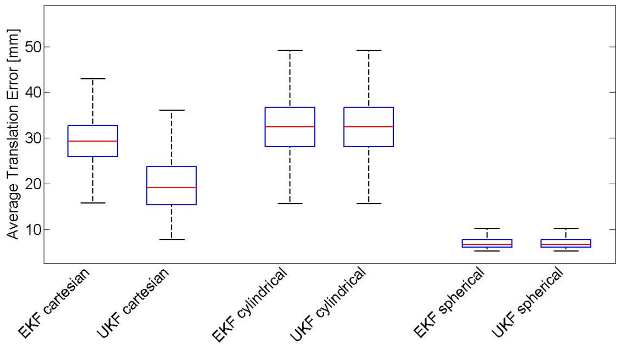


Fig. 4. Comparison between localization quality using different linearization approaches and sensor model coordinate systems with simulated perceptions

Real Perception Process on the Nao. To evaluate the impact on a real system, observations are recorded using a Nao, a humanoid robot which is 58 cm tall and equipped with two cameras in its head with non-overlapping fields of view. The environment is a robot soccer field as used in the Standard Platform League. Any ambiguous observations are associated with maximum likelihood correspondences based on the true robot position. The perception process also produces sporadic false positives. Those sets of observations and correspondences together with artificially generated odometry errors serve as input for all different configurations, whereas each generated set is processed by all approaches so that the random component in the input presents no source of bias. Note that these localization results will not diverge due to the usual problem of wrong correspondence choices once the position estimate contains a certain error, as this experiment is set up to test the sensor models, not the correctness of correspondence choices. The odometry errors contain white noise and a drift component, as this is the usual behavior of real Nao robots which are worn out or even heated up slightly asymmetrically.

An important factor for each algorithm is its parametrization. All different approaches in this experiment use the same motion update and the same process noise, which is chosen to be a certain amount above the artificially generated white noise component to compensate the drift. In common applications the measurement noise magnitudes are normally constants which are part of the parametrization and subject to a tuning process by the developer. Here, they are optimized separately for the approaches using a randomly picked measurement subset which is not used for the following evaluation afterwards. Thus each approach is performing with the parametrization which empirically provides the least squared localization errors.

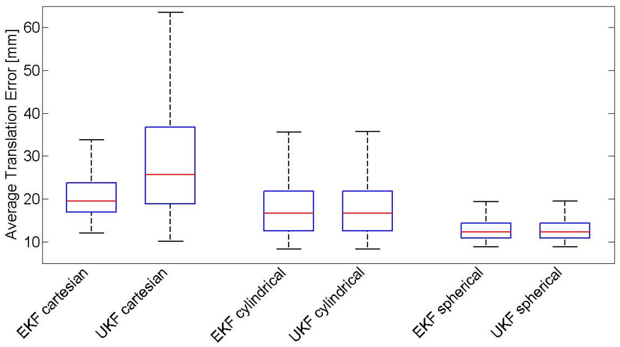


Fig. 5. Comparison between localization quality using different linearization approaches and sensor model coordinate systems with real observations recorded on a Nao

Figure 5 shows the distribution of the sums of localization errors for 1000 sets of measurements and odometry errors. It can be seen that the effect of the coordinate system choice is in general more significant than the distinction between Extended or Unscented Kalman filter. These real world results mostly verify the tendency of the assumptions in the simulated experiments, but also show discrepancies for example in the results using Cartesian coordinates. This implies that the underlying process is not fully described by assuming only normally distributed angular errors in the camera orientation. Expressing the measurement in spherical coordinates, which is intuitively the closest to the underlying process of perception, still clearly outperforms the other coordinate systems' sensor models. To use these results as a basis for development recommendations, the EKF/UKF choice is clearly second to the angular coordinate representation of the robot's measurements.

3.5 Hybrid Modifications

The empirical results above raise the question if already implemented systems which did not use the spherical coordinate system for the sensor model design can still make use of this information. One possibility is to adapt the measurement noise covariance matrix to better reflect the properties of the perception process, e.g. to scale the uncertainty depending on the distance of the observed feature. This has not led to any significant improvements in case of the Cartesian representation, for which more complex modification would be necessary to adapt it to reflect the spherical coordinate system's properties. The cylindrical representation however offers an easy improvement.

Both the cylindrical and the spherical coordinate system already share the horizontal angle; they only differ in distance against vertical angle. Applying the knowledge that errors in the distance mainly result from variations in said vertical angle, it is possible to derive an appropriate scaling factor β for the distance measurement's uncertainty.

$$z_r = \frac{h_{camera}}{\sin z_{\alpha_1}} \tag{8}$$

$$\frac{\partial z_r}{\partial z_{\alpha_1}} = -\frac{h_{camera}}{\sin^2 z_{\alpha_1}} \cdot \cos z_{\alpha_1} \tag{9}$$

$$\beta \propto \frac{h_{camera}}{\sin^2 \operatorname{atan2}(h_{camera}, z_r)} \cdot \cos \operatorname{atan2}(h_{camera}, z_r) \tag{10}$$

Equation 8 gives the relation between range observation z_r and vertical angle z_{α_1} and Equation 9 denotes their partial derivative. Therefore, using the first rows of Equation 6 and 4, the scaling factor β in Equation 10 can be derived. Scaling the (newly tuned) expected range error with β or the corresponding entry in the measurement covariance matrix with β^2 results in the hybrid localization approach with cylindrical coordinates and distance scaled measurement covariance in Figure 6 and 7. While this one presents a significant improvement over the cylindrical coordinates with constant measurement covariance and comes close to the approach in spherical coordinates, the latter one is still superior.

4 Multiple Simultaneous Measurements

Practical Kalman implementations rarely go by the theory of one motion update and one sensor update per time step. Instead there are usually many time steps in which no observation is made, so the sensor update is omitted. In other time steps, several observations are made at once, i.e. several different features are detected in the same time step. The common implementation is usually to execute several consecutive sensor updates. The quality of this approximation however depends on the perception process by which those features have been observed.

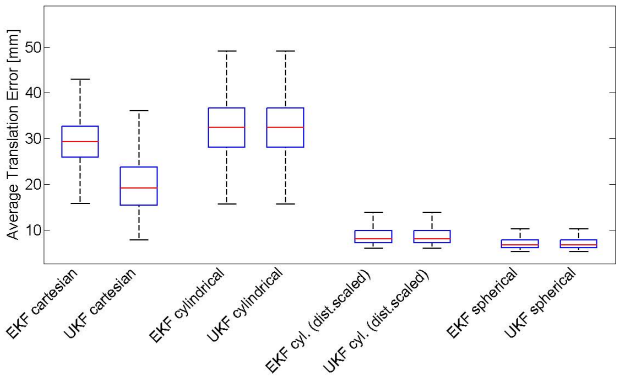


Fig. 6. Comparison between localization quality using different linearization approaches and sensor model coordinate systems with simulated perceptions

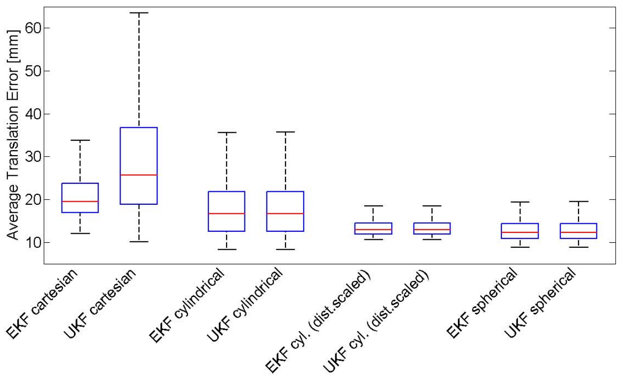


Fig. 7. Comparison between localization quality using different linearization approaches and sensor model coordinate systems with real observations recorded on a Nao

Now consider 2-dimensional feature observations as described above, each with a separate measurement covariance as in Equation 11.

$$C = \begin{pmatrix} s_1^2 & 0 \\ 0 & s_2^2 \end{pmatrix} \tag{11}$$

The stochastically correct sensor update for n detected features would be to execute a single $2n$-dimensional measurement update instead of n separate 2-dimensional updates. In case the different measurements are stochastically independent of each other, i.e. all off-diagonal inter-feature entries of the $2n \times 2n$ measurement covariance are zero, then a single $2n$-dimensional measurement update is approximated well by n 2-dimensional updates.

If multiple measurements originate from the same perception source and are correlated, then this simple approximation neglects potentially useful information and consequently looses in approximation quality. Taking a humanoid robot with camera based perception again as in Section 3.4, multiple observations originate from the processing of a single camera image and it stands to reason that the main source of measurement error is the inaccurately estimated camera orientation due to the walking motion. Such simultaneous measurements would therefore contain nearly the same angular errors. Assuming a spherical coordinate representation for the measurements as described in Section 3.3, the resulting covariance for 2 simultaneous observations is given in Equation 12 with γ close to 1, while $\gamma = 0$ would neglect any dependence between both observations.

$$C' = \begin{pmatrix} C & \gamma C \\ \gamma C & C \end{pmatrix} \tag{12}$$

Figure 8 shows evaluations with simulated test runs consisting exclusively of multiple observations per time step, and illustrates the differences in localization quality for iterative execution of 2-dimensional sensor updates, for $2n$-dimensional updates which neglect the covariance (i.e. with $\gamma = 0$), and for $2n$-dimensional updates with full covariances as in Equation 12. All sensor updates in this example utilize spherical coordinate representations for the observations. As expected, the multiple 2-dimensional updates are an appropriate approximation as long as the separate measurements are independent. When observations are correlated, then significant benefits can be drawn from the information encoded in the full covariance matrix. Note that some Unscented Kalman filter implementations may become unstable for a γ too close to 1, as C' will still be a

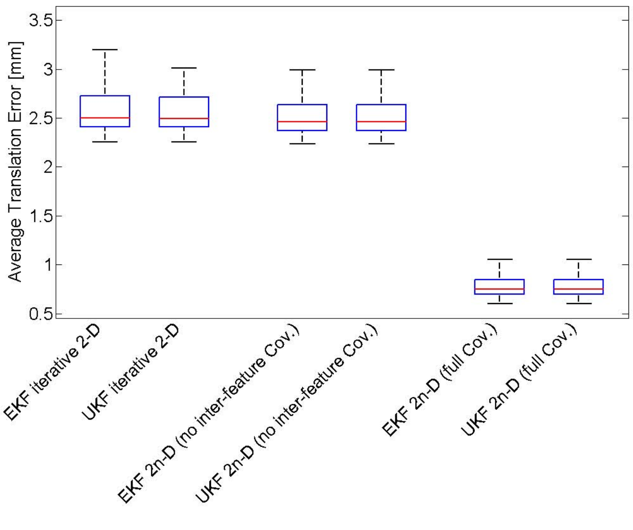


Fig. 8. Comparison between different methods to handle multiple measurements at one time step

valid covariance matrix and therefore positive semi-definite, but very close to not being positive definite any more, which will cause the frequently used Cholesky decomposition to become numerically instable.

5 Conclusion

This paper gives an overview about common design choices which researchers face when developing localization algorithms based on Kalman filters. The most prominent choice between the Extended and Unscented Kalman filter is widely discussed in common literature, but this is by far overrated, which has been illustrated in the previous sections using the example application of camera-based humanoid robot localization. The choice of the measurement's coordinate system representation is mostly disregarded in most publications as well as in common educational books, but provides a simple way to improve localization quality. The same holds for the correct handling of simultaneous observations originating from processing the same camera image.

As such, this paper provides the means for a better understanding of different approximations' impacts when applying Kalman filters, and presents simple guidelines to develop optimal solutions for localization and tracking systems.

References

1. Betke, M., Gurvits, L.: Mobile robot localization using landmarks. IEEE Transactions on Robotics and Automation 13(2), 251–263 (1997)
2. Göhring, D., Mellmann, H., Burkhard, H.D.: Constraint based world modeling in mobile robotics. In: Proc. IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 2538–2543 (2009)
3. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press (2005)
4. Gutmann, J.S., Fox, D.: An Experimental Comparison of Localization Methods continued. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 454–459 (2002)
5. Quinlan, M.J., Middleton, R.H.: Multiple Model Kalman Filters: A Localization Technique for RoboCup Soccer. In: Baltes, J., Lagoudakis, M.G., Naruse, T., Ghidary, S.S. (eds.) RoboCup 2009. LNCS, vol. 5949, pp. 276–287. Springer, Heidelberg (2010)
6. Jochmann, G., Kerner, S., Tasse, S., Urbann, O.: Efficient multi-hypotheses unscented kalman filtering for robust localization. In: Röfer, T., Mayer, N.M., Savage, J., Saranlı, U. (eds.) RoboCup 2011. LNCS, vol. 7416, pp. 222–233. Springer, Heidelberg (2012)
7. Julier, S., Uhlmann, J.: Unscented filtering and nonlinear estimation. Proceedings of the IEEE 92(3), 401–422 (2004)
8. Siegwart, R., Nourbakhsh, I.: Introduction to autonomous mobile robots. Intelligent Robotics and Autonomous Agents. The MIT Press (2004)

Keyword Index

Author Index