Proceedings of the
9th Conference on Autonomous Robot Systems and Competitions

Castelo Branco, Portugal

7th May, 2009

Organized by

Polytechnic Institute of Castelo Branco

Superior School of Technology

Technical Co-Sponsorship

IEEE Robotics and Automation Society

Sponsored by

IEEE Robotics and Automation Society

IEEE Portugal Section

FCT – Portuguese Science Foundation

IEEE Education Society – Student Activities Committee

Portuguese Journal Robótica

Hosted by

Polytechnic Institute of Castelo Branco – Superior School of Technology

Edited by Paulo J.S. Gonçalves, Paulo J.D. Torres, Carlos M.O. Alves. Printed in Portugal

ISBN: 978-972-99143-7-9

http://www.est.ipcb.pt/robotica2009/ec

ORGANIZING AND STEERING COMMITTEES

Conference Chair
Paulo J.S. Gonçalves, Polytechnic Institute of Castelo Branco, Portugal
Conference Co-chair
Paulo J.D. Torres, Polytechnic Institute of Castelo Branco, Portugal
Program Chairs
Paulo J.S. Gonçalves, Polytechnic Institute of Castelo Branco, Portugal
Paulo J.D. Torres, Polytechnic Institute of Castelo Branco, Portugal

Proceedings Production
Paulo J.S. Gonçalves, Polytechnic Institute of Castelo Branco, Portugal
Secretariat
Paula Louro, Polytechnic Institute of Castelo Branco, Portugal
Paula Ribeiro, Polytechnic Institute of Castelo Branco, Portugal
Webdesigner
Ricardo Fontes, Polytechnic Institute of Castelo Branco, Portugal

International Steering Committee
Alessandro De Luca (URoma - IT)
Alicia Casals (UPC - ES)
Bruno Siciliano (UNINA - IT)
Hideki Hashimoto (UTokio - JP)
Jean-Paul Laumond (LAAS - FR)
Lynne Parker (UTK - USA)
P.I. Corke (CSIRO - AU)
Vijay Kumar (UPENN - USA)

Edited by P.J.S. Gonçalves, P.J.D. Torres, C.M.O. Alves - ISBN 978-972-99143-8-6

INTERNATIONAL PROGRAM COMMITTEE

Alfredo Martins (ISEP - PT)
Aníbal Matos (FEUP - PT)
A. Paulo. Moreira (FEUP - PT)
A. Scolari Conceição (UFOP - BR)
Bradley Nelson (ETHZ - CH)
Carlos Cardeira (IST - PT)
Carlos Carreto (IPG - PT)
Eduardo Silva (ISEP - PT)
Estela Bicho (UMinho - PT)
Ezio Malis (INRIA - FR)
Fernando Ribeiro (UMinho - PT)
Filipe Silva (UAveiro - PT)
François Chaumette (IRISA - FR)
Hélder Araujo (UCoimbra - PT)
João M.C. Sousa (IST - PT)
João R. Caldas Pinto (IST - PT)
Jorge Dias (UCoimbra - PT)
Jorge Ferreira (UAveiro - PT)
José Miguel Almeida (ISEP - PT)
José Carlos Alves (FEUP - PT)
Jorge Martins (IST - PT)
J.L. Azevedo (UAveiro -PT)
J.M. Sebastian (UPM - ES)
J. Tenreiro Machado (ISEP - PT)
Luís Almeida (UAveiro - PT)
Luís Paulo Reis (FEUP - PT)
Luís Seabra Lopes (UAveiro - PT)
Maria Isabel Ribeiro (IST - PT)
Nicolas Garcia-Aracil (UMH - ES)
Paolo Fiorini (UNIVR - IT)
Paulo Costa (FEUP - PT)
Paulo Oliveira (IST - PT)
Pedro Lima (IST - PT)
P. Martinet (LASMEA - FR)
Rui Cortesão (UCoimbra - PT)
Urbano Nunes (UCoimbra - PT)
Vicente Matellán (ULeon - ES)
Vítor Santos (UAveiro - PT)

Edited by P.J.S. Gonçalves, P.J.D. Torres, C.M.O. Alves - ISBN 978-972-99143-8-6

FOREWORD

Welcome to ROBOTICA 2009. This is the 9th edition of the conference on Autonomous Robot Systems and Competitions, the third time with IEEE-Robotics and Automation Society Technical Co-Sponsorship. Previous editions were held since 2001 in Guimarães, Aveiro, Porto, Lisboa, Coimbra and Algarve. ROBOTICA 2009 is held on the 7th May, 2009, in Castelo Branco , Portugal.

ROBOTICA has received 32 paper submissions, from 10 countries, in South America, Asia and Europe. To evaluate each submission, three reviews by paper were performed by the international program committee. 23 papers were published in the proceedings and presented at the conference. Of these, 14 papers were selected for oral presentation and 9 papers were selected for poster presentation. The global acceptance ratio was 72%.

After the conference, eighth papers will be published in the Portuguese journal Robótica, and the best student paper will be published in IEEE Multidisciplinary Engineering Education Magazine.

Three prizes will be awarded in the conference for: the best conference paper, the best student paper and the best presentation. The last two, sponsored by the IEEE Education Society - Student Activities Committee.

We would like to express our thanks to all participants. First of all to the authors, whose quality work is the essence of this conference. Next, to all the members of the international program committee and reviewers, who helped us with their expertise and valuable time. We would also like to deeply thank the invited speaker, Jean Paul Laumond, LAAS-CNRS France, for their excellent contribution in the field of humanoid robots. Finally, a word of appreciation for the hard work of the secretariat and volunteers.

Our deep gratitude goes to the Scientific Organisations that kindly agreed to sponsor the Conference, and made it come true.

We look forward to seeing more results of R&D work on Robotics at ROBOTICA 2010, somewhere in Portugal.

Paulo J.S. Gonçalves, Paulo J.D. Torres, Carlos M.O. Alves

Edited by P.J.S. Gonçalves, P.J.D. Torres, C.M.O. Alves - ISBN 978-972-99143-8-6

CONFERENCE PROGRAM

8:30-9:00	**Registration**
9:00-9:10	**Opening Session**
	Session I – Multi-Robot Systems and Architectures for Autonomous Robots **Chairs: Carlos Cardeira (IDMEC/IST, Portugal)**
9:10-9:30	**THREE-STATE MULTIROBOT COLLABORATIVE LOCALIZATION IN SYMMETRICAL ENVIRONMENTS** Fabrizio Abrate, Basilio Bona, Marina Indri, Stefano Rosa, Federico Tibaldi, Politecnico di Torino, Italy
9:30-9:50	**NEW TASK ALLOCATION METHODS FOR ROBOTIC SWARMS** Frederick Ducatelle, Alexander Foerster, Gianni Di Caro, Luca Gambardella, IDSIA, Switzerland
9:50-10:10	**HYBRID AND SECURE CONTROL ARCHITECTURE FOR MOBILE ROBOT NAVIGATION** Lounis ADOUANE, LASMEA - UMR CNRS, France
10:10-10:30	**MODULAR SCALABLE ARCHITECTURE FOR THE NAVIGATION OF THE ATLAS AUTONOMOUS ROBOTS** Miguel Oliveira, Procopio Stein, Jorge Almeida, Vitor Santos, University of Aveiro, Portugal
10:30-10:45	**Coffee-break**
	Session II – Recognition, Localization and Tacking **Chairs: Luís Paulo Reis (FEUP, Portugal)**
10:45-11:05	**PRELIMINARY RESULTS ON ROBUST GLOBAL LOCALIZATION: FAULT TOLERANCE AND ROBUSTNESS TEST ON PROBABILISTIC SHAPING** Luca Carlone, Basilio Bona, Politecnico di Torino, Italy
11:05-11:25	**LINE FOLLOWING AND GROUND VEHICLE TRACKING BY AN AUTONOMOUS AERIAL BLIMP** David Jerónimo, Ricardo Alcácer, Francisco Alegria, Pedro Lima, IST, Portugal
11:15-11:35	**REAL-TIME TEACHING OF INDUSTRIAL ROBOTS USING A SYNCHRONISED STEREOSCOPIC VISION SYSTEM** Paulo Malheiros, António Moreira, Paulo Costa, José Carlos Lopes, FEUP, Portugal
11:35-11:55	**LIGHT 3D MAPPING FOR SEARCH AND RESCUE OPERATIONS** Luís Lemos, Luís Paulo Reis, FEUP, Portugal
12:15-14:00	**Lunch**
14:00-15:00	**Invited Speaker** **Chair: Pedro Lima (ISR/IST, Portugal)**
	ON HUMAN AND HUMANOID MOTIONS Jean Paul Laumond Directeur de Recherche at LAAS-CNRS, Toulouse, France
	Session III – Competitions and Edutainment Robotics **Chair: Fernando Ribeiro (U Minho, Portugal)**
15:00-15:20	**EUROPEAN LAND ROBOTIC TRIAL 2008 (ELROB) - A REALISTIC BENCHMARK FOR OUTDOOR ROBOTICS** Bernd Brüggemann, Timo Röhling, Hans-Ludwig Wolf, Frank-E. Schneider, FGAN, Germany

15:20-15:40 **THE E-PUCK, A ROBOT DESIGNED FOR EDUCATION IN ENGINEERING**
Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-Christophe Zufferey, Dario Floreano, Alcherio Martinoli, EPFL, Switzerland

15:40-16:40 **Coffee-break and Poster Session**

CODE MIGRATION FROM A REALISTIC SIMULATOR TO A REAL WHEELED MOBILE ROBOT
José Gonçalves (ESTiG IPB), José Lima (ESTiG IPB), Paulo Malheiros (FEUP), Paulo Costa (FEUP), Portugal

THE DYNAMIC MODELING OF A BIRD ROBOT
Micael Couceiro (ISEC), Carlos Figueiredo, Nuno Ferreira (ISEC), J. Tenreiro Machado (ISEP), Portugal

PARALLEL TASK EXECUTION, MORPHOLOGY CONTROL AND SCALABILITY IN A SWARM OF SELF-ASSEMBLING ROBOT
Anders Christensen (ISCTE-PT), Rehan O'Grady (IRIDIA), Marco Dorigo (IRIDIA, CoDE, ULB), Belgium

A SLIPPING CONDITIONS OBSERVER IN WHEELED MOBILE ROBOT TRACTION
André Dias (ISEP), José Miguel Almeida (ISEP), Alfredo Martins (ISEP), João Sequeira (IST), Eduardo Silva (ISEP), Portugal

A COMPETITIVE DYNAMIC MODEL FOR DECISION MAKING IN AUTONOMOUS ROBOTS PERFORMING COOPERATIVE TASKS
Flora Ferreira, Estela Bicho, Wolfram Erlhagen, University of Minho, Portugal

MAZE SOLVING WITH ENHANCED MAP REPRESENTATION
João Oliveira, João Certo, Luís Paulo Reis, FEUP, Portugal

REAL-TIME PATH PLANNING USING A MODIFIED A* ALGORITHM
Pedro Costa, António Moreira, Paulo Costa, FEUP, Portugal

SUPERVISED GROUP SIZE REGULATION IN A HETEROGENEOUS ROBOTIC SWARM
Rehan O'Grady (IRIDIA), Carlo Pinciroli (IRIDIA), Anders Christensen (ISCTE - PT) Marco Dorigo (IRIDIA, CoDE, ULB), Belgium

AN ARTIFICIAL LOW COST ROBOT ARM SYSTEM PLAYING TIC-TAC-TOE
Mehmet Güzel (Newcastle University), Yasin Hınıslıoğlu (Ahmet Yesevi University, Turkey

Session IV – Mobile Robots and Humanoids

Chairs: Eduardo Silva (ISEP, Portugal)

16:40-17:00 **HEXAPOD WALKING AS EMERGENT REACTION TO EXTERNALLY ACTING FORCES**
Adam El Sayed Auf, Nico Dudek, Erik Maehle, University of Lübeck, Germany

17:00-17:20 **A PASSIVE SYSTEM APPROACH TO INCREASE THE ENERGY EFFICIENCY IN WALK MOVEMENTS BASED IN A REALISTIC SIMULATION ENVIRONMENT**
José Lima (ESTiG IPB), José Gonçalves (ESTiG IPB), Paulo Costa (FEUP), António Moreira (FEUP), Portugal

17:20-17:40 **ORBITAL OBSTACLE AVOIDANCE ALGORITHM FOR RELIABLE AND ON-LINE MOBILE ROBOT NAVIGATION**
Lounis ADOUANE, LASMEA - UMR CNRS, France

17:40-18:00 **ON THE LEARNING OF INTER-FIELD CONNECTIONS IN A DYNAMIC FIELD ARCHITECTURE FOR HUMAN-ROBOT COLLABORATION**
Emanuel Sousa, Estela Bicho, Wolfram Erlhagen, University of Minho, Portugal

18:00-18:10 **Prizes and Closing Session**

20:30 **Gala Dinner**

Three-State Multirobot Collaborative Localization in Symmetrical Environments

F. Abrate, B. Bona, M. Indri, S. Rosa and F. Tibaldi

***Abstract*— The paper addresses and solves the problem of multirobot collaborative localization in highly symmetrical 2D environments. Such environments can be encountered in various scenarios, e.g., in logistic applications where a team of rovers has to move along several parallel corridors in a large surface, to perform surveillance and monitoring tasks. Because of the environment symmetry, the most common localization algorithms may fail to provide a correct estimate of the position and orientation of the rover, if its initial position is not known, no specific landmark is introduced, and no absolute information (e.g., GPS) is available. The rover can estimate its position with respect to the walls of the corridor, but it could not determine in which corridor it is actually moving. The proposed algorithm is based upon a particle filter cooperative Monte Carlo Localization (MCL), as in [3], and implements a three-stage procedure that leads to global localization as well as accurate position tracking of each rover of a team. The simulation tests, which investigate different situations with respect to the number of involved rovers and their initial positions, show how the proposed solution can lead to the global localization of each rover, with a precision sufficient to be used as starting point for the subsequent rover tracking. A case study, demonstrating the robustness of the algorithm with respect to possible map variations, is also presented.**

I. Introduction

Multirobot collaboration is becoming one of the most challenging and promising research areas in mobile robotics. A team of rovers, suitably coordinated, can be used to execute complex tasks, as in surveillance, monitoring, and mapping, to cite only a few.

In these tasks the correct and reliable localization with respect to a known map is of capital importance, and represents one of the most fundamental problems in mobile robotics: a comprehensive study is reported in [16]. Potentially the multirobot case gives some interesting advantages, since the accuracy of the rovers pose estimates can be improved by a cooperative localization, even if wireless communication and data sharing problems must be considered. Extended Kalman Filters (EKF) and Monte Carlo Localization (MCL) methods are the most common approaches to rover localization. The data association problem is generally solved in the EKF approaches by multi modal distributions that approximate the position probability distribution, sometimes including iterations that propagate also an estimate of the posterior marginal densities of the unknown variances (see e.g., [6], [7], [9], [10], [12], [14]). The MCL methods approximate an arbitrary posterior probability distribution by using particle filters (see e.g., [3], [5], [11], [13]). Cooperative robust multirobot localization has also been proposed, in which unknown but bounded error models are employed for the sensor measurements (see e.g., [8], [15]).

Localization includes two distinct sub-problems: *position tracking* and *global localization*. In the first one, the rover pose is iteratively estimated while the robot moves starting from an initial condition, known with a given uncertainty, while the second one determines the absolute rover position with respect to a given environment map; this problem is the most challenging, since no information of initial pose – or a completely wrong estimation of the actual pose, as in the so-called *kidnapped robot* – is usually available.

Many of the papers cited above use multirobot and/or mutual localization to improve the quality of self-localization estimates that single rovers could achieve on the basis of their own sensors only, implicitly assuming that the measurements provided by such sensors would be sufficient to obtain a sufficiently correct, even if not precise, global localization. Unfortunately, without some external absolute information, a correct global self-localization cannot be performed by a single rover when the environment is wholly symmetrical.

Highly symmetrical environments are commonly encountered in large logistic spaces, like the one considered in this paper, which presents a team of rovers performing surveillance and monitoring tasks. A logistic space is similar to an indoor or outdoor warehouse, i.e., an area where logistic or transport companies receive, store and distribute large quantities of goods, as containers, cars, crates and other similar items. In order to achieve an efficient occupancy of the area and facilitate the handling operations, free corridors among the stored goods form a regular grid, as in Figure 1.

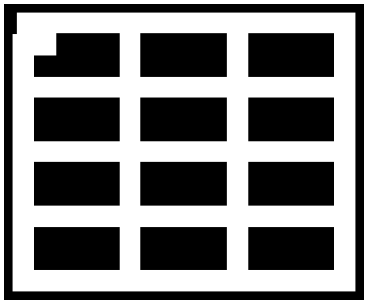


Fig. 1. The map of the environment.

The symmetry of the environment map prevents a reliable global self-localization of each rover when its initial position is unknown, no specific landmarks are introduced to discriminate each corridor, and no external information (e.g., from GPS) is available or exchanged with the other

This work was supported by Regione Piemonte under the "MACP4Log" grant (RU/02/26).

F. Abrate, B. Bona, M. Indri, S. Rosa, F. Tibaldi are with Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy fabrizio.abrate@polito.it

rovers. By using only its own sensors (odometry, laser scanner, sonar, etc.) a rover could estimate its position within the corridors, but cannot determine in which corridor is actually moving.

This paper presents a solution for the global localization problem in highly symmetrical environments, along the lines of the cooperative MCL approach by [3]. Moreover the proposed solution does not use any absolute sensor data (e.g., GPS), that could be unavailable in some indoor areas, but relies only on a minor asymmetry present in the map (see superior left-hand corner in Figure 1).

Simulation tests investigate different cases involving a variable number of rovers in the team, their initial positions, and some possible critical situations. The results show how the proposed algorithm, based upon a three-stage localization procedure, leads to the global localization of each rover, within a precision sufficient to be used as starting point for the subsequent rover tracking. A comparison with the previous work [2] is also presented.

The paper is organized as follows: Section II describes the proposed localization algorithm, called **3SMCL**, Section III describes and reports the different tests performed to demonstrate the effectiveness of the algorithm, and finally Section IV draws some conclusions and discusses future research perspectives.

II. The Three-State Multirobot Collaborative Localization (3SMCL)

A. Preliminaries

The ***T**hree-**S**tate **M**ultirobot **C**ollaborative **L**ocalization* algorithm (**3SMCL**) allows each member of a group of rovers moving in a highly symmetrical area (e.g., a large logistic space) to accurately localize itself and to correctly track its position over time. The **3SMCL** algorithm correctly operates for rovers endowed with at least sonar range sensors, a monocular or an omnivision camera. The camera is used to detect the positions of other rovers when they are in the field of view; the measurement precision may be improved using a laser range finder if available. A binary occupancy grid map of the environment is assumed to be available.

The algorithm has been conceived as a finite state machine, with three states: 1) *global localization*, 2) *undecided*, and 3) *position tracking*. We previously proposed in [2] an approach to solve the multirobot localization problem using an algorithm based on two states only (*Global Localization* and *Position Tracking*), applied to a completely symmetric environment, where absolute heading measurements are only intermittently available. The solution proposed in this paper ensures faster and more reliable localization with respect to the approach of [2], as explained in Section III-A, including also an intrinsic filtering action to possible overconfident estimates (due to multiple observations), thanks to the addition of the new *undecided* state, as it will be discussed in the next subsection.

At the beginning of the execution of the algorithm, each rover is set in the first state, as no information is available about its initial position, and hence it has to be *globally* localized with respect to the map. When the number of feasible hypotheses about its actual position becomes sufficiently limited (as detailed in the next subsection), the rover enters the *undecided* state. Finally, it switches to the *position tracking* state, when it is supposed to be correctly localized with a high degree of confidence. If a sudden increase in the localization error is detected, due for instance to kidnapping or failures in proprioceptive sensors (e.g., wheel encoders rupture) or changes in the map, the algorithm may switch again to the *undecided* state.

Let $\mathcal{R} = \{r_i : i = 1, \ldots, N_R\}$ be the set of rovers deployed in the area; with t we indicate the time variable that clocks the whole localization algorithm. With $d_i(t)$ we indicate data coming from the i-th robot proprioceptive and exteroceptive sensors at time t. In particular we have that

$$d_i(t) = \begin{cases} o_i(t) & \text{if proprioceptive measurement} \\ z_i(t) & \text{if exteroceptive measurement} \end{cases}$$

The proprioceptive measurement $o_i(t)$ is used to perform dead-reckoning, while the exteroceptive measurement $z_i(t)$ contains the range measurements given by the range sensors.

Each rover is able (a) to measure the positions of the other rovers in the field of view of its vision sensor in its local reference frame, concurrently with the **3SMCL** algorithm, (b) to transform the measurements in a global reference frame common to all rovers, using the estimated pose, and (c) to finally send these values to the detected rovers via a wireless link. The detection of the other rovers in the field of view is performed using the simulated vision sensor. Then the position estimation is computed using the laser scanner data, affected by their simulated noise.

Let k denote a time instant at which the position of the i-th rover is detected by a set of rovers $R_i(k) \subseteq R$, ($|R_i(k)|$ being its cardinality). The rovers belonging to $R_i(k)$ send their measurements to the i-th rover, which collects them in the following matrix:

$$h_i(k) = \begin{bmatrix} \hat{x}_i^{\,1}(k) \;\; \hat{y}_i^{\,1}(k) \\ \vdots \\ \hat{x}_i^{\,|R_i(k)|}(k) \;\; \hat{y}_i^{\,|R_i(k)|}(k) \end{bmatrix}. \quad (1)$$

Each row of (1) contains an estimate of the position of the i-th rover expressed in Cartesian global coordinates.

The set of all measurements received by the i-th rover up to time k is defined as $H_i^k = \{h_i(1), \ldots, h_i(k)\}$.

B. The algorithm

We now describe the core of the **3SMCL** algorithm, which runs onboard each rover and it is outlined in Algorithm 1.

The algorithm is basically organized as a Finite State Machine (FMS) with three states. The first state is the default initial state and it is active when the rover performs *global localization* (lines 17-60). Then, when the mean distance over the hypotheses on position of the rover is under a certain threshold, the algorithm enters the *undecided* state. This state indicates that there is one "dominant" position hypothesis, but the algorithm is still uncertain to decide if it corresponds to the true position or to one of the other possible symmetric positions (whose number depends on the geometry of the environment).

Input: χ^{t-1}, $d_i(t)$, $R_i(k)$, H_i^k, N_{min}, N_{max}, N_{hyp}, m
Output: $\Phi_i(t)$, $\phi_i^{best}(t)$
1 **if** *flag = 1* **then**
2 state = 'Position Tracking'
3 $[\chi^t, \mu_k, l]$ = **position_tracking**($d_i(t)$, χ^{t-1}, $h_i(k)$, l)
4 $[\Phi_i(t), \phi_i^{best}(t)]$= **DT_clustering**($\chi_t$)
5 **if** $l > n_{p2u}$ **then**
6 $[\mu_k]$ = **loc_perf**($\phi_i^{best}(t)$, $h_i(k)$)
7 **if** $\mu_k \geq \mu_{p2u}$ **then**
8 $flag = 0$; $l = 0$
9 **end**
10
11 **end**
12
13 **else**
14 **if** *flag = 2* **then**
15 state = 'Undecided'
16 **end**
17 **else if** *flag = 0* **then**
18 state = 'Global Localization'
19 **end**
20 initialize χ_t;
21 **if** $d_i(t) = o_i(t)$ **then**
22 $p_i(t)$ = **sample_motion_model**($d_i(t)$, $p_i(t-1)$);
23 **else if** $d_i(t) = z_i(t)$ **then**
24 $w_i(t)$ = **measurement_model**($d_i(t)$, $p_i(t)$, m)
25 $\bar{\chi}^t = \bar{\chi}^t + \langle p_i(t), w_i(t) \rangle$
26 **if** $R_i(k) = \emptyset$ **then**
27 χ^t = **KLD_1**($\bar{\chi}^t$, N_{min}, N_{max})
28 $[\Phi_i(t), \phi_i^{best}(t)]$ = **DT_clustering**(χ^t) ;
29 **else**
30 $l = l + 1$;
31 χ^t = **KLD_2**($\bar{\chi}^t$, N_{min}, N'_{max}, N_{hyp}, $h_i(k)$)
32 $[\Phi_i(t), \phi_i^{best}(t)]$ = **DT_clustering**(χ^t)
33 Φ_{best} = best_hyp_extraction($\Phi(t)$)
34 **if** *state = 'Undecided'* **then**
35 **if** $l > n_{u2p}$ **then**
36 $[\mu_k]$ = **loc_perf**($\phi_i^{best}(t)$, H_i^k)
37 **if** $\mu_k \leq \mu_{u2p}$ **then**
38 $flag = 1$; $l = 0$;
39 **end**
40 **end**
41 **if** $l > n_{u2g}$ **then**
42 $[\mu_k]$ = **loc_perf**($\phi_i^{best}(t)$, H_i^k)
43 **if** $\mu_k \leq \mu_{u2g}$ **then**
44 $flag = 0$; $l = 0$;
45
46 **end**
47
48 **end**
49 **if** *state = 'Global Localization'* **then**
50 **if** $\bar{p}_i(t) < \mu_{g2u}$ **then**
51 $flag = 2$;
52 **end**
53
54 **end**
55
56 **end**
57
58 **end**
59
60 **end**

Algorithm 1: The 3SMCL algorithm in pseudocode

When the localization performances are sufficiently accurate, as discussed in the following, the algorithms changes its state to *position tracking*.

Ideally, once reached the *position tracking* state, the rovers should never switch back to the *undecided* state. However, the algorithm continues to monitor the localization performances. In case of localization performance degradation, the algorithm switches again to the *undecided* state.

The algorithm is based on particle filters [16]; observing the pseudo-code in Algorithm 1 relative to the *global localization* and *undecided* states, it can be noticed the typical prediction phase at line 22 and the update phase at lines 24-25. The prediction phase computes the vector $p_i(t)$ containing the predicted pose (in terms of global coordinates $\{x,\ y,\ \theta\}$) for each particle, while the purpose of the update phase is twofold. It gives the vector $w_i(t)$ containing the importance factors for each particle, and it verifies whether matrix $h_i(k)$ contains position estimates outside the map. If this is the case, such estimates are weighted using a Bivariate Normal Distribution. Then, at line 26, the algorithm verifies if it has received a matrix of measurements $h_i(k)$ from other rovers of the set $R_i(k)$ at time k. If $R_i(k)$ is empty, a classic *Kullaback-Leibler Divergence* (KLD) Resampling occurs (see [16]); N_{min} and N_{max} are respectively the lower and upper bound of the number of particles N_{kld} employed in the resampling algorithm. If instead $R_i(k)$ is not empty, a modified version of the KLD Resampling has been implemented (line 31). The idea is to exploit the relative position measurements (contained in matrix $h_i(k)$) that the i-th rover receives from the other rovers of $R_i(k)$ to propagate the information about the few asymmetries of the environment. To achieve this goal, the algorithm distributes a subset N'_{kld} of the resampled particle set N_{kld} around the elements of the matrix $h_i(k)$. A new bound N'_{max} on the number of particles N_{kld} is set as:

$$N'_{max} = N_{max} - N_{hyp},$$

where N_{hyp} is the minimum number of particles that can be Gaussianly distributed around the elements of $h_i(k)$, and hence $N'_{kld} \geq N_{hyp}$. Therefore it holds that $N_{min} \leq N_{kld} \leq N'_{max}$. Loosely speaking, the proposed policy employs all the particles not used to approximate the belief of the rover after a laser resampling to approximate each belief contained in matrix $h_i(k)$.

After the resampling phase, a classic *Density-Tree* clustering [3] (lines 4, 28, 32) is always performed, that provides a set of N_h hypotheses $\Phi_i(t) = \{\phi_i^j(t)\}$, $j = 1, \ldots, N_h$, on the position of the i-th rover, among which the best hypothesis $\phi_{best}(t)$ is selected. Each hypothesis $\phi_i^j(t)$ is constituted by the predicted pose $p_i^j(t)$, its covariance matrix $\Sigma_i^j(t)$, and the associated weight $W_i^j(t)$, representing its level of confidence:

$$\phi_i^j(t) = \{p_i^j(t), \Sigma_i^j(t), W_i^j(t)\}. \tag{2}$$

The best hypothesis at time t is defined as

$$\phi_i^{best}(t) = \arg\max_{W_i^j}(\Phi_i(t)) = \{p_i^{best}(t), \Sigma_i^{best}(t), W_i^{best}(t)\}. \tag{3}$$

The mean distance among the hypotheses is defined as

$$\bar{p}_i(t) = \sum_{j=1}^{N_h} \frac{p_i^j(t)}{N_h}. \tag{4}$$

Switching among the three states is based on the following *accordance* function:

$$\mu_k = \sum_{q=k-n}^{k} \sum_{j=1}^{|R_i(q)|} \frac{\sqrt{(\hat{x}_i^j(q) - \hat{x}_i^{best}(t))^2 + (\hat{y}_i^j(q) - \hat{y}_i^{best}(t))^2}}{n|R_i(q)|}, \tag{5}$$

where n is the length of the sliding window used to compute the average in (5). If the rover is in the *undecided* state and it is verifying whether it can switch to *position tracking*, n is set equal to n_{u2p}. If the rover is in the *undecided* state and it is verifying whether it has to switch back to *global localization*, n is set equal to n_{u2g}. The

inner summation of (5) averages the distances among the elements of $h_i(k)$ and the best position hypotheses of the i-th rover. The outer summation of (5) performs a moving average of length n on the results of the inner summation. Therefore μ_k measures the accordance between the actual belief on the position of the i-th rover and the average of the beliefs that the other rovers have on its position at time k.

When μ_k is lower than a certain threshold μ_{u2p} (empirically determined) the algorithm switches to *position tracking*. This phase is aimed to track the position of the rover over time, and it is implemented in a classic way (see [16]). μ_k is computed also during the position tracking phase: if μ_k becomes greater than a given threshold μ_{p2u}, the algorithm switches again to *undecided*.

When the state of the rover is *global localization*, it switches to *undecided* if $\hat{p}_i(t)$ defined in (4) is smaller than a certain threshold μ_{g2u} (see Algorithm 1, line 50-52).

As a general final comment we have to say that belief propagation in a multi robot system can lead rovers becoming overconfident about their position estimation, if observations are integrated more times in a small period of time. In our scenario, this problem may be counteracted by using the finite state machine with three states and the mechanism which allows the rovers to swich from one state to another. Infact, since the continuous observations have a direct impact on the probabilistic representation of the belief of an observed rover, they have a slower influence on the modification of its state, which is the information we rely on when deciding if a rover is correctly localized. Therefore, the increase in the level of confidence due to multiple observations may be filtered in some way by the structure of the algorithm.

III. Simulation tests and Results

In this section we demonstrate the effectiveness of the proposed **3SMCL** algorithm, carrying out a series of localization experiments in simulation.

The software used to simulate the rovers and their environment is *MobileSim* [1]. It is based on the *Stage* library [4], and it simulates MobileRobots platforms. We perform experiments with a team of simulated Pioneer 3 DX rovers, endowed with sonar sensors and laser range finders. The simulator embeds a model of the behavior of sonar and laser range sensors, provides rover odometry pose estimation with cumulative error, and allows multiple rovers simulation.

The simulator has been improved by adding a simple simulated vision sensor and the support for communication among rovers.

We consider a simulated environment of a large logistic area (see Figure 1). The occupied black areas can be thought as containers or similar bulky items stored by transport societies before distribution. The dimension of the whole environment is 80×65 m, the black areas are 20×10 m and the corridors are 5 m wide.

The almost whole symmetry of the environment makes global localization a really difficult task, which the proposed **3SMCL** algorithm successfully performs, as the following experiments show in different situations.

A. Experiment 1

In this experiment we analyze the robustness of the **3SMCL** algorithm with respect to random variations in the initial position of the rovers. We define the following quantity:

$$e_i^{\rho}(t) = \sqrt{(x_i^{gt}(t) - \hat{x}_i^{best}(t))^2 + (y_i^{gt}(t) - \hat{y}_i^{best}(t))^2} \quad (6)$$

where $e_i^{\rho}(t)$ is the distance between the ground-truth position of the i-th rover $(x_i^{gt}(t), y_i^{gt}(t))$ and its position estimation given by the best hypothesis. The pose informations of the best hypothesis are given by $p_i^{best}(t)$ and can be extracted by $\phi_i^{best}(t)$, defined in (3).

We randomly initialize the pose of $N_R = 6$ rovers in free areas of the map, let them move according to a simple obstacle avoidance behavior, and monitor the localization error $e_i^{\rho}(t)$ for $i = 1, \ldots, N_R$ up to $t = 1750$ s. We repeat 100 times the experiment, each time setting randomly the initial position of the rovers. Since we are interested in evaluating the average localization error among the repetitions of the experiments, we define $\bar{e}_i^{\rho}(t)$ as the average of $e_i^{\rho}(t)$ for the i-th rover over 100 realizations.

The results are shown in Figure 2. The localization error

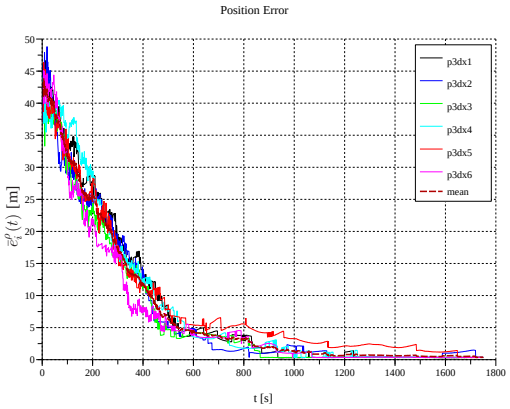


Fig. 2. Experiment 1: average localization errors.

$e_i^{\rho}(t)$, $i = 1, \ldots, N_R$ decreases approximately linearly for all the rovers, and the mean error among all the 6 rovers (dashed line in Figure 2) reaches a final value below 0.4 m. The **3SMCL** algorithm is thus not affected by variations in the initial positions of the rovers. This fact has an important impact on the application side, in particular when considering robotic applications in logistic spaces, since the algorithm does not require any particular initial formation of the rovers, avoiding any human intervention to initially place the rovers in a specific area of interest.

Figure 3 shows the comparison between the first switching time and the last switching time between the *undecided* and the *position tracking* states. The three bars on the left refer to the first switching time, while the three bars on the right refer to the last switching time. The blue bars indicate the minimum of the switching times, the red bars the maximum and the green bars the average. Observing in particular the green bars, it can be noticed that the last switching time occurs only three minutes after the first switching time. This means that the algorithm does not

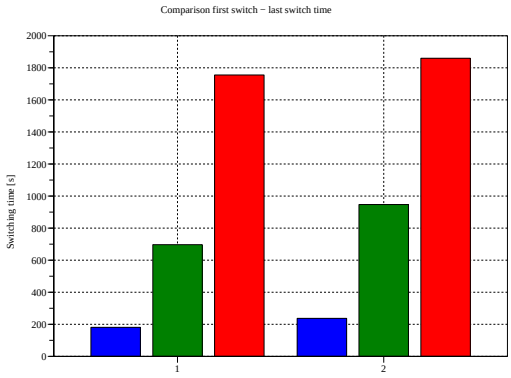


Fig. 3. Experiment 1: switching times to *position tracking* state. Maximum switching time (red bars). Average switching time (green bars). Minimum switching time (blue bars)

bounce for a long time between the *undecided* and the *position tracking* states.

Then we have also performed a comparison of the switching times between the **3SMCL** algorithm and the **SMCL** algorithm that we recently proposed in [2]. This algorithm was originally applied to a case study with a completely symmetric area and intermittent absolute heading measurements. In order to compare the performances of the two algorithms, we tested the **SMCL** algorithm in the same conditions of the **3SMCL** algorithm. The results are reported in Figure 4. The plot on the left shows the last

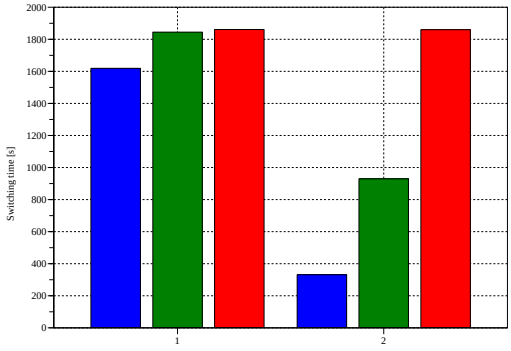


Fig. 4. Experiment 1: comparison between **SMCL** and **3SMCL**. Maximum switching time (red bars). Average switching time (green bars). Minimum switching time (blue bars)

switching time for the algorithm proposed in [2], while the plot on the right shows the last switching time for the **3SMCL** algorithm. Both the plots consider 30 runs of the algorithm with different initial rover positions. Observing in particular the green bars we can state that the time to reach *position tracking* doubles when the **SMCL** algorithm instead of **3SMCL** algorithm.

B. Experiment 2

This experiment has been designed to illustrate how the localization performance of the **3SMCL** algorithm is affected by the number of rovers in the team, in terms of position error.

We define the average position error among the N_R rovers of the team as:

$$E^{\rho}_{N_R}(t) = \sum_{i=1}^{N_R} \frac{\bar{e}^{\rho}_i(t)}{N_R} \tag{7}$$

where $\bar{e}^{\rho}_i(t)$ is the average for the i-th rover over 30 realizations. The results of the simulations for $N_R = 1, 3, 4, 5, 6, 7$ are reported in Figure 5 and in Table I.

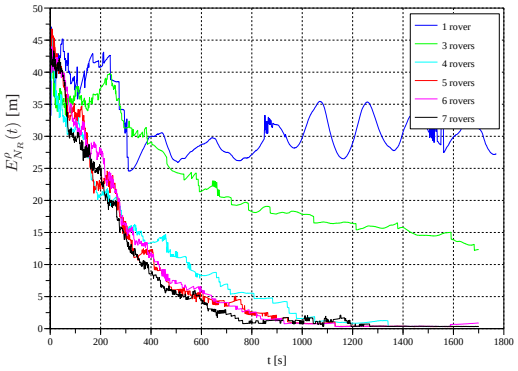


Fig. 5. Experiment 2: average localization errors.

TABLE I
CORRECT LOCALIZATION PERCENTAGE

# of rovers	Correct Localization Percentage
1	70%
3	83%
4,5,6,7	100%

It can be seen that one rover is not sufficient to resolve the ambiguity in localization, and also three rovers are not enough to assure correct localization, since rovers are able to localize themselves correctly only the 83% of the trials. As soon as four rovers are employed, the localization error goes below 2.5 m after nearly 1000 s, and all the trials are successful; increasing the number of rovers up to six and seven does not improve the performances in terms of steady state error, but slightly speeds up the convergence of the algorithm. This is particularly important in practical applications (e.g., handling hazardous events collaboratively), since the path planning algorithms become more effective. Of course, the specific number of rovers to be used in the proposed algorithm to ensure correct localization of all the members of the team depends also on the size of the area where the rovers move. Future investigations will be devoted to study the performance of the proposed algorithm with respect to variations of the ratio between the number of rovers and the area to be covered by the robot team.

C. Experiment 3

This experiment is aimed at demonstrating that, once the rovers are all in the *position tracking* state, the algorithm is robust even with respect to partial variations of the map. To show this robustness, we have set up a case study where $N_R = 6$ rovers are deployed in the same

logistic area of the Experiments 1 and 2. After all the rovers have reached the *position tracking* state, a fork lift is supposed to enter the logistic area in order to remove and add pallets. The fork lift moves ideally at a constant speed of 1 m/s, and removes or adds randomly a pallet in the map, employing 3 seconds to perform these operations. Tests have been performed with a decreasing occupancy percentage, starting from 90% of occupancy in steady state condition, up to 50% occupancy with step of 10%. It is important to say that the informations about the map variations are not communicated to the rovers, therefore the challenge here for the **3SMCL** algorithm is to maintain the *position tracking* condition and to keep the localization error low for all the rovers. Figure 6 shows the $E^{\rho}_{N_R}(t)$ (defined in (7)) for $N_R = 6$, considering only one realization of the experiment. The first plot shows

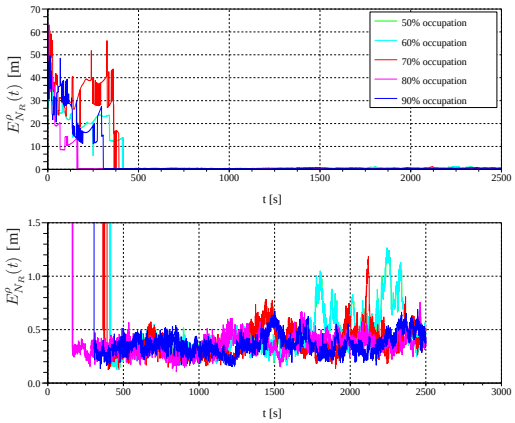


Fig. 6. Experiment 3: case study with variations in the map.

the localization error reduction while the rovers, in each test, reach the *position tracking* condition. On average after approximately 600 s all the rovers in each test are in *position tracking*, and the algorithm that simulates the intervention of a fork lift begins to modify the map.
Observing the second plot, which is simply a zoom of the first one, we see that the error increases, but only from 0.3 m to 0.5 m, with an upper bound of 1.2 m. Therefore the *position tracking* state of the proposed algorithm can be considered as invariant with respect to random variations in the map.

IV. Conclusions

The paper has shown how the problem of correct localization of rovers can be successfully solved by the proposed 3SMCL algorithm. Thanks to the cooperative action of all the rovers of the team, the knowledge about the small asymmetries of the environment is quickly diffused among the rovers, since they can communicate their relative positions when they are in the field of view, allowing them to remove the ambiguity on localization; moreover the approach allows to avoid the use of coded landmarks to distinguish different regions of the area. The proposed solution can then be usefully adopted in practical applications, where a team of vehicles must autonomously move in an area characterized by a regular grid of corridors or streets. The robustness of the 3SMCL approach with respect to the initial position of the rovers is an important advantage in practice, since the initial team formation can be completely arbitrary. Moreover, the automatic switch from the *position tracking* state to the *undecided* state prevents the occurrence of macroscopical errors, due to temporary sensor failures or rover kidnapping. Finally, it is worth noticing that, even if the performance increases with the number of rovers, the tests show that a team of only four rovers can perform the localization task with acceptable results for the considered area.

Future works will consider a more thorough study to investigate the influence of variations of the size of the environment on the performance of the algorithm. They will also include experimental tests, to confirm in practice the effectiveness of the proposed approach, which has been demonstrated in this paper only by means of simulation tests.

References

[1] "Mobilesim," Website, http://robots.mobilerobots.com/MobileSim/.
[2] F. Abrate, B. Bona, M. Indri, S. Rosa, and F. Tibaldi, "Switching multirobot collaborative localization in symmetrical environments," in *IEEE International Conference on Intelligent RObots Systems (IROS 2008), 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV)*, 2008.
[3] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.
[4] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *11th Int. Conf. on Advanced Robotics (ICAR 2003)*, 2003, pp. 317–323.
[5] D. Göring and H.-D. Burkhard, "Multi robot object tracking and self localization using visual percept relations," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 31–36.
[6] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 519–524.
[7] A. Martinelli, "Improving the precision on multi robot localization by using a series of filters hierarchically distributed," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 1053–1058.
[8] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino, "Simultaneous localization and map building for a team of cooperating robots: A set membership approach," *IEEE Trans. on Robotics and Automation*, vol. 19, no. 2, pp. 238–249, 2003.
[9] A. Mourikis and S. Roumeliotis, "Performance analysis of multirobot cooperative localization," *IEEE Trans. on Robotics*, vol. 22, no. 4, pp. 666–681, 2006.
[10] S. Panzieri, F. Pascucci, and R. Setola, "Multirobot localization using interlaced extended kalman filter," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 2816–2821.
[11] M. Peasgood, C. Clark, and J. McPhee, "Localization of multiple robots with simple sensors," in *IEEE Int. Conf. on Mechatronics and Automation*, 2005, pp. 671–676.
[12] G. Pillonetto and S. Carpin, "Multirobot localization with unknown variance parameters," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 1709–1714.
[13] I. Rekleitis, G. Dudek, and E. Milios, "Probabilistic cooperative localization and mapping in practice," in *IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 1907–1912.
[14] S. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, 2002.
[15] C. Taylor and J. Spletzer, "A bounded uncertainty approach to cooperative localization using relative bearing constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 2500–2506.
[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

New task allocation methods for robotic swarms

F. Ducatelle, A. Förster, G.A. Di Caro and L.M. Gambardella

***Abstract*— We study a situation where a swarm of robots is deployed to solve multiple concurrent tasks in a confined arena. The tasks are announced by dedicated robots at different locations in the arena. Each task requires a certain number of robots to attend to it simultaneously. We address the problem of task allocation: how can the robots of the swarm assign themselves to one of the announced tasks in a distributed and efficient way? We propose two novel methods: one relies on simple reactive mechanisms that are based on interaction through light signals, while the other uses a more advanced gossip-based communication scheme to announce task requirements among the robots. We evaluate both methods, and compare their performance. We also address scalability and robustness issues, in order to understand the usefulness of the methods in different swarm deployment conditions.**

I. Introduction

Swarm robotics is a form of collective robotics that takes its inspiration from social insects, such as colonies of ants, and from the related notion of swarm intelligence [20]. The central concept is to use large numbers of identical robots that individually have rather limited capabilities but when combined as a group are able to generate more complex behavior [18]. Swarm robotic systems work in a decentralized way and use only local control and communication. Typical properties of such systems are scalability, since the system can be extended to very large numbers of robots, flexibility, since robots can be dynamically added and removed, and fault tolerance, since individual robots are usually unimportant for the working of the system and there is no central point of failure [15].

In this paper, we address a problem of task allocation for robotic swarms. We consider a situation where a swarm of robots is deployed in a confined arena. Tasks appear at different locations in the arena and each task needs to be served by a certain number of robots simultaneously. The robots need to decide which task each of them will go to. The question we address is how this can be done efficiently in a distributed way, using only local communication.

We develop two mechanisms to deal with this problem. The first takes a very simple reactive approach. It is based on communication through light signals, whereby robots are attracted to one color of light and repulsed from another, in combination with random movements. The second mechanism is based on the explicit communication of structured information. When a task is announced, the number of robots it needs is communicated. This information is then passed on between the robots using a gossip mechanism, so that also robots further away can learn about it and react to it. In an evaluation study, we compare both mechanisms in terms of their efficiency, and we also investigate issues of scalability and robustness to communication failures.

In what follows, we first give a more detailed description of the problem we are addressing, and then discuss the related work in this area. Next, we describe the two task allocation mechanisms we propose. After that, we evaluate and compare the two systems. Finally, we draw conclusions and describe future work.

II. Problem description

The task allocation problem described here is situated in the broader context of a search task performed by a heterogeneous swarm consisting of two types of robots (this can also be seen as two separate swarms that collaborate). The first type are flying robots. They are called *Eyebots*. The second type are robots that move over the ground. They are called *Footbots*. Both types of robots are developed in the context of the EU-funded *Swarmanoid* project on heterogeneous swarm robotics [1], [2]. Images of these robots are shown in Figure 1. Within the Swarmanoid project, also a third type of robots is developed, the Handbots, which are left out of the discussion here to clarify the setup. In future work, they will perform part of the work that is here assumed to be done by the Footbots.

In the search task presented here, the heterogeneous swarm is requested to retrieve a particular target object from a room. To complete the task, the two types of robots

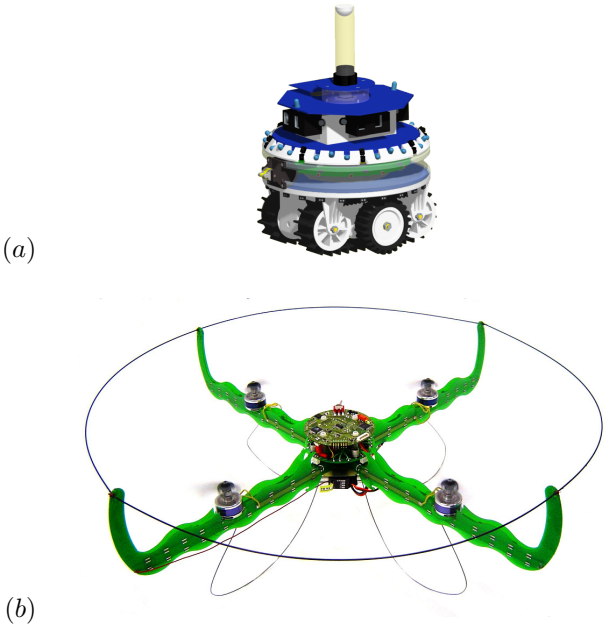


Fig. 1. Swarmanoid robots: (a) the Footbot (CAD draw) and (b) the Eyebot (prototype).

This work was supported by the SWARMANOID project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission under grant IST-022888. The information provided is the sole responsibility of the authors and does not reflect the Commission's opinion. The Commission is not responsible for any use that might be made of data appearing in this publication.

The authors are with the Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Lugano, Switzerland. {frederick,alexander,gianni,luca}@idsia.ch.

cooperate. The flying Eyebots execute a high-level search, obtaining an overview of the room and identifying areas where the target object might be found (e.g., if the target object is a specific book, the Eyebots identify parts of the room where books are present). The Footbots then visit these areas of interest in order to execute a detailed search for the specific target object. This comes down to a two-level search procedure.

From the Footbots' point of view, the areas of interest indicated by the Eyebots are tasks that are announced at different locations in the environment. Here we are interested in the way the Eyebots announce the tasks and the way the Footbots react to this in order to get an efficient spreading of the Footbots over the tasks. The problem we address starts with a simultaneous announcement of multiple tasks by Eyebots at different locations in the arena, and finishes when all Eyebots have gathered enough Footbots around them to complete the local task.

III. Related work

A large number of strategies exist to solve task allocation problems in multi-robot systems. This is partly due to the high number of possible variations of the problem. An effort to formulate a taxonomy for existing task allocation problems was presented in [9]. According to this taxonomy, our work can be classified as a "single-task robots, multi-robot tasks, instantaneous assignment" problem.

One of the most popular approaches to deal with task allocation problems is the *market based strategy* (see [8] for an overview). In such a system, an auctioneer announces tasks, and robots make bids, indicating their cost or utility to deal with the tasks. Based on the different bids, the auctioneer then decides which robot will be assigned to which task. Market based task allocation combines the efficiency of a centralized approach (the auctioneer decides with overview of the situation) with advantages of distributed approaches (much of the calculation is done by the individual robots preparing their bids) [8].

For swarm robotics, market based systems are not always the most appropriate approach. This is because they use the auctioneer as a central decision maker, use an explicit assignment of individuals to jobs, and need some form of global or at least long distance communication. All of these elements reduce the scalability and robustness of the system, and conflict with the distributed and purely local way of working of the swarm paradigm (even though an auction-based system can still be the most efficient [11]).

Instead, the most commonly used task allocation strategies in swarm robotics are *threshold based systems*. These systems are based on observations of task or role allocation processes in social insects, whereby tasks, often implicitly, send out a signal, and the insects/robots react to this signal if it surpasses an internal threshold. Due to differences in this threshold among individuals, task allocation emerges in proportion to the task's signal intensity [12]. In [6], a variation of this mechanism is presented whereby all robots have the same threshold and the differentiation comes from the variability in the local observation of the signal intensity by the individuals. Several works address the issue of dynamically adapting the internal threshold, e.g. according to the estimated job density [5], according to the own past success rate for the task [13], or according to a combination of internal, external and social cues [14]. Other work combines this system with other methods for more complex task allocation [22].

Some of the work on task allocation in swarm robotics diverges from this threshold based mechanism. E.g., in [10], robots adapt their own task allocation based on the observed density of tasks and robots addressing them. In [16], the required target distribution of robots over tasks is known by all robots, and the authors investigate different strategies to get to this distribution, evaluating the amount of communication that is used in each one.

The work presented here is somehow naturally related to auction based systems, as the Eyebots announcing the tasks play a central role and could easily function as auctioneers. However, since we are interested in designing a task allocation method for a swarm robotic system, we want to avoid the bidding and explicit task assignment applied in auction approaches.

IV. Two task allocation methods

In this section we describe the two task allocation methods we developed for swarm robotic systems. First we present the approach that uses light-based interaction, and next we describe the approach that is based on the exchange of structured information through gossiping. We assume that the Eyebots come down to ground level to announce the tasks to the Footbots.

A. Light-based task allocation

In the light-based task allocation system, Eyebots (and Footbots) use the multi-colored LEDs that are placed in a ring around their body [2] to influence Footbot behavior. The Footbots use their omnidirectional camera [2] to detect lights and react to them. The light-based task allocation system is built up around four basic behaviors:

- *Attraction to yellow light*. Footbots are attracted to yellow lights. When they see yellow lights in more than one direction, they go to the closest (they can estimate distance since it relates to vertical position in the field of view of the omnidirectional camera). Eyebots use yellow lights to attract Footbots to a task, as is shown in Figure 2(a). The number of yellow lights they use is proportional to the task size.
- *Repulsion from green light*. Footbots are repulsed from green lights: when they see green lights in the direction they are moving in, they turn away from it. In contrast to the attraction to yellow lights, repulsion is only active at a limited distance (set to about 50cm), so that the combination of a yellow and a green light attracts a Footbot up to a certain distance, after which the Footbot turns away. The repulsion behavior is used in two different ways. First, Eyebots show green lights in addition to the yellow lights, in order to control better the total number of Footbots they attract. This is shown in Figure 2(b). Second, Footbots that are attracted by yellow lights show green lights around, in order to repulse other Footbots from the tasks they are going to. This also limits the number of robots that come to serve a specific task. Moreover, it makes the Footbots that arrive at a task spread out. This is illustrated in Figure 2(c).

- *Internal frustration*. Each Footbot keeps an internal frustration level. This goes up whenever the Footbot experiences at the same time attraction and repulsion (as in the situation of Figure 2(c)), and goes down (but at a slower rate) when there is no repulsion. When the frustration reaches a fixed threshold, the Footbot executes an escape movement. This comes down to turning away from the direction in which attraction is observed and moving forward for a certain distance (enough to get outside the view of the attraction). This makes Footbots move away from events (points of attraction) that are being served by other robots (points of repulsion) and try other parts of the arena to find other tasks. The frustration mechanism is related to the internal motivations used in the Alliance architecture [17].
- *Random movements*. When none of the other three behaviors is active, the Footbots make random movements. These consist of turning in place for a random amount of time, and then moving forward for a random amount of time. This makes the Footbots execute a random search of the arena to find tasks.

The combination of these basic behaviors leads to a spreading of the Footbots over the different tasks, proportionally to the size of the tasks announced. An example is given in Figure 3, where the areas to be searched are bookshelves of different sizes. The three Eyebots indicate 1, 4 and 8 attracting lights respectively, attracting different numbers of Footbots. Note that the final number of Footbots is not necessarily exactly the same as the number of attracting lights: this number depends on the strength of the repulsion between the Footbots.

B. Gossip-based task allocation

The gossip-based task allocation system makes use of the *infrared range and bearing (Ir-RB) module* which is present on each Footbot and Eyebot [4]. This is an adaptation of the system presented in [19]. It consist of 26 infrared emitters and 16 receivers, placed all around the robot. Based on the quality of received signals, the system calculates an estimate of the relative bearing and range to other robots using the same system. The maximum range is 3 meters, and the precision is 20% for range estimates and 30 degrees for bearing estimates. The system also allows line-of-sight communication over the infrared signal with a nominal bandwidth of 40 Kbps. The advantage of this system is that received data can be related to information about the relative position of their sender.

The Eyebots use the Ir-RB system to send task announcement messages, in which they indicate the number of robots needed to complete the task. If they perceive Footbots nearby, they reduce this number. These task announcement messages are then forwarded by the Footbots and the other Eyebots in a gossiped way, i.e. each time they meet new neighbors, so that information about all tasks spreads among the swarm.

Each gossiped message contains information about all tasks a robot knows about. In detail, the following information is transmitted:

- *Robot ID*. The ID of the transmitting robot.

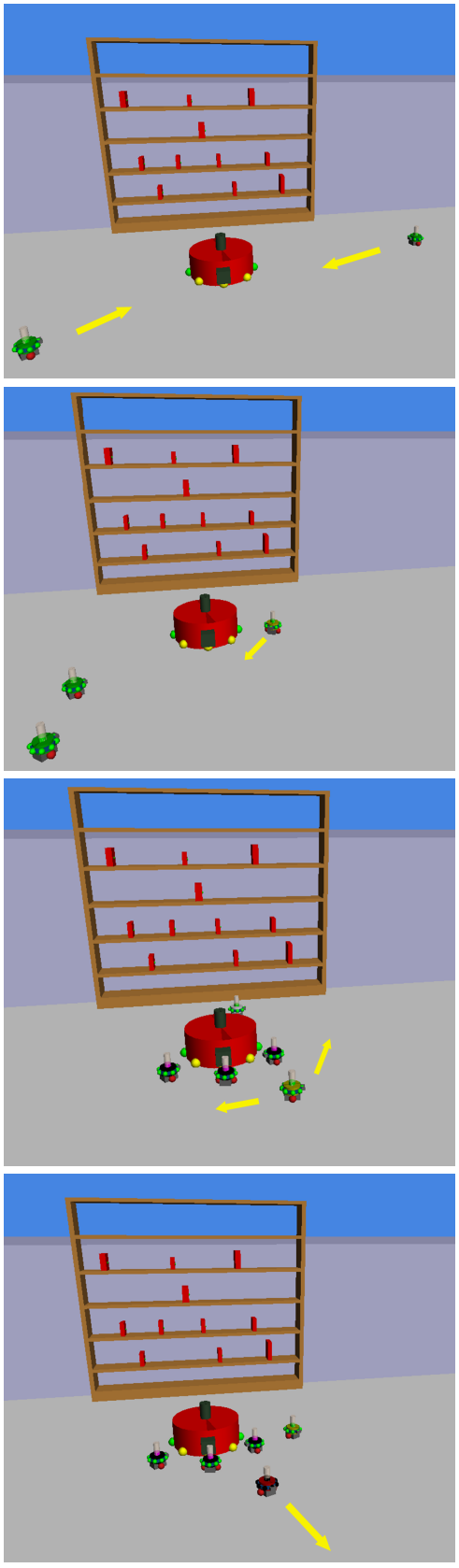


Fig. 2. Overview of the behaviors in the light-based task allocation mechanism: (a) attraction to yellow light, (b) repulsion from green light to get more precise placement, (c) repulsion from green light to fend off other robots, (d) evasive behavior when the frustration threshold is reached. In these figures, the color of the Footbot body illustrates its internal state: dark green means that it feels attraction, light green means repulsion, red means frustration, and black means that the Footbot is in place to perform the task. Yellow arrows show the movement direction of selected robots.

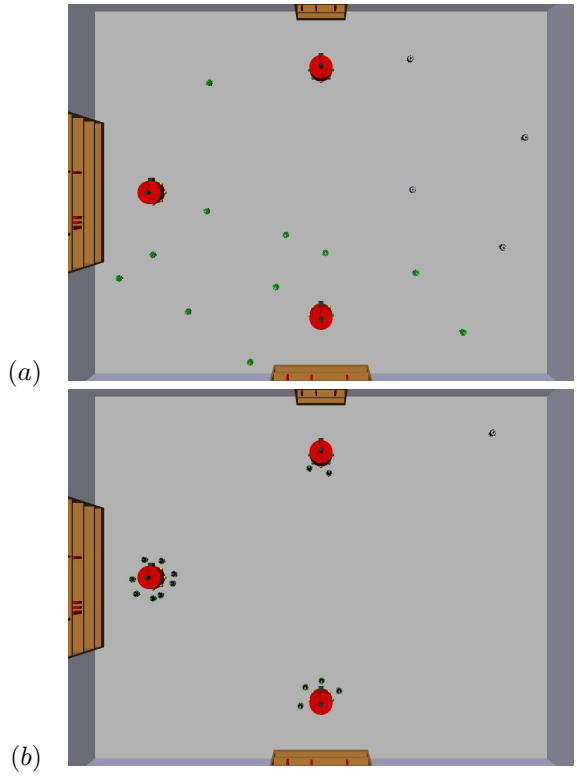


Fig. 3. An example of a task allocation scenario, (a) at the moment tasks are announced and (b) at the moment sufficient Footbots are assigned.

- *Number of tasks*. The number of tasks which the sender has received information about.
- *For each task*:
  - *Task ID*. This corresponds to the ID of the Eyebot announcing the task.
  - *Required workers*. The number of robots the task requires.
  - *Hops*. The number of hops (in terms of communication) to the task.
  - *Route length*. The distance to the task following the hops.
  - *Age*. The age of the information about the task.

When a robot receives information about a task, it needs to recalculate most of it before it can forward it in a message of its own. The number of required workers is decreased if the robot itself decides to go towards this task. The number of hops is increased by 1. The route length is increased by the distance to the robot the message was received from. Finally, the age value is increased. If the task age exceeds a threshold the task information is discarded and not re-sent in the next time-step.

The information about the tasks is used by the Footbots to decide on their actions. In general, the Footbot has four different behaviors:

- *Attraction to next task hop*. The nearest task is defined using the number of hops as first criterion and the route length as second criterion. The attraction of the task is only active when the number of additionally required robots for this task is greater than 0. The robot goes towards the next hop of the task (i.e., the robot it received the task information from), using the bearing information from the Ir-RB system. It steers on a circular path around the robot when it is close to it, until it sees the following hop. The reason to go hop by hop rather than straight to the task is to find obstacle free paths (since the Ir-RB communication only works over line-of-sight).
- *Internal frustration*. The robot has an internal frustration level value for each known task. This level increases with its distance to the task and with the number of robots that are near the task. The frustration decreases each time step with a small amount. When the frustration for a task passes a certain threshold, the robot will not go to this task.
- *Random movements*. This behavior is active when the robot does not know any task to go to (i.e., it knows only tasks that have enough robots or which it has a high frustration level for). The robot steers to a random position in its surrounding area. When the robot reaches this position or detects an obstacle on its way, another random position is generated.
- *Obstacle avoidance*. The obstacle information is based on proximity sensor values. When an obstacle is detected, a motion force in the opposite direction of the obstacle is added to the intended movement of the robot. When the robot is very close to the task itself the obstacle avoidance behavior is suppressed to stabilize already aggregated robots to tasks.

Compared to the light-based system described in Subsection IV-A, the gossip-based task allocation system is a bit more complex, as it requires the exchange and processing of structured information. However, unlike a market-based system (see section III) the task allocation is still entirely based on autonomous decisions of the individual robots, and no explicit assignment of robots to tasks is needed. An advantage compared to the light-based system is that information about tasks is disseminated over larger distances. Moreover, the fact that we use gossiping entails that the information can spread over the network of robots using only local message exchanges, without the need for full connectivity at any time. This way, information is flowing opportunistically between robots, which is an important advantage in sparse networks [23].

V. Evaluations and Comparisons

In the following we evaluate and compare the two task allocation methods. All tests are done using the Swarmanoid simulator [3], which uses the Open Dynamics Engine library [21] for the calculation of the physical movements and collisions of the robots and their environment, and the OpenGL library [7] for visualization.

We carried out experiments using three different setups, as shown in Figure 4: (a) an open environment (room size $6 \times 10m^2$), (b) an environment with obstacles (room size $6 \times 10m^2$), and (c) a maze (room size $9 \times 9m^2$). In setups (a) and (b), there are three Eyebots announcing tasks, whereby one task requires one Footbot, the second three, and the last five. In setup (c) there are two Eyebots announcing tasks, one of which requires four Footbots, and the other five. We carry out tests with increasing numbers of Footbots in the room: from 10 up to 40.

In the first place we are interested in efficiency: how quickly can the different task allocation systems assign the correct number of robots to each task. Figure 5

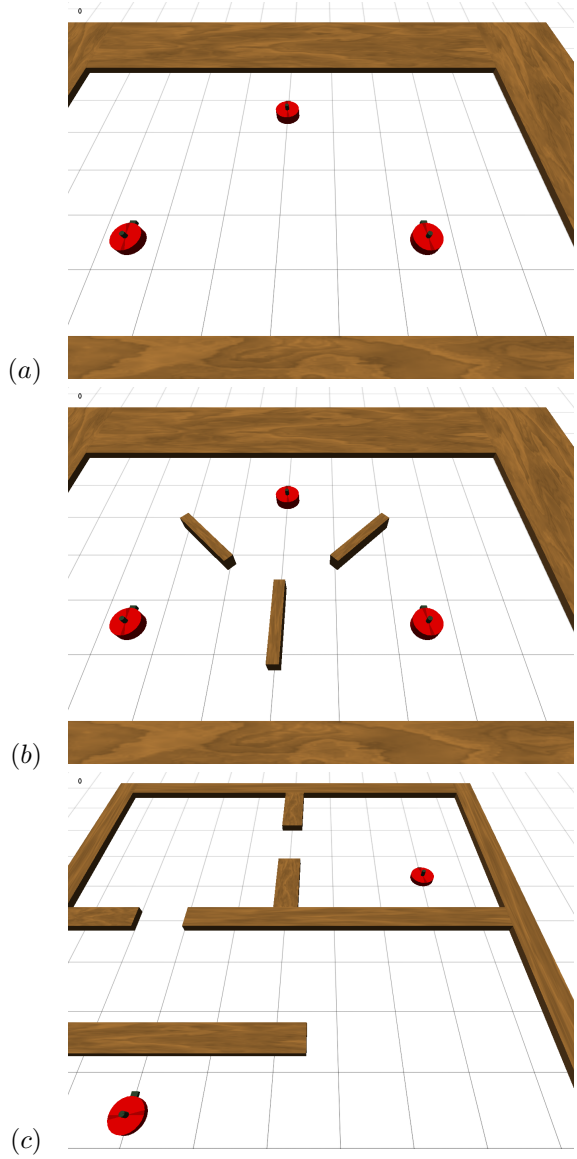


Fig. 4. The three different test setups we used: (a) an open environment, (b) an environment with obstacles and (c) a maze.

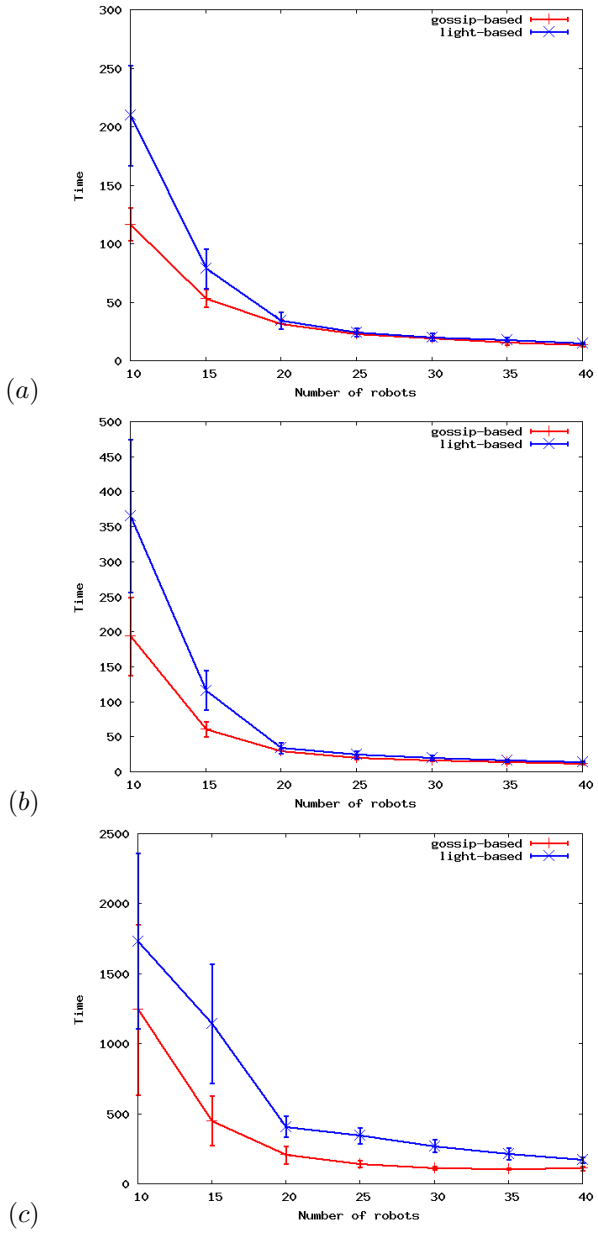


Fig. 5. The time needed to obtain the required allocation of robots to tasks in the three different setups, with increasing numbers of robots. Error bars show a 90% confidence interval using a t-test.

shows the time needed to reach the desired configuration in each of the three setups, with error bars indicating a 90% confidence interval. In all three setups, and for both algorithms, the performance improves with increasing numbers of Footbots, as more robots are available to serve the different tasks. A number of 20 robots or more seems to guarantee good performance in all setups, while especially in the maze setup performance suffers considerably for lower numbers of robots. In general we can see that the gossip-based task allocation method gives better results than the light-based approach: the presence of structured information propagated between the robots allows to make better decisions. The difference between the two methods decreases with an increasing number of Footbots though, and becomes insignificant for 20 robots or higher both in the open and the cluttered environment.

Despite its good performance, the gossip-based task allocation method has an important drawback compared to the light-based method: it requires the use of wireless communication in order to exchange information. Different from light signals, wireless packet transmission over the Ir-RB system may fail due to interference with other transmissions or with other infrared signals in general (e.g., signals produced by distance and proximity sensors). Interference increases with growing numbers of robots, and hence limits the scalability of the system.

Precise evaluations of the level of interference are not possible through our simulation system, and will therefore be addressed in later tests on the real robots. Here, we focus on the effect that the loss of communication packets has on the performance of the system, i.e. how robust the gossip-based system is with respect to packet loss. Figure 6(a) shows the bandwidth consumption per robot for the setup of Figure 4(b) with 15 Footbots, when only a fraction of the scheduled messages are sent (fraction 1 means 1 packet every control step of 100ms, while fraction 0.01 means 1 packet per 10 seconds). While these levels of bandwidth can nominally be supported by the Ir-RB system, it is expected that competition for the wireless channel and the

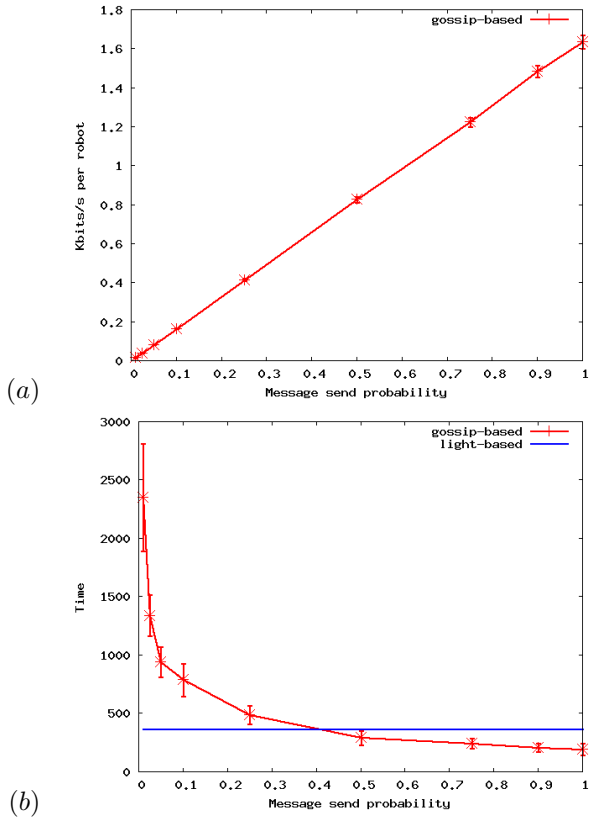


Fig. 6. For the setup of Figure 4(b) with 15 Footbots, (a) the bandwidth consumption and (b) the time required to obtain correct task allocation, when only a fraction of the messages are sent (e.g., 0.5 means that 50% of the scheduled messages are sent). The performance of the light-based approach in the same setup is also shown.

lack of centrally controlled medium access control will lead to packet loss even for a relatively low number of robots. Figure 4(b) shows the performance of the gossip-based system when only a fraction of the packets are sent. As can be seen, the performance suffers considerably when less than 25% of the messages are sent. At that point, the light-based approach becomes preferable.

In summary, the gossip-based approach works better than the light-based approach when the density of robots is low and operating conditions are good. However, it scores less good for scalability and robustness to communication failures, which are important issues in swarm robotics. Further tests on the real system have to indicate how severe these problems can be.

VI. Conclusions and Future Work

In this paper we have described two task allocation mechanisms for swarm robotic systems. The first is based on light signalling between robots, while the other relies on the gossiped exchange of structured information about tasks. Neither of the algorithms applies an explicit allocation of specific robots to tasks. In comparison tests, we found that the gossip-based algorithm is more efficient than the light-based algorithm in highly cluttered environments or when the total number of robots is low, while the difference between the algorithms is insignificant when more robots are deployed and the environment is not too complex. We also found that the gossip-based approach has limited robustness to packet loss and may therefore be less scalable. As a consequence, the light-based approach might be more appealing in large swarm robotic systems.

In future work we will complement the results we got via simulation with tests using the real Swarmanoid robots, in which we will explicitly investigate issues of interference, robustness to packet loss and scalability. Also, we want to investigate how to integrate the two different task allocation methods, so that the swarm can switch between them according to the deployment scenario, and get the best of both systems. Finally, we want to extend these mechanisms to the full Swarmanoid system, which means including the third kind of robot, the Handbot.

References

[1] Swarmanoid: Towards humanoid robotic swarms. `http://www.swarmanoid.org`.
[2] Hardware design. Internal Deliverable D4 of IST-FET Project *Swarmanoid* funded by the European Commission under the FP6 Framework, 2007.
[3] Simulator prototype. Internal Deliverable D7 of IST-FET Project *Swarmanoid* funded by the European Commission under the FP6 Framework, 2008.
[4] Swarmanoid prototype evaluation and redesign. Internal Deliverable D11 of IST-FET Project *Swarmanoid* funded by the European Commission under the FP6 Framework, 2008.
[5] W. Agassounon and A. Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems AAMAS-02*, 2002.
[6] W. Agassounon, A. Martinoli, and R.M. Goodman. A Scalable, Distributed Algorithm for Allocating Workers in Embedded Systems. In *Proc. of the IEEE Conf. on System, Man and Cybernetics*, 2001.
[7] OpenGL Architecture Review Board. *OpenGL(R) Reference Manual: The Official Reference Document to OpenGL, Version 1.4*. Addison-Wesley Professional, 2004.
[8] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7), July 2006.
[9] B.P. Gerkey and M.J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of robotics Research*, 23(9), September 2004.
[10] C. Jones and M.J. Mataric. Adaptive division of labor in large-scale minimalist multi-robot systems. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, October 2003.
[11] N. Kalra and A. Martinoli. A comparative study of market-based and threshold-based task allocation. In *Proceedings of Distributed Autonomous Robotic Systems (DARS)*, July 2006.
[12] M.J.B. Krieger and J.-B. Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1–2), 2000.
[13] T.H. Labella, M. Dorigo, and J.-L. Deneubourg. Self-organized task allocation in a group of robots. In *Proc. of the 7th Int. Symp. on Distributed Autonomous Robotic Systems (DARS)*, 2004.
[14] W. Liu, A.F.T. Winfield, J. Sa, J. Chen, and L. Dou. Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots. *Adaptive Behavior*, 15(3), 2007.
[15] A. Martinoli. Collective complexity out of individual simplicity. *Artificial Life*, 7(3), 2001.
[16] J. McLurkin and D. Yamins. Dynamic task assignment in robot swarms. In *Proceedings of Robotics: Science and Systems*, 2005.
[17] L.E. Parker. Alliance: An architecture for fault tolerant multi-robot cooperation. *IEEE Trans. on Robotics and Automation*, 14(2), 1998.
[18] L.E. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1), 2008.
[19] J. Pugh and A. Martinoli. Relative localization and communication module for small-scale multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006.
[20] A.J.C. Sharkey. Swarm robotics and minimalism. *Connection Science*, 19(3), September 2007.
[21] Russell Smith. *Open Dynamics Engine v0.5 User Guide*, 2006.
[22] D. Zhang, G. Xie, J. Yu, and L. Wang. Adaptive task assignment for multiple mobile robots via swarm intelligence approach. *Robotics and Autonomous Systems*, 55(7), 2007.
[23] Zhensheng Zhang. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *Communications Surveys and Tutorials, IEEE*, 8(1):24–37, 2006.

Hybrid and Safe Control Architecture for Mobile Robot Navigation

Lounis Adouane
LASMEA, UBP-UMR CNRS 6602, France
Email: Lounis.Adouane@lasmea.univ-bpclermont.fr

***Abstract*— This paper deals with a multi-mode control architecture for robot navigation while using hybrid control. It presents, an adaptive and flexible mechanism of control which guarantees the stability and the smoothness of the switch between controllers. Moreover, a specific safety mode is proposed and applied on the robot which navigates very close to obstacles. The overall architecture allows to obtain very smooth trajectories while guaranteeing very safe obstacle avoidance. Many simulations on different robot configurations and cluttered environments permits to confirm the reliability and the robustness of the proposed control architecture. In addition, an appropriate indicator is proposed to quantify the trajectory smoothness.**

I. Introduction

The control of mobile robot navigation in cluttered environment is a fundamental problem that has been receiving a large amount of attention. The main issues in this field is how to obtain accurate, flexible and reliable navigation? One part of the literature in this domain considers that the robot is fully actuated with no control bound and focuses the attention on path planning. Voronoï diagrams and visibility graphs [1] or navigation functions [2] are among these roadmap-based methods. However, the other part of the literature considers that to control a robot with safety, flexibility and reliability, it is essential to accurately take into account: robot's structural constraints (e.g., nonholonomy); avoid command discontinuities and set-point jerk, etc. Nevertheless, even in this method, there are two schools of thought, one uses the notion of planning and re-planning to reach the target, e.g., [3] and [4] and the other more reactive (without planning) like in [5], [6] or [7]. Our proposed control architecture is linked to this last approach. Therefore, where the stability of robot control is rigourously demonstrated and the overall robot behavior is constructed with modular and bottom-up approach [8].

To guarantee multi-objective criteria, control architectures can be elaborated in a modular and bottom-up way as introduced in [9] and so-called behavioral architectures [8]. These techniques are based on the concept that a robot can achieve a complex global task while using only the coordination of several elementary behaviors. In fact, to tackle this complexity, behavioral control architecture decompose the global controller into a set of elementary behavior/controller (e.g., attraction to the objective, obstacle avoidance, trajectory following, etc.) to master better the overall robot behavior. In this kind of control, it exists two major principles for behavior coordination: action selection and fusion of actions which lead respectively to competitive and cooperative architectures of control. In competitive architectures (action selection), the set-points sent to the robot actuators at each sample time are given by a unique behavior which has been selected among a set of active behaviors. The principle of competition can be defined by a set of fixed priorities like in the subsumption architecture [9] where a hierarchy is defined between the behaviors. The action selection can also be dynamic without any hierarchy between behaviors [10], [11]. In cooperative architectures (fusion of actions), the set-points sent to the robot actuators are the result of a compromise or a fusion between controls generated by different active behaviors. These mechanisms include fuzzy control [12] *via* the process of defuzzification, or the multi-objective techniques to merge the controls [13]. Among these cooperative architectures, schema-based principle [14], [8] is among the ones that has important diffusion in the scientific community. Moreover, it is considered in a lot of studies the investigation of the potentialities of the hybrid systems controllers [15] to provide a formal framework to demonstrate the robustness and the stability of such architecture. In their most simple description, hybrid systems are dynamical systems comprised of a finite state automaton, whose states correspond to a continuous dynamic evolution, and whose transitions can be enabled by particular conditions reached by the continuous dynamics themselves. Therefore, this formalism permits a rigorous automatic control analysis of the performances of the control architecture [16].

Specifically, obstacle avoidance controllers play a large role to achieve autonomously and safely the navigation of mobile robots in a cluttered and unstructured environments. An interesting overview of obstacle avoidance methods is accurately given in [17]. The proposed control architecture integrates obstacle avoidance method which uses limit-cycle vector field [18], [19], [20]. Moreover, it introduces an adaptive and flexible mechanism of control which guarantees the stability and the smoothness of the switch between controllers.

The rest of the paper is organized as follows. Section II gives the specificities of the proposed control architecture. In section III, the control architecture is applied to the task of navigation in the presence of obstacles. It presents the model of the considered robot and the different modules constituting the proposed control architecture. Section IV deals with safety mode mechanism. Section V is devoted to the description and analysis of the simulation results. This paper ends with some conclusions and further work.

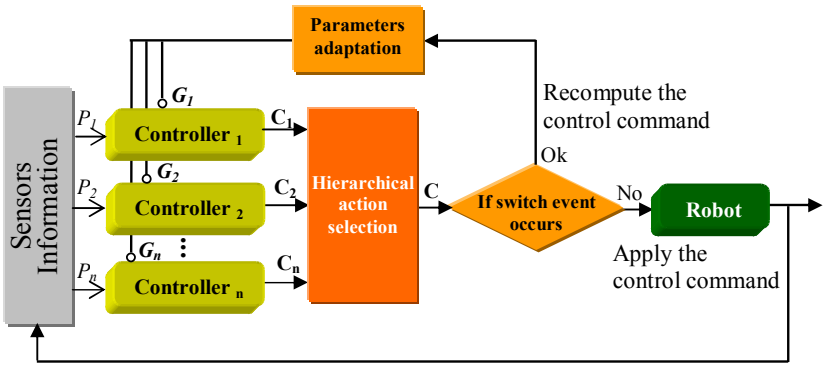


Fig. 1. The proposed hybrid control architecture for mobile robot navigation

II. Control architecture

The proposed control architecture (cf. Figure 1) is dedicated for mobile robots navigation in presence of obstacles. It permits to manage the interactions between different elementary controllers while guaranteing the stability and the smoothness of the overall control. Moreover, a specific "safety mode" is proposed to avoid undesirable robot behaviors. The robot can therefore have very smooth trajectories while guaranteeing safe obstacle avoidance. This control architecture permits for example to an autonomous applications of travelers transportation [21] to have more comfortable displacements while guaranteeing the security of passengers. The specific blocks composing this control are detailed below. Concrete control architecture applied in real task is proposed in section III.

A. Hierarchical action selection

The activation of one controller in favor of another is achieved completely with a hierarchical manner like the principle of the subsumption proposed initially by Brooks in [9]. Therefore, specific stimuli perceived by the robot (e.g., the robot-obstacle distance) are responsible to trigger the switch between controllers behaviors.

B. Controllers

Every controller F_i is characterized by a stable nominal law which is represented by the function:

$$F_i(P_i, S_i, t) = \eta_i(P_i, S_i, t) \quad (1)$$

with:

- P_i perceptions useful to the controller "i",
- S_i set-points given to the controller "i".

Otherwise, in order to avoid the important controls jumps at the time for example of the switch between controllers (e.g., from the controller "j" toward the controller "i" at the instant t_0), an adaptation of the nominal law is proposed, F_i becomes thus:

$$F_i(P_i, S_i, t) = \eta_i(P_i, S_i, t) + G_i(P_i, S_i, t) \quad (2)$$

with $G_i(P_i, S_i, t)$ (cf. Equation 3) a monotonous function that tends to zero at the end of a certain constant time "$T = H_i(P_i, S_i)$". The value of this constant depends on the criticality of the controller$_i$ to join quickly the nominal law $\eta_i(P_i, S_i, t)$. It constitutes thus the controller safety mode (cf. Section III-C for a specific example for obstacle avoidance controller).

$$G_i(P_i, S_i, t_0) = F_j(P_j, S_j, t_0 - \Delta t) - \eta_i(P_i, S_i, t_0) \quad (3)$$

where Δt represents the sampling time between two control set-points.

The definition of $G_i(P_i, S_i, t)$ allows to guarantee that the control law (cf. Equation 2) tends toward the nominal control law after a certain time T, thus:

$$G_i(P_i, S_i, T) = 0 \quad (4)$$

The function of adaptation $G_i(P_i, S_i, t)$ is updated by the "Parameters adaptation" block every time a hard control switch concerning the "i" controller occurs (cf. Section II-C) (cf. Figure 1). The main challenge introduced by this kind of control structure is to guarantee the stability of the updated control law (cf. Equation 2) during the period where $G_i(P_i, S_i, t) \neq 0$.

C. Parameters adaptation

This block has as input the "conditional block" (cf. Figure 1) that verifies if specific control switch event occurs. So, if it is the case then it must update "adaptation function" corresponding to the future active controller (cf. Equation 3). The different configurations which need the activation of parameters adaptation block are given below:

1) when a controller which should be active at the current "t" instant is different than the one which was active at the "t-Δt" instant,
2) when an abrupt transition in the set-points S_i of the controller$_i$ is encountered.

III. Navigation in presence of obstacles task

The navigation in an unstructured environment task has as objective to lead the robot to reach specific position in its environment (the target) while avoiding obstacles (cf. Figure 2).

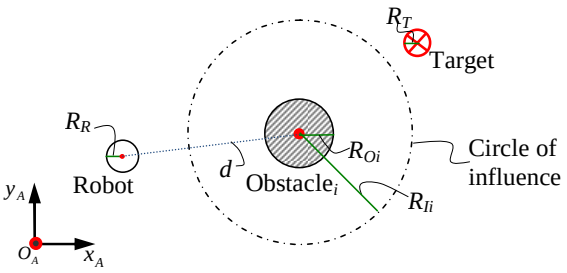


Fig. 2. The used perceptions for mobile robot navigation

The robot trajectory need to be safe, smooth and fast. One supposes in the setup that obstacles and the robot are surrounded by bounding cylindrical boxes with respectively R_O and R_R radii [22]. The target to reach is also characterized by a circle of R_T radius. Several perceptions are also necessary for the robot navigation (cf. Figure 2):

- d distance between the robot and the obstacle "i",
- R_{Oi} radius of the obstacle "i" to avoid,
- For each detected obstacle we define a *circle of influence* with a radius of $R_{Ii} = R_R + R_{Oi} + Margin$. *Margin* corresponds to a safety tolerance which includes: perception incertitude, control reliability and accuracy, etc.

A. Model of the used robot

Before proposing appropriate elementary controllers to achieve the considered task, it is important to know the robot model. Its model is given by the kinetic model of a unicycle robot which is given by (cf. Figure 3):

$$\dot{\xi} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l_2\cos\theta - l_1\sin\theta \\ \sin\theta & -l_2\sin\theta + l_1\cos\theta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \quad (5)$$

with:

- x, y, θ: configuration state of the unicycle at the point "P_t" of abscissa and ordinate (l_1, l_2) according to the mobile reference frame (X_m, Y_m),
- v: linear velocity of the robot at the point "P_t",
- w: angular velocity of the robot at the point "P_t".

Knowing the model of the robot as well as the task to achieve, one presents below the controller of *Attraction to the target* and the *Obstacle avoidance* controller which are necessary to the mobile robot navigation in presence of obstacles. The set of these controllers will be synthesized while using the Lyapunov theorem.

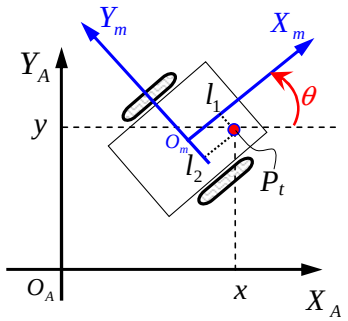


Fig. 3. Robot configuration in a cartesian reference frame

B. Attraction to the target controller

This controller guides the robot toward the target which is represented by a circle of center (x_T, y_T) and of R_T radius (cf. Figure 2). The used control law is a control of position at the point P_t = (l_1, 0) (cf. Figure 3). As we consider a circular target with R_T radius, thus, to guarantee that the center of robot axis reaches the target with asymptotical convergence, l_1 must be $\leq R_T$.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l_1\sin\theta \\ \sin\theta & l_1\cos\theta \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} = M \begin{pmatrix} v \\ w \end{pmatrix} \quad (6)$$

with M invertible matrix.

The errors of position are: $\begin{cases} e_x = x - x_T \\ e_y = y - y_T \end{cases}$

The position of the target is invariable according to the absolute reference frame (cf. Figure 2) $\Rightarrow \begin{cases} \dot{e}_x = \dot{x} \\ \dot{e}_y = \dot{y} \end{cases}$

Classical techniques of linear system stabilization can be used to asymptotically stabilize the error to zero [23]. We use a simple proportional controller which is given by:

$$\begin{pmatrix} v \\ w \end{pmatrix} = -K \begin{pmatrix} \cos\theta & -l_1\sin\theta \\ \sin\theta & l_1\cos\theta \end{pmatrix}^{-1} e = - K \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta/l_1 & \cos\theta/l_1 \end{pmatrix} \begin{pmatrix} e_x \\ e_y \end{pmatrix} \quad (7)$$

with $K > 0$ and $l_1 \neq 0$ (cf. Figure 3).

To guarantee the right transition between controllers as described in section (II-B), the modification of the controller law (7) must be done, it becomes thus:

$$\begin{pmatrix} v \\ w \end{pmatrix} = -K \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta/l_1 & \cos\theta/l_1 \end{pmatrix} \begin{pmatrix} e_x \\ e_y \end{pmatrix} + \begin{pmatrix} G_{A\_v}(t) \\ G_{A\_w}(t) \end{pmatrix} \quad (8)$$

Let's consider the following Lyapunov function

$$V_1 = \tfrac{1}{2}d^2 \quad (9)$$

with $d = \sqrt{e_x^2 + e_y^2}$ (distance robot-target). The proposed controller is asymptotically stable if $\dot{V}_1 < 0$. After some simplification we can deduce that:

$$K > \frac{-(G_{A\_v}(t)e_x + G_{A\_w}(t)e_y)}{e_x^2 + e_y^2} \quad (10)$$

As we said above $G_{A\_v}(t)$ and $G_{A\_w}(t)$ functions which must be chosen with respect to the constraints given in section (II-B). In fact, the absolute value of these functions must be monotonically decreasing according to the time "t", they will be equal to zero after a certain time "T". Therefore, in order to have always bounded K, we must have: $-(G_{A\_v}(t)e_x + G_{A\_w}(t)e_y) \leq e_x^2 + e_y^2$. Thus, to guarantee this assertion, it is sufficient to impose that $G_{A\_v}(t)$ decreases more quickly to zero than e_x and also that $G_{A\_w}(t)$ decreases more quickly to zero than e_y.

C. Obstacle avoidance controller

The objective of this controller is to avoid obstacles which hinder the robot movement toward the objective. In what follows we will give only few details about the overall obstacle avoidance algorithm in order to focus the attention only around the proposed mechanisms of control which can guarantee at the same time: the stability and the smoothness of the switch between controllers. Accurate details about the proposed obstacle avoidance algorithm is given in [20].

To implement the obstacle avoidance behavior, limit-cycles was used [18], [24], [20]. The differential equations giving these desired robot trajectories are given by two differential equations:

- For the clockwise trajectory motion (cf. Figure 4(a)):

$$\begin{aligned} \dot{x}_s &= y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= -x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \tag{11}$$

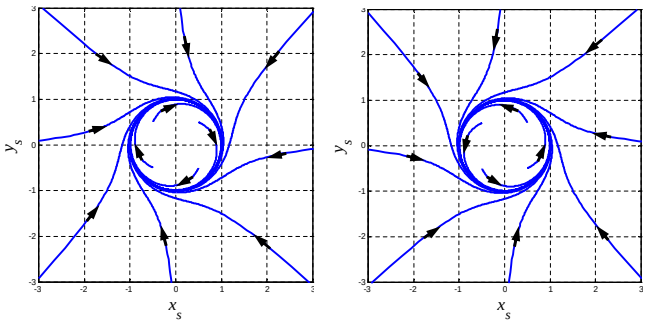
- For the counter-clockwise trajectory motion (cf. Figure 4(b)):

$$\begin{aligned} \dot{x}_s &= -y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \tag{12}$$

where (x_s, y_s) corresponds to the position of the robot according to the center of the convergence circle which is characterized by an R_c radius. Figure 4 shows that the circle of "$R_c = 1$" is a periodic orbit. This periodic orbit is called a limit-cycle. Figure 4(a) and 4(b) show the shape of equations (11) and (12) respectively. They show the direction of trajectories (clockwise or counter-clockwise) according to (x_s, y_s) axis. The trajectories from all points (x_s, y_s) including inside the circle, move towards the circle.

Summarily, the obstacle avoidance algorithm [20] follow these steps:

- Detect the most disturbing obstacle which avoids the robot to reach the target (cf. Figure 2). (x_{Oi}, y_{Oi}) and R_{Ii} are respectively, the position and the radius of influence circle of corresponding obstacle$_i$. (x_{Oi}, y_{Oi}) constitutes the center of the limit-cycle.
- According to specific stimuli, the direction of avoidance (clockwise or counter-clockwise) is obtained,
- Robot go into the orbit of the obstacle$_i$ to avoid (**Attractive phase**). The radius of the limit cycle to follow is given by $R_c = R_{Ii} - \xi$, , with ξ a small constant value as $\xi \ll$ *Margin* (cf. Section III) [20].



(a) Clockwise (b) Counter-Clockwise

Fig. 4. Shape possibilities for the used limit-cycles

- Robot go out the orbit of the obstacle$_i$ (**Repulsive phase**). The radius of the limit cycle to follow is given by $R_c = R_c + \xi$.

Controller law definition: The proposed control law which permits to follow these trajectories is an orientation control, the robot is controlled according to the center of its axle, i.e., while taking (l_1, l_2) = (0, 0) (cf. Figure 3). The desired robot orientation θ_d is given by the differential equation of the limit-cycle (11) or (12) as:

$$\theta_d = \arctan(\frac{\dot{y}_s}{\dot{x}_s}) \tag{13}$$

and the error by

$$\theta_e = \theta_d - \theta \tag{14}$$

We control the robot to move to the desired orientation by using the following nominal control law:

$$w = \dot{\theta}_d + K_p\theta_e \tag{15}$$

with K_p a constant > 0 and $\dot{\theta}_e$ is given by:

$$\dot{\theta}_e = -K_p\theta_e \tag{16}$$

To guarantee the right transition between controllers as described in section (II-B), the modification of the controller law (7) must be done, it becomes thus:

$$w = \dot{\theta}_d + K_p\theta_e + G_O(t) \tag{17}$$

where $G_O(t)$ the adaptive function.

$\dot{\theta}_e$ is given then by:

$$\dot{\theta}_e = -K_p\theta_e - G_O(t) \tag{18}$$

Let's consider the following Lyapunov function

$$V_2 = \tfrac{1}{2}\theta_e^2 \tag{19}$$

$\dot{V}_2$ is equal then to $\theta_e\dot{\theta}_e = -K_p\theta_e^2 - G_O(t)\theta_e$. To guarantee that the proposed controller is asymptotically stable we must have $\dot{V}_2 < 0$, so:

$$K_p > -\frac{G_O(t)}{\theta_e} \tag{20}$$

where $G_O(t)$ function is chosen with respect to constraints given in section II-C and to the fact that it decreases more quickly to zero than θ_e.

D. Hierarchical action selection block

The activation of a controller in favor to another is achieved according to complete hierarchy as given below:

if *It exists at least one constrained obstacle.* {*i.e.,* $d \leq R_{Ii}$ *(cf. Figure 2)* } **then**
  Activate obstacle avoidance controller
else
  Activate the attraction to the target controller
end

Algorithm 1: Hierarchical action selection

E. Parameters adaptation block

In the applied navigation, the "conditional" block activate the "parameters adaptation" block (cf. Figure 1) when at least one of the following switch events occurs:

- the "Hierarchical action selection" block chose to switch from one controller to another,
- the "obstacle avoidance" algorithm chose an other obstacle to avoid,
- the "obstacle avoidance" controller switch from attractive phase to the repulsive phase (cf. Section III-C).

IV. Obstacle avoidance safety mode

The adaptive function $G_O(t)$ (cf. Equation 17) permits mainly to obtain smooth control when a switch event occurs. However, during "T" time (cf. Section II-B) the obstacle avoidance controller is far from its nominal law (given when $G_O(t) \neq 0$) and the robot can collide with obstacles [7]. Therefore, to insure the smoothness of the control without neglecting the robot safety, G_O will be parameterized according to the robot-obstacle distance "d" (cf. Figure 2), G_O becomes thus:

$$G_O(t,d) = A.e^{Bt} \tag{21}$$

where:

- A value of the control difference between the control at the instants "$t - \delta t$" and "t" (cf. Equation 3),
- $B = Log\left({}^{\varepsilon}/_{|A|}\right) {}^{1}/_{T(d)}$

  with:
  - ε very small constant value ≈ 0,
  - $\begin{cases} T(d) = T_{max} & \text{if } d > R_{Ii} \\ T(d) = c.d + e & \text{if } R_{Ii} \geq d \geq R_{Ii} - (p.Margine) \\ T(d) = 0 & \text{if } d < R_{Ii} - (p.Margine) \end{cases}$

    where:
    - $Margine$ defined in section III,
    - p positive constant < 1 which allows to adapt the maximum distance "d" where the adaptive function must be resetting to zero. As small as p is, more the priority is given to the safety behavior instead to the smoothness of controllers switch,
    - $c = \left[{}^{T_{max}}/_{p.Margine}\right]$
    - $e = \left[T_{max}(Margine - R_{Ii}/p)\right]/_{Margine}$

Therefore, $T(d)$ goes from T_{max} until 0 while following a linear decrease. If the robot is out of R_{Ii} than $T = T_{max}$ and decrease linearly to become 0 when $d < R_{Ii} - (p.Margine)$. This function permits thus, when $d < R_{Ii} - (p.Margine)$, to remove completely the effect of adaptive control (which promote the smoothness of control) and insures thus the complete safety of the robot navigation.

V. Simulation results

Figure 5 shows the smoothness of the robot trajectory when the proposed control architecture is applied in cluttered environment. It shows also the clockwise and counter-clockwise robot obstacle avoidance. Figure 6 shows the progress of v and w controls when the adaptive functions are used. These controls are thus less abrupt and smoother than those obtained without adaptive functions (cf. Figure 7).

Moreover, to quantify the smoothness of the control setpoints, we propose this two indicators:

$I_v = \int_0^{T_{Simulation}} |v'|dt$ and $I_w = \int_0^{T_{Simulation}} |w'|dt$

where v' and w' are the derivative functions of v and w. According to these indicators we can observe a significant gain in smoothness of v and w controls which are equal respectively to 6% and 50%.

The seconde step of simulations permits to demonstrate the relevance of the proposed safety mode specially when the robot navigate very close to obstacles. Figure 8 shows the case where obstacle avoidance controller apply and do not apply the safety mode (cf. Section IV). When it do not apply it, the robot hit the obstacle (cf. Figure 8(a)).

Figure 9 gives the progress of adaptive function when the safety mode is applied (cf. Figure 9(b)) or not (cf. Figure 9(a)). We observe in figure 9(b) that the maximal time T_{max} to achieve the interpolation decreases every time that the robot moves dangerously closer to the obstacle.

Figure 10 shows that the overall proposed structure of control is stable, and that the Lyapunov function attributed to each controller $V_i|_{i=1..2}$ decreases always asymptotically to

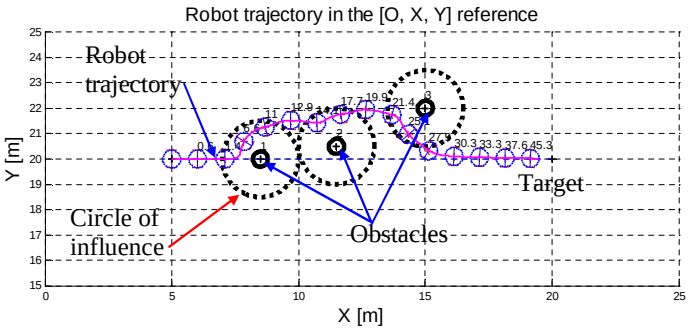


Fig. 5. Smooth robot trajectory obtained with the proposed control architecture

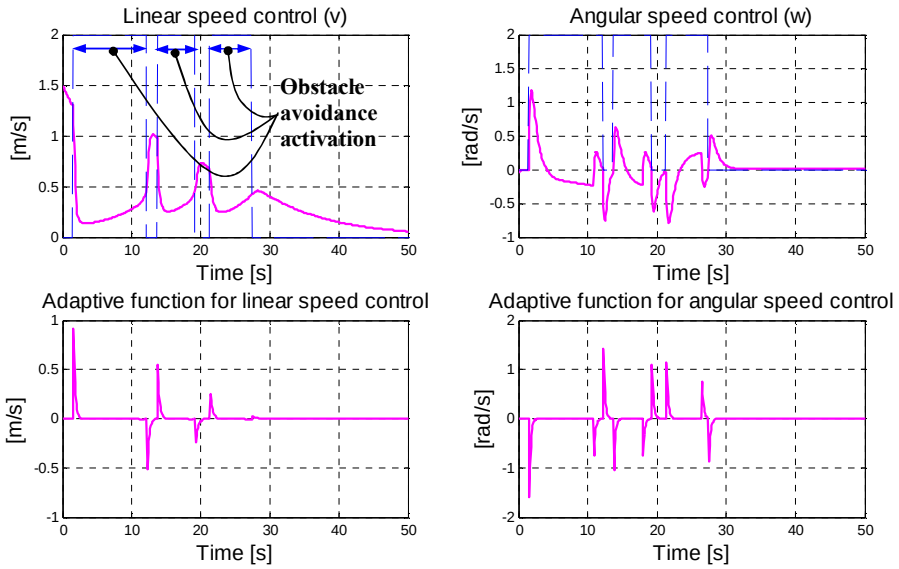


Fig. 6. Control with adaptive mechanism

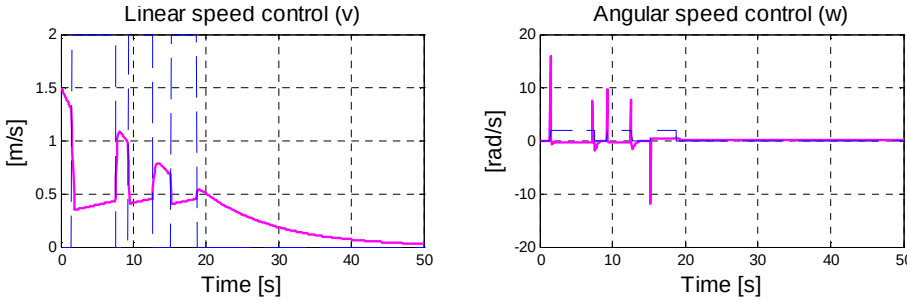


Fig. 7. Control without adaptive mechanism

the equilibrium point even when the adaptive safety mode is applied.

VI. Conclusion and Further Work

In this paper, a hybrid and safe multi-controller architecture is proposed and applied to the navigation of mobile robot in cluttered environments. The stability and the smoothness of the switching between these multi-control's modes are guaranteed according to a specific adaptive mechanism. Moreover, to obtain safer robot navigation an appropriate safety mode is proposed and experimented in cluttered environment. The robot can therefore have very smooth trajectories while guaranteeing obstacle avoidance. Many simulations confirm the robustness of the proposed control architecture. Future work will first test the proposed control architecture on the CyCab vehicle [21]. The second step is to adapt the proposed control structure to more complex tasks like navigation in highly dynamical environments.

References

[1] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[2] E. Rimon and D. E. Koditschek, "Exact robot navigation using artficial potential flelds," *IEEE Transactions on Robotics and Automation*, vol. 8(5), pp. 501–518, Oct. 1992.

[3] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21(5), pp. 864–874, Oct. 2005.

[4] D. C. Conner, H. Choset, and A. Rizzi, "Integrated planning and control for convex-bodied nonholonomic systems using local feedback," in *Proceedings of Robotics: Science and Systems II*. Philadelphia, PA: MIT Press, August 2006, pp. 57–64.

[5] M. Egerstedt and X. Hu, "A hybrid control approach to action coordination for mobile robots," *Automatica*, vol. 38(1), pp. 125–130, 2002.

[6] J. Toibero, R. Carelli, and B. Kuchen, "Switching control of mobile robots for autonomous navigation in unknown environments," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1974–1979.

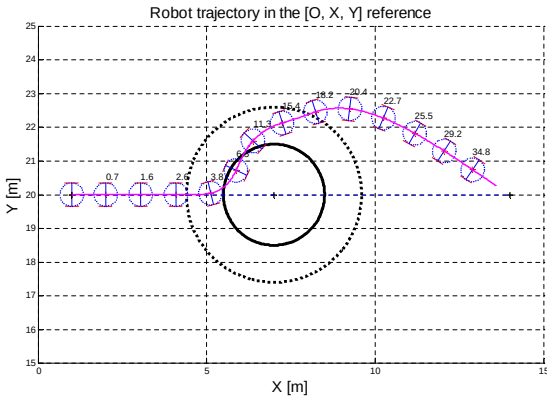
[7] L. Adouane, "An adaptive multi-controller architecture for mobile robot navigation," in *10th IAS, Intelligent Autonomous Systems*, Baden-Baden, Germany, July 23-25 2008, pp. 342–347.

[8] L. Adouane and N. Le Fort-Piat, "Behavioral and distributed control architecture of control for minimalist mobile robots," *Journal Européen des Systèmes Automatisés*, vol. 40, no. 2, pp. pp.177–196, 2006.

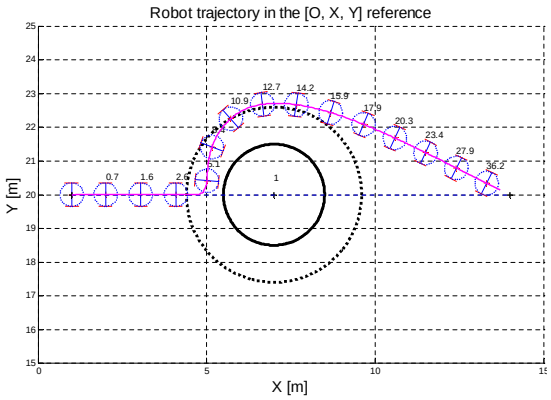
[9] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. pp.14–23, March 1986.

[10] P. Maes, "The dynamics of action selection," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI)*, Detroit, 1989, pp. 991–97.

[11] M. J. Matarict', M. Nilsson, and K. Simsarian, "Cooperative multi-robots

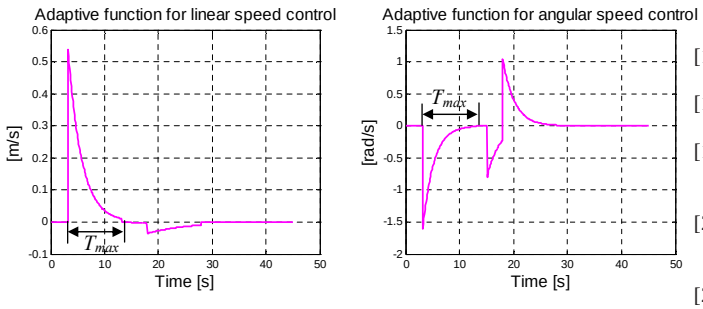


(a) Without safety mode

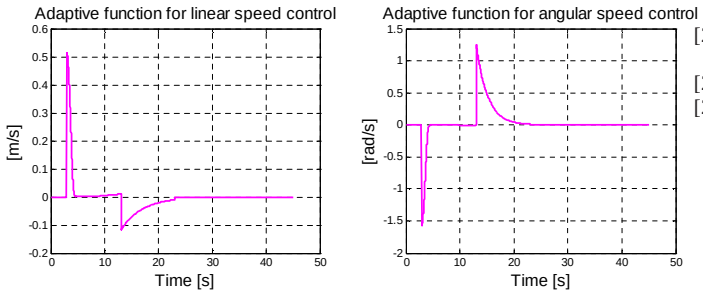


(b) With safety mode

Fig. 8. Robot trajectories



(a) Without safety mode



(b) With safety mode

Fig. 9. Adaptive function progress

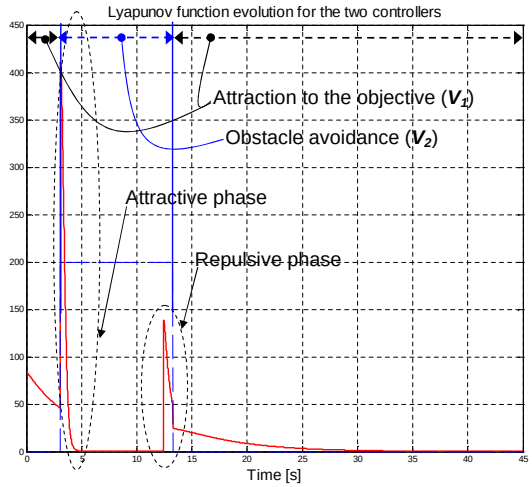


Fig. 10. V_1 and V_2 Lyapunov functions evolution when the safety mode is used

box-pushing," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 3, 1995, pp. 556–561.

[12] M. Wang and J. N. Liua, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. pp.625–643, 2008.

[13] P. Pirjanian, "Multiple objective behavior-based control," *Journal of Robotics and Autonomous Systems*, vol. 31, no. 1, pp. pp.53–60, April 2000.

[14] R. C. Arkin, "Motor schema-based mobile robot navigation," *International Journal of Robotics Research*, vol. 8, no. 4, pp. pp.92–112, 1989.

[15] M. Zefran and J. W. Burdick, "Design of switching controllers for systems with changing dynamics," in *IEEE Conference on Decision and Control CDC'98, Tampa, FL*, 1998, pp. 2113–2118.

[16] M. S. Branicky, "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transaction on Automatic Control*, vol. 43(4), pp. 475–482, 1998.

[17] J. Minguez, F. Lamiraux, and J.-P. Laumond, *Handbook of Robotics*, 2008, ch. Motion Planning and Obstacle Avoidance, pp. pp.827–852.

[18] H. K. Khalil, *Frequency domain analysis of feedback systems*, P. Hall, Ed. Nonlinear Systems: Chapter7, 3 edition, 2002.

[19] D.-H. Kim and J.-H. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robotics and Autonomous Systems*, vol. 42(1), pp. 17–30, 2003.

[20] L. Adouane, "Orbital obstacle avoidance algorithm for reliable and online mobile robot navigation," in *9th Conference on Autonomous Robot Systems and Competitions*, Portugal, May 2009.

[21] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier, "The cycab: a car-like robot navigating autonomously and safely among pedestrians," *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 51–68, 2005.

[22] M. Yerry and M. Shephard, "A modified quadtree approach to finite element mesh generation," *Computer, Graphics and Applications*, 1983.

[23] J.-P. Laumond, *La robotique mobile*. Hermès, 2001.

[24] M. S. Jie, J. H. Baek, Y. S. Hong, and K. W. Lee, "Real time obstacle avoidance for mobile robot using limit-cycle and vector field method," *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 866–873, October 2006.

This page is left blank intentionally

Modular Scalable Architecture for the Navigation of the ATLAS Autonomous Robots

M. Oliveira, P. Stein, J. Almeida, V. Santos, *Member, IEEE*

***Abstract* — This paper describes a solution to integrate disparate devices, both for perception and actuation, distributed amid distinct processing entities. Although the philosophy may be applied to many systems and machines, emphasis will be made on autonomous mobile robots' perception, actuation and intercommunication abilities. The solution uses inter-process communication (IPC) and encapsulation of messages in standard forms in the same trend as the CARMEN framework, where inspiration was gotten from. The modular architecture derived thereof allows to continuously increase the system complexity without changing whatever was previously implemented. Besides this scalable nature, the resulting architecture represents a unified approach that makes it hardware-independent where each machine simply relies on small specific modules. The IPC and CARMEN insights have been successfully adapted to two different robots within only one development project. The implementation will focus particularly on image efficient transfer among processes for real-time autonomous navigation.**

I. Introduction

AUTONOMOUS robots tend to become more and more complex both in hardware and software when the challenges raise and performance is paramount. The increasing affordability of more advanced sensors and devices induces researchers to include them on their mobile robots for more robust perception and navigation abilities. However, and due also to the disparity of standards and protocols, adding up off-the-shelf equipment along with custom designed boards or devices is not always a straightforward task. Moreover, keeping the pace in software development when there are changes in the team of programmers is often a nuisance for project managers.

To address these problems, the CARMEN (Carnegie Mellon Robot Navigation Toolkit) framework [1] appears as a tempting choice. It is a collection of modular software for mobile robots, and provides several useful functions for development of new modules and also for information exchanging amongst them. The communication between modules relies on the Inter Process Communication (IPC), developed by Reid Simmons [2] to be a flexible and efficient message exchanger; the potentialities of this IPC system are well assessed by its usage in paradigmatic projects related to NASA or the well known DARPA Challenge [6] [7] [8].

This work details the experience of adapting this new architecture in two competitions robots, named ATLAS MV and ATLAS 2008 [3], shown in Figure 1.

In the remainder of the paper, the next chapter will detail the installed hardware in both ATLAS robots. Chapter III describes the previous installed software architecture, its drawbacks, and how the new proposed architecture overcomes them. Following that, different techniques of information exchange between modules are discussed, emphasizing the development of a new technique to overcome IPC transfer limitations. Finally, the currently developed modules and message structures are presented, followed by the conclusion and future steps.

Manuscript received February 15 2009. All authors are with the Department of Mechanical Engineering, TEMA, University of Aveiro, 3810 Aveiro, Portugal, (e-mails: {mriem, *procopio, almeida.j, vitor}* *@ua.pt)*

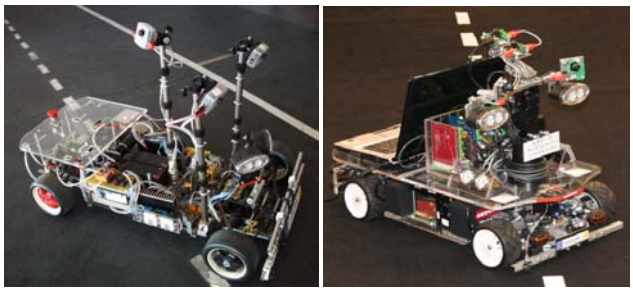


Figure 1 – Atlas 2008 (left) and the Atlas MV (right) robots.

II. Hardware Involved

Both ATLAS robots have similar basic hardware interface like cameras, traction motor, steer motor and digital input/output. The ATLAS-MV, which is a redesign of its ancestor ATLAS-2008, also incorporates some innovative hardware. One of them is a Pan and Tilt Unit (PTU) that it is used to change the cameras orientation providing active perception capabilities. The system also accounts for a Laser Range Finder with an accuracy of a few centimeters and a range of up to 30 meters.

The cameras are connected with the computer using a FireWire (IEEE 1394) interface and Direct Memory Access (DMA), which allow them to read or write to memory independently of the main computer processor. The ATLAS 2008 uses two cameras for navigation plus one for traffic lights recognition, while the new ATLAS MV uses four cameras mounted atop the PTU, and are employed in several tasks involving navigation and obstacles recognition. For the interface with the traction and steer motors, a microcontroller is used and connects to the computer using a RS232 link. The digital input/outputs are also interfaced using a microcontroller and a RS232 connection to the computer. The digital outputs are responsible for the brake and lights activation. Digital inputs, in other hand, receive readings from digital auxiliary sensors. This microcontroller board is projected in a way that it can easily incorporate more and distinct sensors. Both the PTU and the Laser Range Finder have their own controllers incorporated and are connected to the computer using an USB port.

III. Previous Vs new architecture

The previous software architecture installed on the robots has achieved successful results, but the code was based on a single large loop that would go through all hardware inputs/outputs sequentially, changing variable states that would affect the chosen behavior for the robot. This conveys a series of problems since small changes affect the whole code and pose difficulties in interfacing with new hardware. As the program was sequential, important events could be missed if the processor was taking too long to complete the loop, and if a "bug" or a deadlock happened, the whole program could hang.

To overcome these limitations, a modular architecture based on the CARMEN toolkit [1] has been devised. The idea is to split the old code into several modules, where each part is responsible for a small number of tasks. Information is exchanged between modules using standard messages, so new processing techniques or hardware could be easily incorporated. The new system architecture relies on separate modules, i.e., computer processes that run in parallel. These modules were divided from the previous architecture bearing in mind that each would contain a simple task. Each module processes the information received and outputs the result. Hence, the way information must travel from one module to the other is a critical issue. Communicating with the sensors often requires constant monitoring by the process running on the computer. This new architecture uses small, dedicated modules that handle hardware and communicate with other parallel modules via IPC. The modular architecture is also more robust, because redundant parallel modules may compensate fails of others. Also, because the IPC [2] based communication is performed through TCP/IP connections, messages can be easily exchanged between processes running in different machines. Hence, the entire program may be distributed in several computers, increasing the computational power of the robots, if so required. This is usually an important issue when running real time vision-based algorithms.

IV. Information exchange

The information exchanged among modules must be classified so that a module receives only what it actually requires, and not everything else. This is accomplished by encapsulating information into messages. A module interested in some particular information can then ask to receive a specific message. On the other hand, a module that produces some specific output can constantly send a message containing a certain type of information.

A. Publish/Subscribe Method

Message exchanging can be done in several ways. The simplest method is called publish/subscribe (Figure 2).

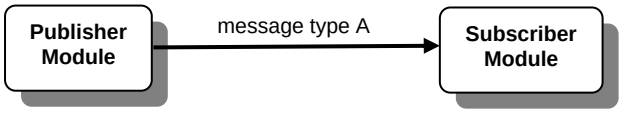


Figure 2 – A simple publish subscribe setup.

In this scenario, a publisher module generates information that is packed into a message of a given type, let's say, "A". Every other module interested in the information contained in message type "A" should subscribe to it. When that information is available, the publisher module publishes a message type "A" that will be redirected by IPC to all modules that have previously subscribed to it. One disadvantage is that the publisher module requires that the subscribing module accepts all messages of type "A", even if it does not require them all. This may become critical if the publisher's cycle time is shorter than the one of the subscriber. Figure 3 shows a message exchange where this phenomenon takes place.

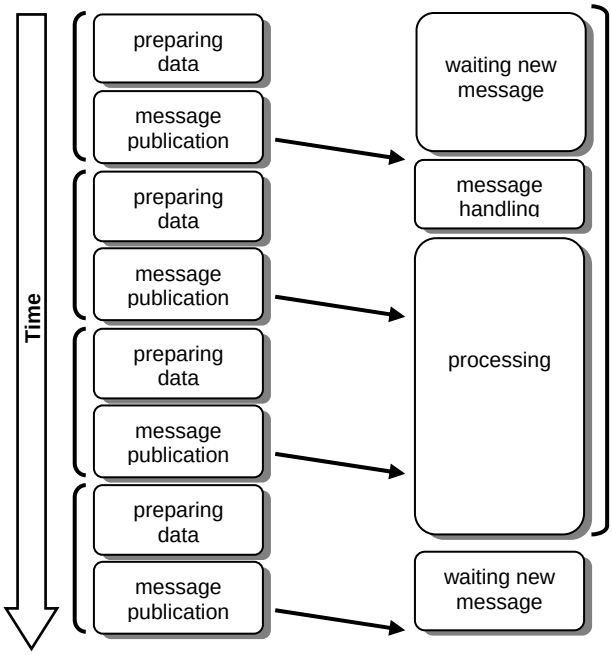


Figure 3 – An example of a time sequence (vertical axis) of a publish/subscribe message exchange. The publishing module (left) has a faster cycle time (represented by brackets) than the subscriber module (right). Because of this, the subscriber module receives 2 messages while still processing the first one.

In this case, there is going to be a growing queue of messages that, at some point, will overflow and cause the IPC central module to crash, as observed in practice, especially for large messages (hundreds of kilobytes). Because of this, the publish/subscribe message exchange methodology should be considered only if the subscriber module is faster than the publisher, or if the handling of the message reception performed by the subscriber is a very fast routine. For this reason, it is not advisable to send large messages containing images or laser scans using this specific methodology.

B. Query/Respond

As stated, the publish/subscribe method is appropriate for exchanging small messages or for fast processing modules. However, as seen in chapter II, the robots have several cameras/lasers onboard. Cameras installed on ATLAS robots usually produce 320×240, 3 channels RGB images at a rate of about 30 FPS. This means that a module performing image acquisition generates approximately 7 Megabytes of information every second (320 × 240 pixels × 3 channels × 30 FPS ≈ 7 MB). Publishing such a large amount of information caused IPC

to overload during our tests. This occurred especially when the subscriber module was not able to handle the information flow. Furthermore, if the subscriber modules cannot process the information fast enough, the messages are sent but not processed. These limitations can be circumvented using a second methodology available in IPC: the module that can send the information, called in this case the *server module*, does so only when asked by the receiving module, here called the *query module*. Message exchange rate is set by the *query module*, instead of being arbitrarily defined by the server.

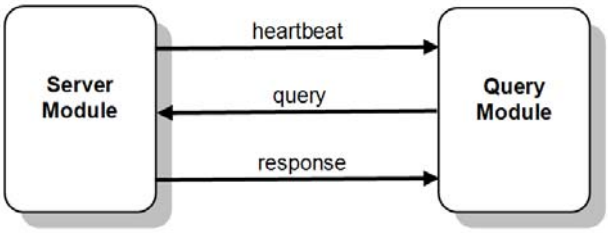


Figure 4 - A query respond with heartbeat setup.

To support this method of exchange, three messages are defined: the *heartbeat*, the *query* and the *response* messages (Figure 4). The heartbeat message indicates that the server has new information available and is able to send it; it is a small message, a mere notification to whoever is interested in the new information generated that a new one is available. Because of this, heartbeat messages are broadcasted by the server module, using the publish/subscribe technique, thus allowing several modules to be informed. The image acquisition module would publish thirty heartbeat messages every second.

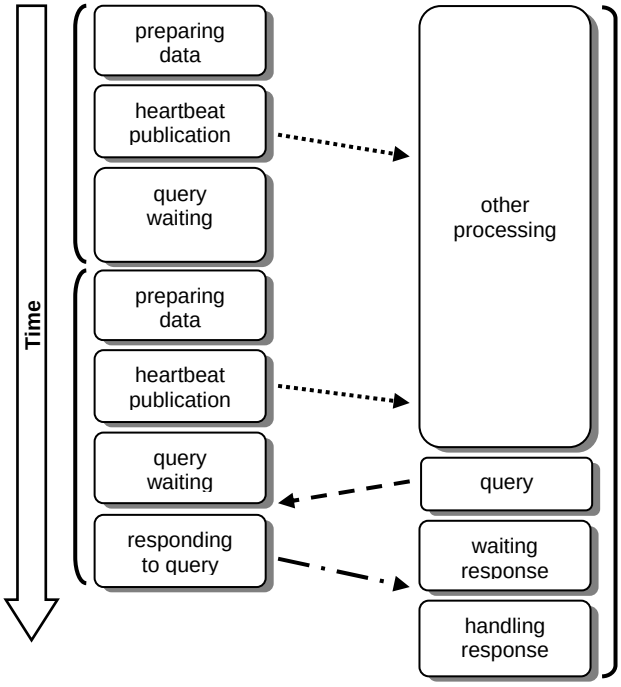


Figure 5 – A query response time flow. Heartbeat messages (dotted line) are published whenever the server module (on the left) generates new information. The query module (on the right) queries the information (dashed line) only when it actually requires it.

The query message is also a small one, and is sent by the query module to the server in a peer to peer communication. After sending the query message, the query module waits for the response. The response message can be large (could contain an image, for example), and is sent by the server to the query module only in reply to a query. The response is also peer to peer.

Though more complex, this setup is particularly useful for transmitting large messages since that transmission only occurs when the query module actually needs the information, reducing the message traffic. If the query module is faster than the server module, heartbeat messages ensure that no query is done unless new information is available on the server's side. The flow of messages is shown in Figure 5. Heartbeat and queries are control messages that synchronize both modules so the information is exchanged only when actually needed. In this case, the complexity increase is compensated by avoiding sending unneeded large messages. Because of this, this setup should only be employed when the messages to be exchanged are large.

C. *Shared Memory Query/Respond Method*

The disadvantage of the query/respond method is that because messages are queried by a specific module, the server responds only to this one, i.e. it is a peer to peer communication. If the server module is meant to send large amounts of information to several query modules, the message traffic would increase as many times as the number of receiving modules. In the worst case scenario, if the number of query modules is large enough, the server module may not be able to respond to them all in the time allocated to listen to the queries.

To solve this problem, a new method has been devised, where the server module writes the message containing the response to a shared memory address. This technique takes advantages of IPC marshal/unmarshal functionalities. Marshaling is the process of transforming the message into a configurable easily reversible linear byte array. Messages are defined as C language structures, but before being sent must previously be transformed, i.e., marshaled, into byte arrays. To be marshaled, the format of the message structure is first defined. Figure 6 shows an example of a message.

```
typedef struct
{
  int var1;
  char var2;
  int var3[2];
}type_msgA;

#define msgA format "{int, char, <int:2>}"
```

Figure 6 - An example data structure in C and its format definition. This could be a message to be exchanged between processes.

In the case of Figure 6, marshaling would take the format of the structure (*msgA_format*) as one integer followed by a char followed by an array of 2 integers. The modules use IPC to marshal every message structure when sending, and to unmarshal when receiving.

In this particular information exchange setup, the server module allocates a shared memory segment with the size of the message, and then stores the message directly onto the shared memory. After this operation, a heartbeat message is published indicating that new information is available.

The query module attaches itself to the memory segment and unmarshals the information to a variable of

type *type_msgA*. The attach operation requires the shared memory address (Figure 7). For this purpose a new message is defined containing shared memory information, its address and size. This message is queried to the server during the query module's initialization procedures. Afterwards, whenever a heartbeat is received, the query module just unmarshals the information from the shared memory and gets a copy of the message.

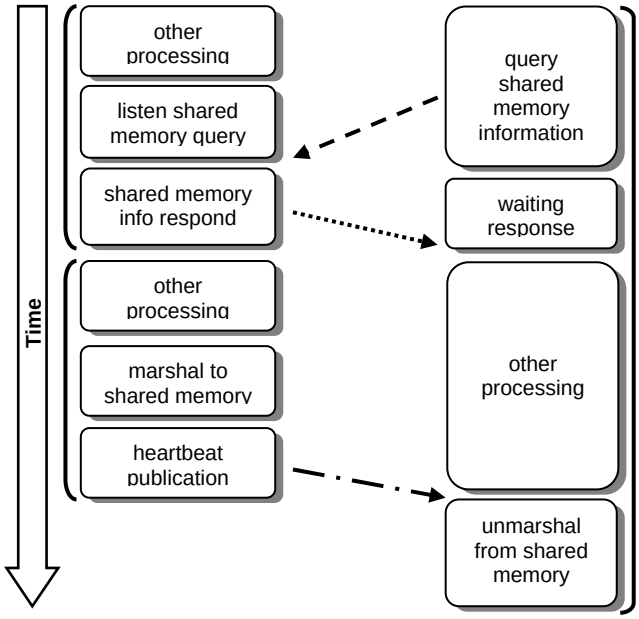


Figure 7 – A query response time flow using shared memory. The query module (on the right) queries for the shared memory id (dashed arrow). The server module (on the left) responds with the information (dotted arrow). The query process then attaches itself to the shared memory and unmarshals the data whenever a heartbeat is received.

A limitation of this method is that, because it is not based on TCP/IP, it does not work when the processes run on separate machines. Another glitch is that on large messages, there is some possibility that the query module may be still reading part of the message while the server is writing. This may lead to reading messages that are actually a combination of two different messages. In our particular case, this is not an issue, although if required, it could nevertheless be prevented by using semaphores. The bottom line is that the method is very fast and addresses the problems discussed deriving from the usage of multiple query modules.

V. Proposed Modules for ATLAS

Several modules have been developed in the process of migration from the old architecture to the new, modular one. The modules can be roughly divided into three categories: hardware interface, features extractors and planning/decision. The first is responsible for every interaction with the hardware: data acquisition and motors command. The feature extractors will process the acquired data, where each module is responsible for one type of feature such as obstacle or lane detection. The last category is dedicated to the reasoning capabilities of the robots. It makes use of all the detected features and, based on the context, will decide and plan the robot's behavior. In this paper, only the first category of hardware communication modules will be described.

A. *Cameras acquisition module*

The cameras acquisition module (Figure 8) is intended to acquire images from the Firewire cameras installed on the robots. All message exchange methodologies have been developed: publish/subscribe query/response and shared memory.

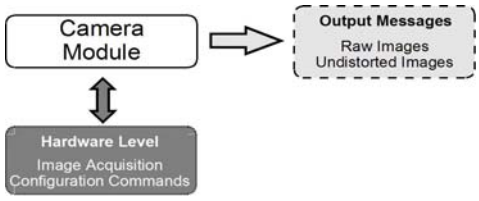


Figure 8 – The cameras acquisition module.

The published messages have the format of Figure 9.

```
typedef struct
{
  int width;                                /*width in pixels*/
  int height;                              /*height in pixels*/
  int bytes_per_pixel;                           /* 3 (RGB) */
  int image_size;                                 /*w*h*3*/
  char *image;                        /*pointer to image data*/
  double timestamp;                    /*timestamp of message*/
  char *host;                                  /*used by IPC*/
} lar_ubcamera_image_message;
```

Figure 9 – The image message format structure.

This module also enables the real time setting of the cameras parameters. Brightness, saturation, white balance, shutter and others can be set to a particular value. The module is also capable of handing out distortion corrected images taken from wide angle lens cameras, if a prior chessboard calibration has been performed.

B. *Laser acquisition module*

The laser acquisition module (Figure 10) acquires laser data from the laser and sends the information to other modules.

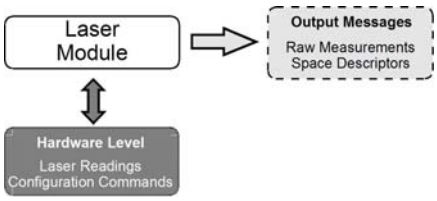


Figure 10 – The laser acquisition module.

The message structure of the laser module is shown in Figure 11. Several laser parameters can be set at startup: angular resolution, start/end scan angle, among others.

```
typedef struct
{
  carmen_laser_laser_type_t laser_type;       /*laser model*/
  double start_angle;              /*angle of the first beam*/
  double fov;                    /*field of view of the laser*/
  double angular_resolution;                  /*up to 0.25*/
  double maximum_range;                        /*30 meters*/
  double accuracy;                            /*0.1 meters*/
  carmen_laser_remission_type_t remission_mode;  /*not used*/
} lar_laser_laser_config_t;

typedef struct
{
  int id;                                        /*laser id*/
  carmen_laser_laser_config_t config;               /*above*/
  int num_readings;             /*number of range values sent*/
  float *range;                        /*laser range values*/
  int num_remissions;            /*number of remission values*/
  float *remission;                 /*remission laser values*/
  double timestamp;                  /*timestamp of message*/
  char *host;                                  /*used by IPC*/
} lar_laser_laser_message;
```

Figure 11 – The laser message format. Structure proposed by CARMEN.

It is also possible to use any one of the three methods described in chapter IV to exchange the laser messages.

C. *PTU module*

The PTU module (Figure 12) is capable of commanding the pan and tilt unit based on orders received from other modules. It can also inform other modules of the pan and tilt state, i.e., axis position, speed and acceleration.

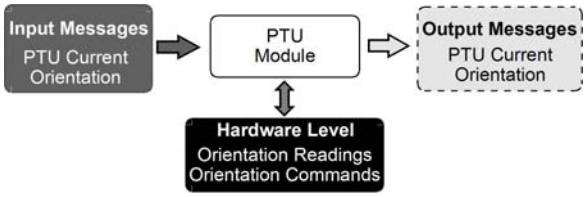


Figure 12 – The PTU control module.

Because the messages related to the PTU control/monitoring are small, only the publish/subscribe method has been implemented. If a module requires information on the state of PTU it subscribes to messages of type *lar_ptu_status_message*, defined in Figure 13.

```
typedef struct {double pan;double tilt;}TYPE_pantil;

typedef struct {
  char *dev;                          /*serial port device*/
  int devnum;                  /*serial port device number*/
  int baudrate;                   /*communications baurate*/
  struct {
  TYPE_pantil position;                        /*in radians*/
  TYPE_pantil speed;                /*in radians per sec*/
  }current;                   /*current position and speed*/
  struct {
  TYPE_pantil position;                        /*in radians*/
  TYPE_pantil speed;                /*in radians per sec*/
  }desired;                   /*desired position and speed*/
typedef struct{
  char purevelocity;                /*is pure velocity set*/
  char imediatepositionexecution;                /*IPE flag*/
  }flg;                                /*ptu state flags*/
  double timestamp;                 /*timestamp of message*/
  char *host;                                  /*used by IPC*/
}lar ptu status message;
```

Figure 13 – The message format to get information on the PTU's state.

If, on the other hand, a module wishes to command the PTU positioning or speed, it should publish a message of the type *lar_ptu_command_message* (Figure 14). During startup, the PTU control module subscribes to this message type. It is actually possible to have different modules competing for the PTU's command sending the same message type. The PTU module will just receive all the messages and execute them sequentially.

```
typedef struct {
  TYPE_pantil position;                  /*ordered position*/
  TYPE_pantil speed;                       /*ordered speed*/
  int usepurevelocity;                  /*use pure velocity*/
  double timestamp;                 /*timestamp of message*/
  char *host;                                  /*used by IPC*/
}lar_ptu_command_message;
```

Figure 14 – The message format used to give orders to the PTU module.

D. *Robot Base module*

This module is responsible for interfacing with the traction and steer motors. Also digital input readings and digital output commands are performed here (Figure 15).

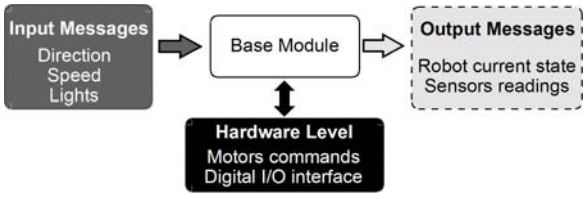


Figure 15 – The base module.

The Robot base module receives the message *lar_atlas_dir_and_speed_message* (Figure 16), and translates that information to the specific robot hardware. In this way, different robots will have different *base modules*, but will be able to receive the same command messages. This module also publishes a message containing information about the robot current speed, steer angle, lights state and digital inputs readings, by means of a *lar_atlas_status_message*.

```
typedef struct {
  double dir;                                 /*steer direction*/
  int speed;                                        /*speed*/
  double timestamp;                        /*timestamp of message*/
  char *host;                                       /*used by IPC*/
}lar_atlas_dir_and_speed_message;
```

Figure 16 – Definition of the message for commanding the base.

E. *Teleoperation module*

This module allows the remote command of the hardware related features of the robot using a standard gamepad. It does not have any specific publication/subscription routines. Instead, it simply makes use of the routines implemented by each of the modules it wants to control.

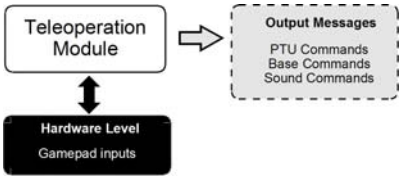


Figure 17 - Definition of the message for commanding the base.

It has two distinct operation modes: a command mode and a tutorial mode. In the first, it can command the speed, steering, lights and the PTU unit, publishing a *lar_atlas_dir_and_speed_message* (Figure 16) and a *ptu_command_message* (Figure 14). In tutorial mode, pressing the buttons or axes provides an audio tutorial. This is accomplished by instructing the sound player module to reproduce the audio that matches to the button pressed.

F. *Sound Player module*

The Sound Player module (Figure 18) is capable of generating audio output. For synchronization purposes, it can also inform other modules if it is busy playing a sound.

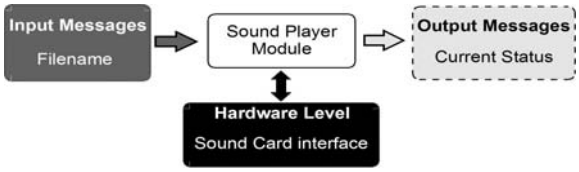


Figure 18 – Sound player Module.

Any module can require a sound message to be played by publishing a *lar_soundplayer_message*. This message is defined in Figure 19, with the identification of the media to be played, given as a file name (string) or as a numeric identification.

```
typedef struct {
  int filenumber;                                 /*numeric id*/
  char *filename;                      /*file name to be played*/
  int mode;                  /*mode (id by number or by name)*/
  double timestamp;                          /*timestamp of image*/
  char *host;                                       /*used by IPC*/
}lar_soundplayer_message;
```

Figure 19 - Definition of the message for commanding the sound player.

Upon receiving the command, the sound player makes

use of *Libao* library [5] functions to reproduce it. The sound player module indicates its status (busy or available) by publishing a *lar_soundplayer_status_message*, defined in Figure 20. This module allows improved user/robot interactivity and also provides debug facilities.

```
typedef struct {
  int status;                /*Status of the sound generator*/
  double timestamp;                    /*timestamp of image*/
  char *host;                                 /*used by IPC*/
}lar_soundplayer_status_message;
```

Figure 20 – The message format to get the status of the sound player.

G. *Sensor Fusion Module*

The sensor fusion module (Figure 21) is the responsible for merging the information coming from the cameras and the laser sensors installed on the robots. It creates a common reference representation of the measurements taken, whether they are images or range scans. Because cameras are mounted on the PTU, cameras' positions are a function of the PTU orientation.

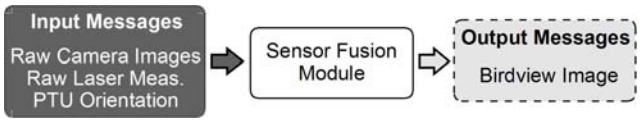


Figure 21 – The sensor fusion module's input and output messages.

The module can fuse several images captured from multiple cameras along with laser information, generating enhanced images of the road, in a birdview perspective. This algorithm is described in detail in [4]. Figure 22 shows a birdview of the road obtained by merging the images of two different cameras.

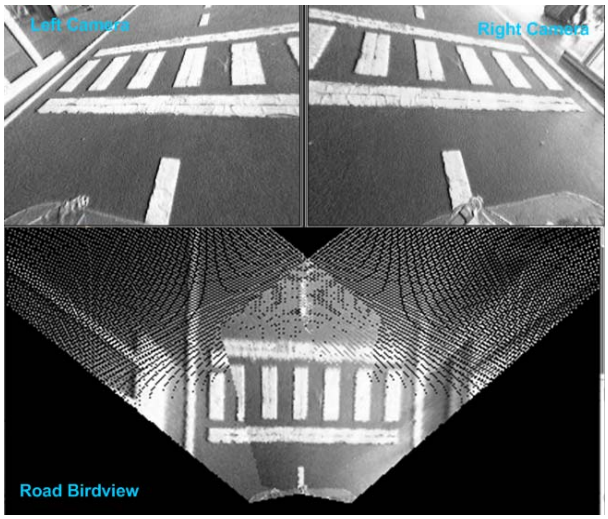


Figure 22 – The images taken from the left and right camera and the birdview of the road obtained by merging both images.

During initialization, this module reads some parameters that define a box of interest. This rectangular region, viewed in Figure 22, is the area where the robot is interested on receiving sensory data. If, for some reason, one particular sensor harvests information of an uninteresting area for the robot, i.e., an area outside the box of interest, this module ensures that this information, being regarded as unimportant, is clipped away from the merged information. The ultimate goal of this task is to find a common representation for a multitude of sensor types and/or configurations. This ensures that, no matter the specific sensorial setup of a given robot, it is reshaped into a common reference. The advantage here is that subsequent modules (like feature extractors, road detectors, obstacle detectors, etc) can rely on a constant, predefined representation of the data and so may work without need for reconfiguration, regardless of the current sensorial setup, regardless of the robot.

VI. Conclusions and Final Remarks

The paper described a successful adaptation of the CMU CARMEN and IPC approaches to two distinct autonomous robots. The resulting architecture has proven fully scalable since any modules can be added or suppressed without compromising the global operability. The encapsulation of information in predefined messages, by dividing the code in small task oriented modules and experimenting different forms of information exchange, has been a central issue.

The IPC framework available at the CARMEN community proved reliable except for some limitations regarding the transmission of large data sets at a high frequency. This was overcome with the development of a mixed method that involved the well known shared memory intercommunication together with IPC structures and functions, resulting in a seamless integration with the already developed modules, and can be seen as an extension of them.

The modules presented are all hardware-related (with exception of the Sensor Fusion Module, which is a preprocessing module), as this was the first step in the effort of migrating to the new architecture, and as so it had to be carefully planned to serve as the foundation for the further development of higher level modules as the features extractors and decision/planning modules.

A final important outcome of this work is the strongly organized software and functional architecture on the ATLAS robots, allowing an unlimited team of developers to cooperate together. This was indeed the major breakthrough, and represents a significant step towards more demanding projects on complex perception and autonomous navigation of advanced machines.

VII. References

[1] CARMEN, Carnegie Mellon Navigation Toolkit, found at http://carmen.sourceforge.net/ on February 2009.

[2] R. Simmons, and D. Apfelbaum, " A task description language for robot control". In 1998 Proceedings of the Conference on Intelligent Robots and Systems (IROS), Victoria, CA.

[3] M. Oliveira, V. Santos, A Vision-based Solution for the Navigation of a Mobile Robot in a Road-like Environment, Robótica, nº69, 2007 p.8 (ISSN: 0874-9019)

[4] Oliveira M., Santos V., Multi-Camera Active Perception System with Variable Image Perspective for Mobile Robot Navigation, 8th Conference on Mobile Robots and Competitions, Portuguese Robotics Open, Aveiro, April 2008.

[5] Libao, Open Source Audio Output Library, found at http://www.xiph.org/ao/ on January 2009.

[6] DARPA Grand Challenge, http://www.darpa.mil/grandchallenge/, February 2009.

[7] M. Montemerlo, *et al.*, "Junior: The Stanford entry in the Urban Challenge," *Journal of Field Robotics*, vol. 25, 2008, pp. 569-597.

[8] C. Urmson, *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, 2008, pp. 425-466.

Preliminary Results on Robust Global Localization: Fault Tolerance and Robustness Test on Probabilistic Shaping

Luca Carlone and Basilio Bona, *Member, IEEE*

***Abstract*— Interactive human-in-the-loop tasks in service or domestic robotics raise concerns about safety and dependability of localization systems. As physical redundancy is no longer suitable for low-cost-low-power applications, decisional mechanisms are required in order to assure reliable localization. We investigate the case of beacon-based positioning with particular emphasis on fault tolerance and robustness to environmental perturbations. We provide a model for range-only localization, called *Probabilistic Shaping*, that includes uncertainty on beacons' position, noise on measurements and information from navigation sensors. Our approach is based on a closed-form probability distribution, called *Radial Gaussian*. We prove the robustness and the fault tolerance of our localization model in a simulation scenario, analyzing the response of the system in terms of accuracy and resilience. All relevant information on simulations are given in order to provide a virtual benchmark for dependable localization.**

I. Introduction

POSITION estimation is a crucial task for autonomous mobile systems. In service and domestic robotics, in particular, the knowledge of robot's position is necessary to accomplish complex tasks in uncertain human-in-the-loop scenarios. The strong interaction with humans raises concerns about safety and dependability, requiring a deep insight of fault hypothesis and predicted response of localization system to adverse situations. We investigate the case of beacon-based localization in indoor environment. Many GPS-less localization problems are solved within an infrastructure of low-cost radio beacons inferring range information from *received signal strength* [1]-[2]. A widespread approach is to associate geometric lateration with Kalman Filter or model data through other probabilistic methods (Particle Filters, Grid localization etc.). Different algorithms, usually called *non-linear sliding batch*, use Gauss-Newton or Levenberg-Marquardt optimization to find the optimal path given all the data collected. The complexity of batch methods makes them unsuitable for on-line computation [3]. Few authors studied localization with radio frequency devices addressing the problem of sensor failures. In [4] a comparison of Monte Carlo methods and Kalman Filter can be found. Particular emphasis is put on response of probabilistic filters to extensive sensor silence. This work shows how traditional methods though being accurate in nominal situations, exhibit poor performances in adverse scenarios. Linearization errors in EKF framework lead to inconsistency as estimate diverges from true pose [5]. Similar problems could occur using Particle Filters when no particles are in the proximity of the true state (*particle deprivation problem* [6]).

In this context we propose a probabilistic model for beacon localization, called *Probabilistic Shaping*, and we test the robustness and fault tolerance of the algorithm applied to a range-based positioning system. Our simulations are performed in a realistic scenario that can be a useful benchmark for performance evaluation and comparison. In the following section we introduce a general framework on robustness and fault tolerance in dependable systems, applying it to localization. In Section III we present our method, describing underlying assumptions and practical implementation. Then in Section IV we show the results of simulation in a realistic scenario. Conclusion are drawn in Section V.

Manuscript received February 15, 2009. This work was supported in part by Regione Piemonte under MACP4log Grant (RU/02/26).

L. Carlone is with CSPP, Laboratorio di Meccatronica, Politecnico di Torino, Torino, 10129, Italy (phone: +39-338-9072246; e-mail: luca.carlone@polito.it).

B. Bona is with Dipartimento di Automatica e Informatica, Politecnico di Torino, Torino, 10129, Italy (e-mail: basilio.bona@polito.it).

II. Robustness and Fault Tolerance

Robustness and fault tolerance characterize the resilience of a system in delivering services despite adverse situations [7]. Robustness can be considered as a superset of fault tolerance but usually the following distinction is applied [8]:

--*Robustness* is the capability of delivering the correct service when adverse environmental issues raise;

--*Fault Tolerance* is the capability of delivering correct service despite faults affecting system resources.

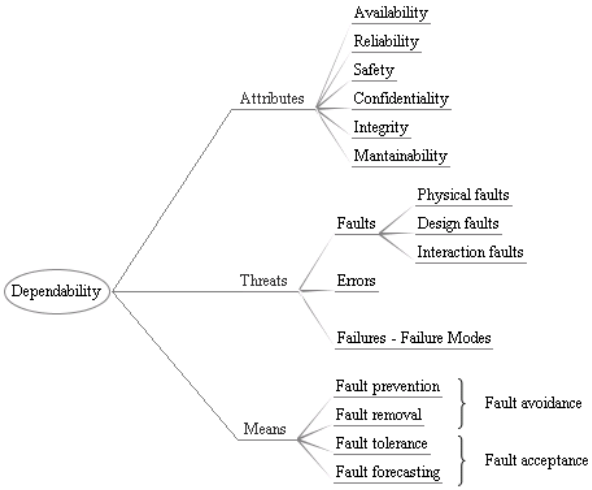


Fig. 1. Dependability tree.

In our case study the service required is localization: one or more robots, in general *localization targets*, estimate their own position using the information from on-

board portable devices, later called *readers*, able to communicate with *tags* dispersed in the environment in known positions. The reader acquire range-only measurements from tags within the reading range. No prior information on robot position are available: the robot can only measure reader-tag distances and acquire motion information from its navigation sensors (IMU, gyros, odometry). The core of our localization system is the probabilistic model that performs estimation from data collected. Robustness and Fault tolerance are connected with both the infrastructure and the algorithm. In order to formulate our fault hypothesis we use a basic framework on dependability for autonomous systems, proposed in [8] and summarized in Fig. 1.

Neglecting the attributes of dependability we focus on threats to localization in order to evaluate the means for providing correct position estimation despite faults and errors. We identified the following threats emerged from the study of radio frequency technology:

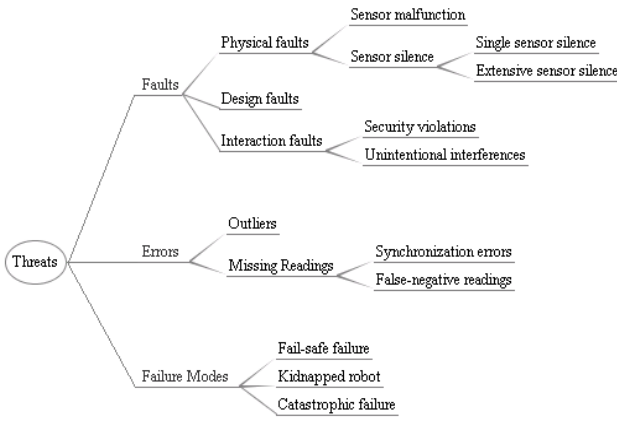


Fig. 2. Threats to dependable beacon-based localization.

Fault hypothesis can be deduced from [4] and [9]. They can be divided in:

--*Physical faults*: are faults in the physical infrastructure. *Sensor malfunction* is characterized by an alteration of signal strength or by synchronization errors. *Sensor silence* is the temporary or permanent absence of information from one or more tags in the reading range. We categorize the former as *single sensor silence.* The latter is already known as *extensive sensor silence* [4]. In this case during a long period of time no range measurements are received from the beacons. Notice that the extensive sensor silence can be connected with multiple tag faults or with reader's fault. The probabilistic model we propose is able to detect physical faults;

--*Design faults*: are faults in the implementation of the system itself. A common solution to design faults is redundant and diversified design;

--*Interaction faults*: are faults due to accidental (*unintentional interferences*) or malicious interventions (*security violations*). Radio frequency technology already solves the problem of external undesired interferences applying authentication techniques based on cryptography.

Errors can cause failures and can be considered as symptoms of a fault in the system. We consider two main error category:

--*Missing readings*: they comprehend *synchronization errors* and *false-negative readings*. The formers are due to anti-collision protocols or to delay of tag's response. As consequence of synchronization errors distance measurements are not acquired in the same instant of time. If the robot is moving, measures are referred to different positions and geometric lateration performs worse. F*alse-negative readings* occur when a tag in the reading range is not detected by the reader. They are typical of RFID technology but can be symptoms of sensor faults.

--*Outliers* [10]: they are characterized by a large error in distance measurements. In some cases a tag can be read also outside the reading range (*false-positive reading*). Signal strength fluctuations are peculiar of radio transmission but the category includes also anomalies due to sensor malfunction. High corrupted data can be rejected [11] or its effect can be reduced;

As consequence of faults a system is not able to provide the service required. In particular it is possible to distinguish the following failure modes:

--*Fail-safe failure*: localization errors remain upper bounded within acceptable values, depending on the particular application. Common sense suggests that mean errors should be comparable with the size of the robot whereas a reasonable value for upper-bound of the error is within the same order of magnitude. Most common mechanisms for recovery in fault tolerant systems are *rollback error recovery*, *rollforward error recovery* and *error compensation.* In the first case, after error detection, the system returns in a previous stable state. In *rollforward error recovery* the system is lead to a new acceptable state, whereas *error compensation* is based on anomalies rejection. In the last case redundancy is required in order to detect and compensate faults;

--*Kidnapped robot*: as consequence of a fault or malicious attack the robot cannot perform localization. As new readings are acquired the target has to be able to perform a correct estimation of its position. Notice that as consequence of the external attack, or after a period of *extensive sensor silence*, the robot position could change abruptly from last correct estimate;

--*Catastrophic failure*: as consequence of a fault in the system the estimate diverges from the real position. This is the worst case since without external intervention no correct estimate can be provided. Moreover, if the error remains undetected, the robot takes decisions in a *misguided optimism* [7].

A *fault tolerant* localization algorithm allows position estimation avoiding catastrophic failures despite sensor faults. Moreover a system is *robust* if it is also able to localize the target in adverse environmental situations. In beacon-based localization the uncertainty of the environment is connected with uncertain position of tags. We will prove in simulation that our localization model is fault tolerant and robust to tag position errors. We do not consider physical redundancy as solution to faults but a decisional mechanism is introduced in order to evaluate the accuracy of estimation as metric for error detection. Our localization system is able to perform fault detection and avoidance without any further complexity or cost. After presenting our model we will

provide a simulation scenario for adverse simulation tests. All relevant information on simulations are given in order to provide a virtual benchmark for dependable localization.

III. Probabilistic Shaping

A. Model Overview

The probabilistic model that we propose is based on a closed-form probability distribution. This function is called *Radial Gaussian* and represents the belief of the robot after a range measurement from a beacon. If we store measurements in a buffer it is possible to obtain a robust multilateration algorithm. Probabilistic Shaping can be summarized in the following phases:

1) *Measurement update*: when a range measurement is acquired it is stored in a FIFO buffer. The buffer also contains the position of the tag corresponding to the measure and the variance on measurement;
2) *Prediction*: while the robot is moving the system updates data in the buffer. Tag Positions corresponding to previous measurements are changed in agreement with information from navigation sensors. Also the correspondent variances increase since motion introduces further uncertainty;
3) *Estimation*: data collected is included into the joint distribution and estimation is performed using *circular search*. Estimation can be performed asynchronously and it is independent from previous steps;
4) *Judgement*: the result of estimation phase is a best guess on robot position. Judgement phase provides a decisional mechanism for reliability. When an estimate is not reliable, localization is performed through dead-reckoning (*rollforward error recovery*).

B. Measurement update

Measuring the exact distance r_k from a known reference point x_k we can represent our belief as a circle centered on the beacon with radius equal to r_k. In a real case the noise on the range measurement influences our belief, extending the space of possible positions. Assuming that the uncertainty on range measurement can be modeled as gaussian, after the measurement k acquired at time t, the probability distribution representing our position belief in a general *n-dimensional* space is:

$$p\left(x_t \mid z_t^k\right) = \frac{1}{\eta} \cdot \exp\left(- \frac{\left(\left\| x_t - x_k \right\| - r_k\right)^2}{2\sigma_{k0}^2} \right) \quad (1)$$

where $\|\cdot\|$ is the *Euclidean norm*, x_k is the position of the beacon in the *n-dimensional* space, r_k is the measured distance, σ_{k0} is the standard deviation on measurement and η is a *normalization factor* that makes the integral of the distribution equal to 1. In [12] η is computed for the 2D and 3D case and some interesting properties of the function are proposed. Planar and three-dimensional Radial Gaussian are plotted in Fig. 3, in which color scale gives information about the probability corresponding to each point.

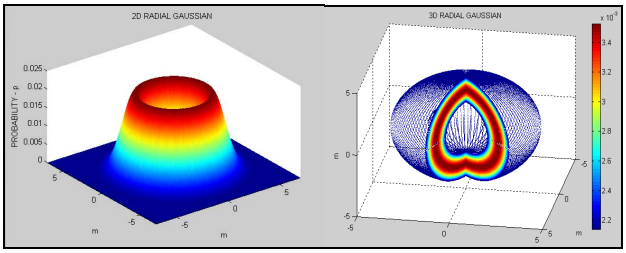


Fig. 3: Two-dimensional (left) and three-dimensional (right) Radial Gaussian. Color scale gives information on probability. In 3D case only points having distance inside the interval $[r_k\text{-}3\sigma_{r,k};\ r_k+3\sigma_{r,k}]$ are plotted. Cross section shows gaussian trend centered on the radius $r_k=3$ *m*.

If the position of the beacon is not exactly known the uncertainty on tag location contributes to increase the variance of the distribution. In this context we suppose that the belief on the position of landmarks can be modeled as Gaussian, with diagonal covariance matrix in the form $\Sigma=diag(\sigma_{tag}^2)$. The circular symmetry of the problem allows us to reduce our formulation to a mono-dimensional case and it is possible to demonstrate that the overall variance, that takes into account both uncertainty on beacon position and on measurement is:

$$\sigma_k^2 = \sigma_{k0}^2 + \sigma_{tag}^2 \quad (2)$$

Since each tag has a unique identifier no ambiguity occurs and after a measurement the belief of the robot can be parameterized, in a planar case, by four variables (x_k, y_k, r_k, σ_k). Storing this parameters in a FIFO buffer we can keep memory of measurement history. As explained in Section IV, the length of the buffer is a compromise between accuracy, correlation between measurements and delay of error response.

C. Prediction

At time t the robot acquire a measurement from a beacon and the localization algorithm stores the corresponding data, that represents $p(x_t|z_t^k)$. After a sampling period we need to actualize past belief introducing information about navigation sensors. In other terms we need to obtain $p(x_t|z_{t-1}^k,s_t)$ that is the state probability at present time, given past measurement z_{t-1}, after applying the command s_t. It is possible to compute $p(x_t|z_{t-1}^k,s_t)$, starting from previous belief $p(x_{t-1}|z_{t-1}^k)$, assuming a state transition probability. In our model we assume that the state transition probability is normally distributed, with covariance matrix $\Sigma=diag(\sigma_{\Delta x}^2(\mu_{\Delta x}))$, where $\mu_{\Delta x}$ and $\sigma_{\Delta x}$ are the mean value and the variance that parameterize mobile robot displacement at time t. As consequence, for each measurement in the buffer, the prediction phase can be computed using the *law of total probability*:

$$p\left(x_t \mid z_{t-1}^k, s_t\right) = \int p\left(x_t \mid x_{t-1}, z_{t-1}^k, s_t\right) \cdot p\left(x_{t-1} \mid z_{t-1}^k\right) dx_{t-1} \quad (3)$$

Applying (3) to a Radial Gaussian that represents $p(x_{t-1}|z_{t-1}^k)$ we obtain another Radial Gaussian distribution

with center $x_{k,t}=x_{k,t-1}+\mu_{\Delta x}$ and variance $\sigma_{k,t}^2=\sigma_{k,t-1}^2+\sigma_{\Delta x}^2$. This motion model is general and allows to decouple the problem from the particular robotic platform. As consequence it can be applied also to human, animal, or object localization. The only strict assumption is on the structure of covariance matrix Σ. On the other hand if the belief on robot position is distributed in a different manner a conservative covariance estimate can be used.

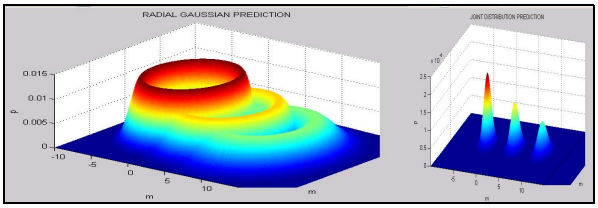


Fig. 4. Prediction phase. On the left a Radial Gaussian and its prediction 2 steps later is shown. On the right the joint probability of three Radial Gaussian is presented.

D. Estimation

After update and prediction phases the buffer contains a number of measurements equal to the buffer length N. Each row contains the information about measure z^k (here we omit the time index because after prediction phase the measures are actualized to present time). To compute the overall probability $p(x_t|z^{1:N})$ we use *Bayes theorem*:

$$p\left(x_t \mid z^{1:N}\right)=\eta_1 \cdot p\left(z^{1:N} \mid x_t\right)\cdot p\left(x_t\right) \tag{4}$$

Under the hypothesis of independence between the noise in each individual measurement, the overall probability $p(z^{1:k}|x_t)$ can be computed as the product of the individual likelihood:

$$p\left(z^{1:N} \mid x_t\right)=\prod_{k=1}^{N} p\left(z^k \mid x_t\right) \tag{5}$$

This assumption is strict, due to the correlation introduced by beacons position and prediction phase, but simulation shows that the method proposed is robust to these violations.

Our method performs global localization so we assume that $p(x_t)$, i.e. the belief without any measurement, is a uniform distribution in the state space. Also $p(z_t)$ *is* usually considered constant within the state space [6]. Therefore we included both $p(x_t)$ and $p(z_t)$ in the normalization factor. Substituting (5) in (4) and applying *Bayes theorem* we obtain:

$$p\left(x_t \mid z^{1:N}\right)=\eta_1 \cdot \prod_{k=1}^{N} p\left(z^k \mid x_t\right)\cdot p\left(x_t\right)=\eta_2 \cdot \prod_{k=1}^{N} p\left(z^k \mid x_t\right)=$$
$$=\eta_3 \cdot \prod_{k=1}^{N} p\left(x_t \mid z^k\right)\cdot p\left(z_t\right) \tag{6}$$

So we calculated the belief at time t from all the information included in the buffer, summarized in the second term of the following equation:

$$p\left(x_t \mid z^{1:N}\right)=\eta \cdot \prod_{k=1}^{N} p\left(x_t \mid z^k\right) \tag{7}$$

Obviously all the conditional probabilities calculated before underlie the dependence on the knowledge of the map (position of the tags in our case), but it is common in localization problems. Another interpretation of our method can be based on measurement likelihood, but we preferred previous passages for a better understanding of individual phases.

The closed-form product (7), shown in Fig. 5 in the case of $N=3$, contains all the information about the history of measurement we stored. The distribution can be intrinsically multimodal or unimodal. A possible estimate of position can be computed as:

$$\hat{x}_t = \arg\max_{x_t}\left(p\left(x_t \mid z^{1:N}\right)\right) \tag{8}$$

There are many algorithm for maximum search in literature (*hill-climbing algorithm*, *gradient ascent*, etc). In this context we propose a different approach that uses some assumptions to improve computational efficiency. Our search method does not need any prior position estimate and is not affected by local maxima problems.

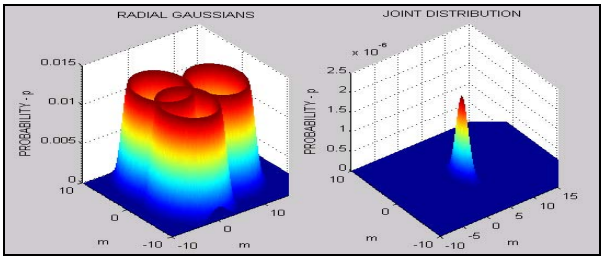


Fig. 5. Multilateration with three beacons. Radial Gaussians corresponding to the landmarks are shown on the left. On the right side the joint probability is plotted.

Necessary condition to have a maximum in (7) different from zero with coordinate $\hat{x}$ is that each Radial Gaussian, contained in the product, must be positive in $\hat{x}$. Let's consider a Radial Gaussian: since it has a normal cross-section the function is always positive and it tends asymptotically to zero for distances far from r_k. In practice probability is negligible for distances outside the range $[r_k\text{-}4\sigma_{r,k};r_{k+}4\sigma_{r,k}]$. This assumption allows us to search the maximum along the circumferences centered on one beacon with radius within the above-mentioned range. As described in next paragraph this is not a limitation because if the maximum was located outside this zone, it would be considered unreliable. Our algorithm is called *circular search*. Let's remember that the approach is the same for three-dimensional localization. In 3D the algorithm searches maximum on spherical surfaces. Estimation can be performed asynchronously and is independent from previous steps, so Probabilistic Shaping provides a localization service *on demand*.

E. Judgement

The result of estimation phase is a best guess on robot position. Substituting the Cartesian coordinate of the estimate into the closed-form distribution that represents joint probability we can obtain the corresponding probability p_{max}. Our decisional mechanism uses p_{max} as a metric for reliability. We can consider estimates to be reliable when $p_{max} > p_{th}$, where p_{th} is a threshold that depends on the parameters of the algorithm and on requirements of robustness of localization system. When an estimate is not reliable localization is performed through dead-reckoning (*rollforward error recovery*). Notice that Probabilistic Shaping estimates only position of the robot. Heading information can be easily obtained from navigation sensors.

IV. Robustness and Fault Tolerance Test

A. Scenario

Simulation scenario is a *logistic space* in which generic goods are stored. A possible representation of simulated environment is shown in Fig. 6. The robot can move through the corridors, avoiding obstacles (blue blocks). Robot's real path is shown in green. Tags (cyan dot) are dispersed in the environment and they are used by the robot as beacons for localization. Tags positions is not exactly known but their location is normally distributed with covariance matrix $\Sigma = diag(\sigma_{tag}^2)$. In nominal conditions σ_{tag} = 0.2 m. Mean positions of tags are on the vertices of 5 m × 5 m squares. In standard conditions each tag provides range information within 8 m and the distance measurements are normally distributed with standard deviation σ_r = 1 m.

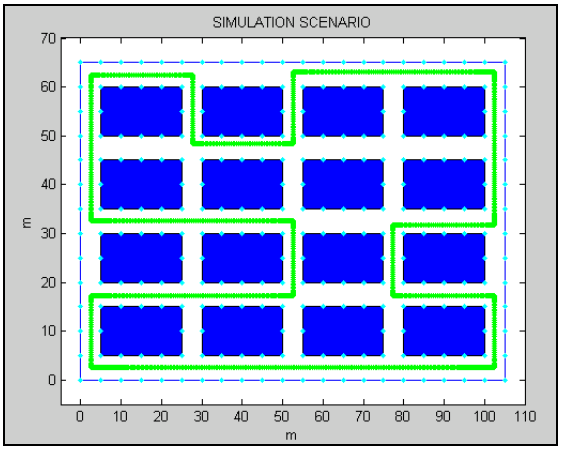


Fig. 6. Simulation scenario. Robot moves along green trajectory. Cyan dots represent nominal positions of tags.

Robot move along straight lines (this information is not used *a priori*) with parabolic speed profile: it starts from speed equal to zero, reaches maximum velocity (1 m/s) and stops at the end of straight line. The overall path should be long enough for providing reliable statistical results. In our simulations it is 5.1 km long (green path is covered ten times). Navigation sensors are able to provide a position estimation that can be drawn from a symmetrical normal distribution centered on real position, with standard deviation equal to 5% of distance travelled. Robot is equipped with a single reader that interrogates tags within reading range, with period T_m = 0.05 s. After an interrogation a tag provides a successful answers with probability p_A = 0.95. Otherwise we obtain a false-negative reading as mentioned above. The system must provide a position estimate with period T_e = 0.5 s.

TABLE I
Nominal Conditions Parameters

Symbol	Quantity	Nominal value
l	Square side	5 m
σ_{tag}	Standard deviation of tag position	0.2 m
R	Reading range	8 m
σ_r	Standard deviation of range measurements	1 m
v_{MAX}	Maximum speed of parabolic trajectories	1 m/s
d	Minimum distance covered	5 km
σ_x	Standard deviation of navigation sensors estimation	5% of distance travelled
T_m	Measurement period	0.05 s
T_e	Estimation period	0.5 s
p_A	Probability of tag's answer after reader interrogation	0.95

B. Adverse Situations Hypothesis

We classify under the general term of *adverse situations* all the threats to dependability. In our framework they can be divided in *fault hypothesis* and *environmental perturbations*. The former tests fault tolerance of localization system. The latter deals with robustness. With respect to Section II we limit our investigation to physical faults. As described later some cases of design faults can be solved within our framework. Interaction faults, instead, are extensively solved using ad hoc protocols and cryptography. In our scenario we simulate the following fault hypothesis:

--*Sensor Malfunction*: first experiment tests system response when the number of missing readings increases. Simulations are performed for p_A = 0.8 and p_A = 0.6. The second test verifies the response to outliers. A certain number of sensors (5% of the total number of tags), randomly chosen, transmits signals corresponding to a wrong power level. In a *power boost* situation range measurements are 1/2 of nominal measures (affected by noise). This error can cause false-positive readings. When a *power deficit* occurs, instead, the range read is 2 times nominal range;

--*Sensor Silence*: in the case of *single sensor silence*, 5% of the total number of tags does not transmit any information. The system must detect fault and perform localization. In *extensive sensor silence* situation all tags belonging to 2 contiguous squares don't give range information. Localization system has to assure fail-safe recovery, coming back to normal mode when correct measurements are acquired again.

In our beacon-based scenario environmental perturbation can be divided in:

--*Tag perturbation*: robustness is tested increasing uncertainty on tags positions. Statistics are performed for σ_{tag} = 0.3 m and σ_{tag} = 0.4 m. Beyond these values a SLAM approach assures better results;

--*Kidnapped robot*: in this case the robot, moving along its trajectory, does not acquire neither range measurements

nor information from navigation sensors during a period of time. This can be the case of a malicious attack or can occur when there is a fault in the controller (no navigation information or range measures are available). As consequence robot position can abruptly change from last correct estimate.

C. Probabilistic Shaping Settings

As a rudimental form of *Geometric Dilution of Precision* (GDOP) we consider only the information from the four nearest tags, in order to reduce non modeled dynamics. Reader interrogates tags ten times to provide a single estimation. The FIFO buffer contains the last 40 measurements (ten measurements from each tag). Increasing buffer length improves accuracy but also increases the delay of system recovery. We evaluate accuracy of localization using the euclidean distance between each estimate and correspondent real position. Table II contains the maximum, the mean value and standard deviation of localization error in normal conditions simulation.

TABLE II
LOCALIZATION ERROR IN NOMINAL CONDITIONS (SEE TABLE I)

Mean error (m)	Maximum error (m)	Standard deviation (m)
0.32	1.27	0.17

D. Fault Tolerance Results

Increasing the number of missing readings the FIFO buffer is updated slowly and *prediction phase* is affected by larger errors due to navigation sensors uncertainty. *Probabilistic Shaping* is able to obtain an accurate estimation also when measures acquired are not enough for other methods. Geometric triangulation for example needs at least three contemporary measures to solve planar localization. Probabilistic Shaping, instead, preserves memory of previous acquisitions, that are used when readings are not enough. Table III contains the outcome of simulation for different value of p_A.

TABLE III
TOLERANCE TO MISSING READINGS

p_A	Mean error (m)	Maximum error (m)	Standard deviation (m)
0.8	0.36	1.44	0.19
0.6	0.39	1.54	0.22

The second test on sensor malfunction verifies the tolerance to outliers. Results are shown in Table IV.

As stated in [12] Probabilistic Shaping provides accurate localization also when less than 30% of estimates are considered reliable. It means that the *breakdown point* [8], i.e. the proportion of outliers that an estimator can tolerate, is superior to 70% of the readings.

TABLE IV
TOLERANCE TO OUTLIERS

	Mean error (m)	Maximum error (m)	Standard deviation (m)
Power boost	0.46	1.59	0.29
Power deficit	0.41	1.44	0.23

Keeping memory of judgement history it is possible to detect sensor malfunction: if estimates that involve the same sensor are repeatedly considered unreliable, the corresponding fault probability increases. In order to give a fault alarm the following metric can be used: approximating the probability of successful estimation with percentage of reliable estimate (p_s), after k unreliable estimations, we can compute the fault probability as $(1-p_s)^k$. Comparing the fault probability corresponding to each tag with a probability threshold it is possible to detect sensor malfunction.

Results of single and extensive sensor silence are summarized in Table V.

TABLE V
TOLERANCE TO SENSOR SILENCE

	Mean error (m)	Maximum error (m)	Standard deviation (m)
Single Sensor Silence	0.40	1.46	0.22
Extensive Sensor Silence	0.40	1.47	0.22

When localization is performed the robot can compare its readings with nearest tags of the map, obtaining an indicator of missing readings for each tag. Given p_A (in a real situation can be found empirically) the probability of sensor fault, after h missing readings, is $(1-p_A)^h$. Comparing the indicator of missing readings with a probability threshold, the system is able to detect anomalies. In our simulation sensor silence situations are always detected and no false alarm occurred. We stored missing readings indicator and fault probability of each tag in two vectors, called *fault vectors*. Through statistic on these vectors we are able to disambiguate normal conditions from fault situations. An example of fault vector is shown in Figure 7.

E. Robustness Results

Perturbing tag positions we obtained the result shown in Table VI. Also in this case localization error remains upper-bounded within acceptable values. In a real case, tag perturbation errors occur when operators place tags in inexact positions during system start-up. We simulated the case of uncertain positioning of all tags. Note that if a perturbation is limited to a single sensor, it can be similar to the case of sensor malfunction, because the bias in tag's position causes an offset in distance measurement.

TABLE VI
ROBUSTNESS TO TAG PERTURBATION

σ_{tag}	Mean error (M)	Maximum error (m)	Standard deviation (m)
0.3 m	0.37	1.36	0.2
0.4 m	0.49	1.65	0.26

During system start-up it is also possible that an operator places a tag in a completely wrong position. In this case the mismatch between real position and ideal one is so large that no reliable estimation is obtained when this

tag is read. As consequence, the localization system detects the situation as sensor fault, allowing users to solve it. These are examples of design faults that can be solved within our method.

In the case of kidnapped robot accuracy can not be used as metric for performance evaluation. When target is kidnapped, localization error explodes, because the robot is moving without any information to perform localization. In order to avoid catastrophic failure, when new measurements are acquired, the system has to recover with a time delay of the same order of magnitude of estimation period. *Recovery time* can be used for performance evaluation. In nominal conditions, recovery time for Probabilistic Shaping is less than $2T_e$ (only one estimation is completely wrong). In general it depends on the buffer length and on FIFO policy, since, in order to have a correct estimate, all measurements in the buffer should be relative to the new position. In our simulations, using a common personal computer, an estimation period is performed in 0.12 s.

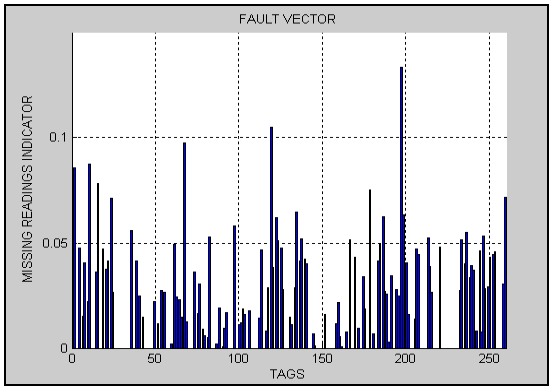


Fig. 7. Fault vector in nominal condition simulation.

V. Conclusion

We proposed fault tolerance and robustness tests for beacon-based localization. Applying concepts of robust statistics, we provided a set of fault hypothesis and environmental perturbations, in order to study the response of localization systems to adverse situations. We tested our localization model, called *Probabilistic Shaping*, that performs global localization. The method stores measurements in a buffer and provide an estimate *on demand*, since estimation phase can be computed independently from measurement acquisition. This consideration implies that the only computational complexity is limited to estimation and no overhead occurs when localization service is not required. The method provides position-only estimates, so heading, when needed, can be derived from navigation sensors measurements. The outcome of the tests is that localization error remains upper-bounded within less than 2 m also when anomalies occur. Mean errors is inferior to 0.49 m in every simulation and maximum error occurred was 1.65 m (starting from range measurements with standard deviation 1 m and uncertain beacons' position). Our model is able to assure fail-safe mode and to detect faults. Fault detection is performed through statistic on *fault vectors*, that contain information on missing readings and sensor malfunction. In all tests no catastrophic failure occurred. Probabilistic Shaping can also solve kidnapped robot problem assuring real-time system recovery. Robustness is achieved using a decisional mechanism, called *judgement*, that evaluates reliability of each estimate. Our probabilistic model is particularly suitable for localization based on radio beacons: it performs a spatial average of measurements, overcoming fluctuations and local biases, and can tolerate outliers without error explosion. Current work is focused on comparative study between classical Bayesian methods (Kalman Filter, Particle Filter) and the proposed model.

References

[1] J. Hightower, R. Want and G. Borriello, *SpotON: An indoor 3D location sensing technology based on RF signal strength*, UW CSE 00-02-02, University of Washington, Department of Computer Science and Engineering, Seattle, WA, February, 2000. http://www.cs.washington.edu/homes/jeffro/pubs/hightower2000indoor/hightower2000indoor.pdf

[2] A. Bekkali, H. Sanson and M. Matsumoto, *RFID Indoor Positioning Based on Probabilistic RFID Map and Kalman Filtering*, Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMOB), 2007.

[3] D. Kurth, *Range-only robot localization and slam with radio*, Master's thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May, 2004.

[4] J. Djugash, S. Singh, and P.I. Corke, *Further Results with Localization and Mapping using Range from Radio*, International Conference on Field & Service Robotics (FSR '05), July, 2005.

[5] A.I. Mourikis, S.I. Roumeliotis, *Performance analysis of multirobot Cooperative localization*, IEEE Transactions on Robotics, 22(4), pp. 666-681, 2006.

[6] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, Cambridge (Mass.), MIT Press, 2005.

[7] B. Lussier, R. Lampe, R. Chatila, J. Guiochet, F. Ingr, M.-O. Killijian, and D. Powell, *Fault Tolerance in Autonomous Systems: How and How Much*, In Proceedings of the 4th IARP/IEEE-RAS/EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, 2005.

[8] B Lussier, R Chatila, F Ingrand, M.-O. Killijian and D Powell, *On fault tolerance and robustness in autonomous systems*, in Proceedings of the 3rd IARP-IEEE/RAS-EURON Joint Workshop on Technical Challenges for Dependable Robots in Human Environments, 2004.

[9] D. Hahnel, W. Burgard, D. Fox, K. Fishkin and M. Philipose, *Mapping and localization with RFID technology*, in Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2004), New Orleans, 2004.

[10] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, Springer-Verlag, pp. 98-99, 2008.

[11] E. Olson, J. Leonard and S. Teller, *Robust range-only beacon localization*, in Proceedings of Autonomous Underwater Vehicles, 2004.

[12] L. Carlone and B. Bona, *Probabilistic Shaping with radial gaussian: a probabilistic approach to 2D and 3D robotic multilateration*, in Proceedings of the 2nd Israeli Conference on Robotics (ICR 2008), Israel, 2008.

This page is left blank intentionally

Line Following and Ground Vehicle Tracking by an Autonomous Aerial Blimp

VIDEO

David Jerónimo [1], *Ricardo Alcácer* [1], *F. C. Alegria* [2], *Pedro U. Lima* [3]

Abstract – **In this paper we introduce an autonomous aerial blimp testbed. The robot has onboard vision and computation capabilities, based on a digital signal processor. We also present the realistic hardware-in-the-loop simulator developed over USARSim. This development environment enabled fast prototyping and implementation of navigation primitives for the blimp, namely vision-based line following and ground vehicle tracking. Results of the indoor operation of the real blimp are presented.**

Keywords: Aerial blimp; autonomous navigation; vision-based path following; vision-based vehicle tracking.

I. Introduction

Aerial blimps, together with fixed wing airplanes, helicopters and quad-copters, have been among the most popular unmanned aerial vehicles (UAVs) used in research in recent years. They have, over the other types of UAVs, the advantage of intrinsic stability and safeness, low noise, low vibration, vertical take-off and landing with hovering capabilities for a higher payload-to-weight-and-energy consumption ratio [5]. On the other hand, their dynamics is hard to identify, and non-conventional methods may have to be used to determine unknown parameters [3]. Previous work on blimps has used single camera vision to fly around targets in indoor environments [1], to emulate (indoors) underwater station keeping and docking operations [6], to track simple objects [4] or for outdoor environmental monitoring [5]. Stereovision was used in [2] for outdoor terrain mapping. Indoor solutions often use non-fully autonomous blimps, as the control algorithms run on ground stations that communicate through radio-frequency (RF) signals with the blimp motors and cameras. On the other hand, larger outdoor blimps have enough payload to carry on board more sensors, such as GPS, gyroscopes or wind speed meters.

In this paper, we introduce an indoor autonomous aerial blimp with onboard vision and computation capabilities, based on a digital signal processor (DSP), shown in Fig. 1. We also present the realistic hardware-in-the-loop simulator developed over the USARSim simulator. This development environment enabled fast prototyping and implementation of navigation primitives for the blimp, namely vision-based line following and ground vehicle tracking in real outdoor scenarios. Results of the indoor operation of the real blimp are presented.

The work is part of a long-term project of the Institute for Systems and Robotics (ISR) at Instituto Superior Técnico, in the area of cooperative navigation of rescue robots [7]. This project aims at endowing a team of outdoor (ground and aerial) robots with cooperative navigation capabilities, so as to demonstrate the ability of the robots to act individually and cooperatively in search and rescue-like operation scenarios. This project intends to integrate a number of autonomous agents working in formation capable of interacting and co-operating between each other in a disaster situation such as an earthquake, where conditions are too adverse or difficult for human intervention and a rapid intervention of rescue teams is essential so as to prevent or minimize casualties. The blimp's mission is to survey the land while mobile robots on the ground move in, keeping permanent contact with the blimp and obtaining information about the ground and other matters, thus serving as an information transmission relay between the land robots and the base station.

This paper is focused on the design, development and implementation of all the blimp electronics, sensing and control systems that enable its full autonomy. The linearized blimp dynamics was identified using measurements made with the real robot flying in an indoor sports pavilion, and control algorithms were designed to follow ground lines and to track a ground vehicle.

Fig. 1. The Passarola blimp.

The control algorithms were implemented in the onboard DSP and the whole system was calibrated using both

The authors' affiliations are:
1 Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, davidjeronimo@gmail.com
2 Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, ricardoalcacer@gmail.com
3 Instituto de Telecomunicações, Instituto Superior Técnico, Lisbon, Portugal, falegria@lx.it.pt
4 Institute for Systems and Robotics, Instituto Superior Técnico, Lisbon, Portugal, pal@isr.ist.utl.pt

simulation tests and the real blimp. Results with real robot indoor experiments are presented.

II. BLIMP'S HARDWARE

We have designed the blimp's navigation system, by adequately integrating several off-the-shelf components:

- 1 non-rigid envelope filled with helium;
- 2 dual blade main propellers (279x120 mm);
- 1 dual blade tail propeller (180x80mm);
- 1 shaft for adjustment of the angle of the main propellers;
- 1 Analog Devices ADSP-BF561 Blackfin evaluation board;
- 1 Sony HQ1 Helmet Camera;
- 2 Graupner Speed 400 motors for the main propellers;
- 1 Multiplex Permax 480 motor for the tail propeller;
- One motor and encoder for the main propellers' shaft;
- PWM controllers for the motors;
- 1 Reedy Black Label 2, Ni-MH battery (7.2 V, 3700 mAh);
- 1 Reedy Black Label 2, Ni-MH battery (7.2 V, 3300 mAh);
- 2 Flight Power EVO 25, Li-Po batteries (11.1 V, 1500 mAh).

In Fig. 2 we show how the different electronic components are connected.

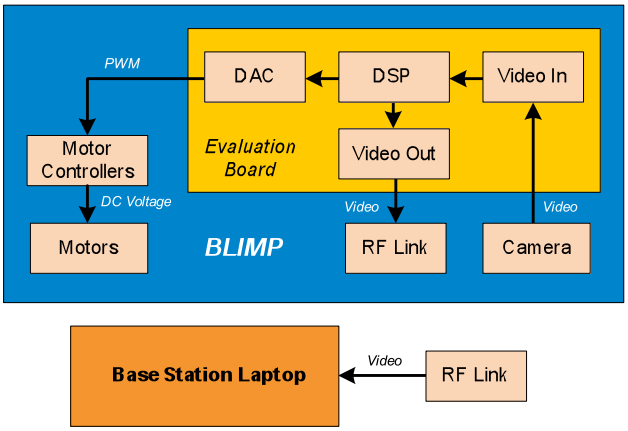


Fig. 2. Connection diagram of the hardware inside the blimp.

The DSP is used to implement the image processing algorithms, which determine the relative position of the blimp regarding the target (line or vehicle on the ground). It also implements the controller and generates the PWM signals, which are sent to the propeller motors to control their rotational speed. Although the blimp has the possibility to adjust the angle of attack of the main propellers in order to adjust the blimps height, this was not implemented in the current control loop.

All the electronics of the blimp are housed in two canopies attached to the underside of the envelope as seen in Fig. 3. The video camera can be seen between the two canopies. In the same figure some cables can be seen coming out of the rear canopy and which go to the ground. They are used to load the software, developed in C using Analog Devices' VisualDSP++ programming environment, into the DSP, and to charge the batteries. During normal operation of the blimp they are removed. An RF link is used to transmit an image from the blimp to ground control in order to monitor the blimp's operation.

Fig. 3. Underside of the blimp showing the two canopies which contain the hardware and the main propellers.

In order to reduce the development time, an evaluation board for the DSP was used. This board has, besides the DSP, video coders and decoders and the corresponding connectors for video input and output (used for debug purposes), as well as digital-to-analogue converters for audio output. These audio outputs are used to drive the motor controllers. A proper pulse width modulation signal is created in software with the appropriate duty cycle to set the rotational speed of the motors (Fig. 4).

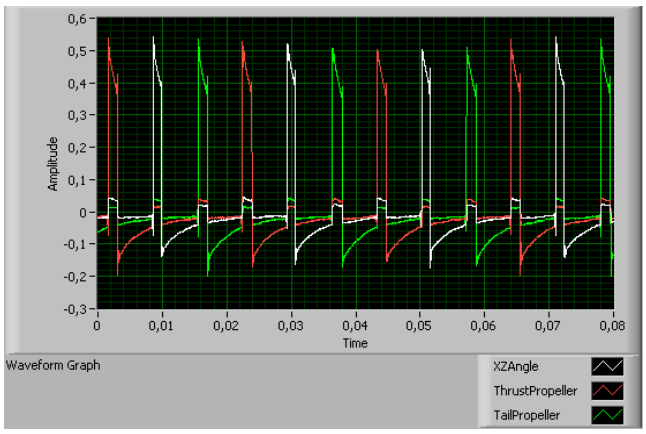


Fig. 4. Motor control signals acquired by the ADC and represented in LabVIEW used in the hardware-in-the-loop simulation setup.

III. SIMULATION SETUP

In the development stage, a setup was built which allows the closed loop control software to be tested with hardware in the loop. This setup, whose diagram can be seen in Fig. 5,

consists of two personal computers (PCs): the Development PC and the Simulation PC. The former, running VisualDSP++, is used for software coding and debugging. The latter, running USARSim [8], is used to create the virtual world in which a virtual blimp would exist. USARSim is a high-fidelity simulation of robots and environments based on the Unreal Tournament game engine. A dynamically accurate model of the Passarola blimp was created using vehicle classes from the Karma Physics Engine, which is a rigid multi-body dynamics simulator that is part of the Unreal development environment.

In order to ease as much as possible the transition from software development to full system deployment, a hardware-in-the-loop type of setup is used in which the blimp DSP is inserted in a loop together with the USARSim simulator (see Fig. 5).

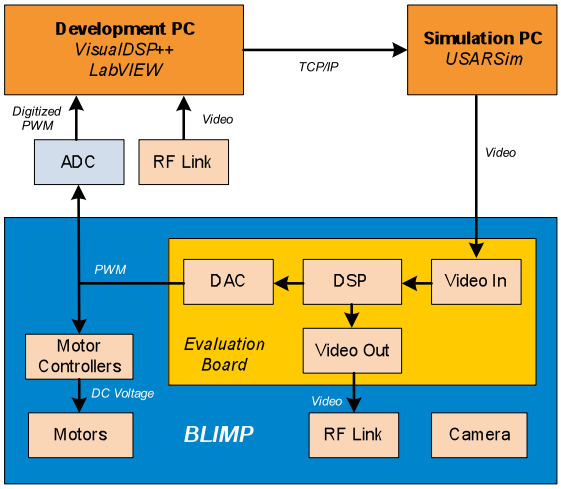


Fig. 5. Connection diagram of the simulation setup.

The image of the virtual world (Fig. 6), from the view point of the blimp's video camera is fed to the DSP and the control signals produced by the DSP (audio outputs of the DSP evaluation board) are digitized, using a National Instruments USB-9233 data acquisition board, and transferred to the development PC using LabVIEW. This data, in turn, is sent to the Simulation PC in order to control the speed of the virtual blimp propellers.

The testbed with the simulation setup is depicted in Fig. 7. The DSP signals for motor control were also connected to the real blimp motors in the laboratory to assert that they operated correctly.

IV. Modeling and Vision-Based Control

For vehicles with 6 degrees of freedom (DoF), 6 independent coordinates are needed to determine their position and orientation. The first 3 coordinates $\boldsymbol{\nu}$ and their temporal derivatives corresponds to the position and translational velocity of the vehicle, along the x, y and z axes, while the last 3 coordinates $\boldsymbol{\eta}$ and their temporal derivatives are used to describe its orientation and its rotational velocity.

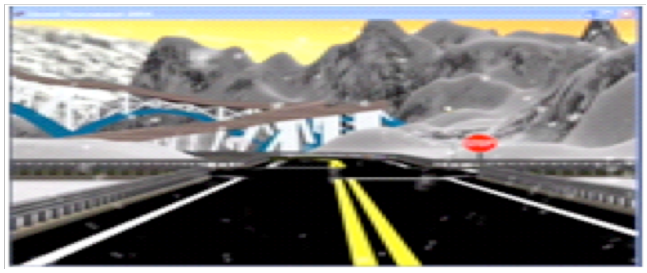


Fig. 6. Image of the virtual world from the point of view of the blimp's camera.

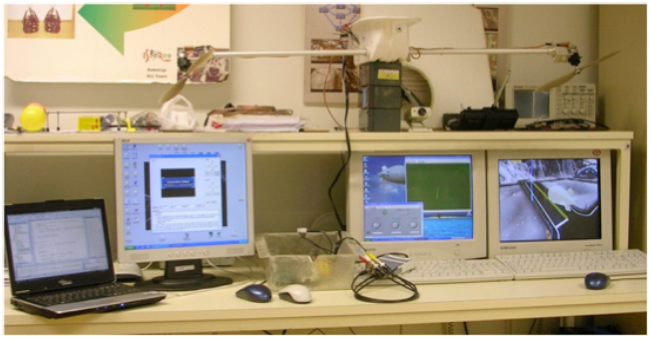


Fig. 7. Picture showing the hardware-in-the-loop simulation setup.

The PASSAROLA blimp is under-actuated, i.e., it has less control inputs than DoF, so the vehicle control is limited.

A. Dynamic Model

The equations of movement of vehicles that move immersed in a fluid, can be written in a vector form [10]:

$$\dot{\eta} = J(\eta)\nu$$
$$M\dot{\nu} + C(\nu)\nu + D(\nu)\nu + g(\eta) = \tau$$

where J is the Jacobian matrix, M is the system inertia matrix, C is the Coriolis-centripetal matrix, D is the damping matrix, g is the vector of gravitational/buoyancy forces and moments and τ is the vector of applied torques and moments. Expressed in the blimp center-of-mass centered frame (with the x axis pointing towards the movement direction, over the longitudinal axis of the blimp, the z axis pointing downwards, and the y axis completing an orthonormal coordinate system), the linearization of the kinematic and dynamic equations leads to the state space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0_{6\times6} & I_{6\times6} \\ -M^{-1}g & -M^{-1}D \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0_{6\times3} \\ M^{-1}B \end{bmatrix} u$$

where x_1 is the 6x6 position and translational velocity vector and x_2 the 6x6 orientation and rotational velocity vector. B is the actuators position matrix, in the blimp's frame. Observing carefully the achieved model, it is clear that it can be divided in two different systems, entirely decoupled. Consequently, we are in the presence of two independent systems, one that describes the blimp's motion on the vertical plane with $u = [X \quad Z]^T$ (X and Z force

components) and the other that models its rotational motion over the z axis with $u = F_{MT}$ (heading moments).

The model parameters were identified by adequate experimental tests carried out on the aerial blimp, using the transfer function version of the linear state equations above.

Briefly, the accomplished experiments were made separately for the two decoupled subsystems. With the robot at rest, it was applied (at t=0 s) a step at the appropriate input of each subsystem, corresponding to a sudden change on the PWM value of the actuators, equivalent to a 2 N force. The sequence of blimp positions/orientations was measured for each test and the final motion was plotted for visual inspection and comparison with Simulink blimp model simulations.

The system identification was carried out using the Matlab *Ident* toolbox, from the input/output set obtained for each subsystem and using the ARX parametric model to find the best match for each subsystem.

B. CONTROL LOOP

A feedback linearization control law was successful tested in simulation, using the identified blimp dynamic model, and provoking mismatches between the parameters of the actual model and of the model used by the controller. However, the blimp onboard DSP has not enough power to implement the full controller for the real blimp. Therefore, we only used the inertia matrix in the control law

$$u = K_{3\times 6} M (K_{P_{6\times 6}} e + K_{D_{6\times 6}} \dot{e})$$

and introduced the gain matrices K, K_P and K_D to scale the error e between the desired and actual blimp positions (or orientations), measured on the image, and its derivative.

Due to the decoupling between the x-z vertical plane and rotational motion models, two separate controllers were designed, one for the translation along the x axis (the altitude was not controlled and simply kept by the balance between the blimp impulsion and weight) and another for the rotation over the z axis (*yaw*). The former keeps a desired blimp speed when following a line or tracks the ground vehicle speed, while the latter keeps the blimp aligned with a desired direction (the ground line tangent or the ground vehicle heading). The last two subsections focus on the image processing for each of the cases.

C. GROUND LINE FOLLOWING

To determine ground lines to be followed, a Sobel edge detection algorithm [9] was applied to the image acquired by the video camera (Fig. 6).

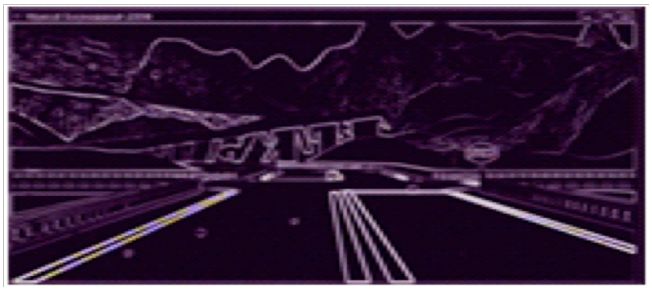


Fig. 8. Result o the edge detection algorithm. The gray level is proportional to the luminosity gradient.

The gradient image obtained, depicted in Fig. 8, was transformed into a black and white image (Fig. 9) by the use of a threshold.

Fig. 9. Black and white image of the edges.

The next step was to use the Hough Transform [9] to detect the straight lines in the image (Fig. 10). The four more predominant straight lines were selected.

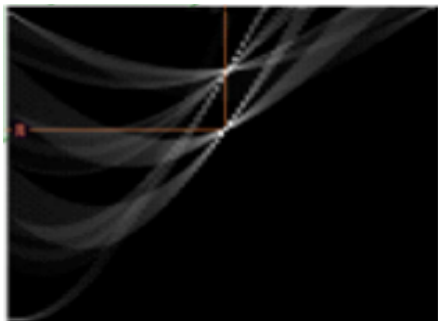


Fig. 10. Hough transform space. The horizontal coordinate is the straight line angle and the vertical one is the straight line distance to the origin.

In order to determine the target direction, a novel procedure was adopted. This procedure consists in intersecting the four straight lines determined previously with a half-circle centred at the image centre and with a diameter of half the image's height (red half-circle in Fig. 11). This results in at most 4 points over the circle (there may be straight lines that do not intersect the circle at all).

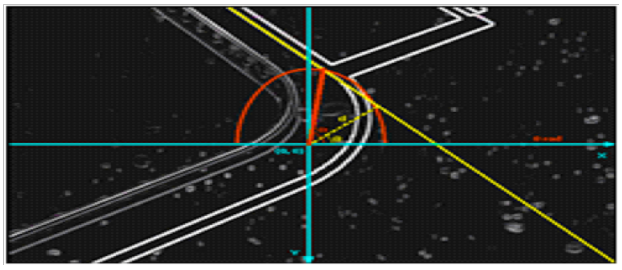


Fig. 11. Result of the image processing algorithm. The target direction is represented by the red line segment.

In the initial stages of the algorithm, when the autonomous navigation system is started, the closest intersection point to the image vertical axis is selected. In the following steps, the selected intersection point is the one closest to the previous intersection point.

The desired blimp speed along the *x*-axis is kept constant, but multiplied by a reduction factor that depends on the trajectory characteristics, e.g., the speed reduction is larger on curved lines.

D. GROUND VEHICLE TRACKING

In this case, the two control systems must track dynamic references.

The *x*-axis reference is the *x* coordinate of the tracked vehicle centre of mass in the blimp image frame. The goal of the control system is to reduce this coordinate to zero, by actuating on the speed of the blimp blade main propellers.

The *y* coordinate of the tracked vehicle centre of mass in the blimp image frame and the angle between the tracked vehicle centre of mass in the blimp image frame, and the angle between the image frame y-axis and the vector *E* that links the image frame origin to the vehicle centre of mass (see Fig. 12), play a role in the rotation controller, which acts on the blimp blade tail propeller to reduce the angle to $\pi/2$.

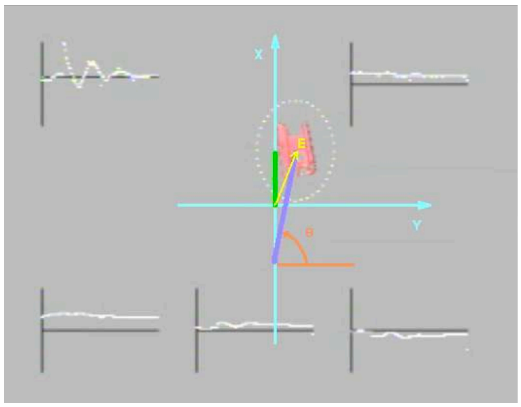


Fig. 12. Real blimp following a line on the ground. The image includes plots of the temporal evolution of different quantities such as the duty cycle of the motor control PWM signals.

V. INDOOR TESTS

After the image processing and control algorithms were completed and tested in the virtual environment of the simulation setup the blimp was assembled and tested in real conditions, in an indoor sports pavilion, under changing light conditions but no wind. Fig. 13 shows the blimp successfully following a white line in the pavement of a sports arena where other white lines were present.

In Fig. 14 shows the image sent by the blimp to the ground station using the RF link. This image includes plots of the line being followed (blue line) as well as of the temporal evolution of different quantities such as the duty cycle of the motor control PWM signals.

Fig. 13. Real blimp following a line on the ground.

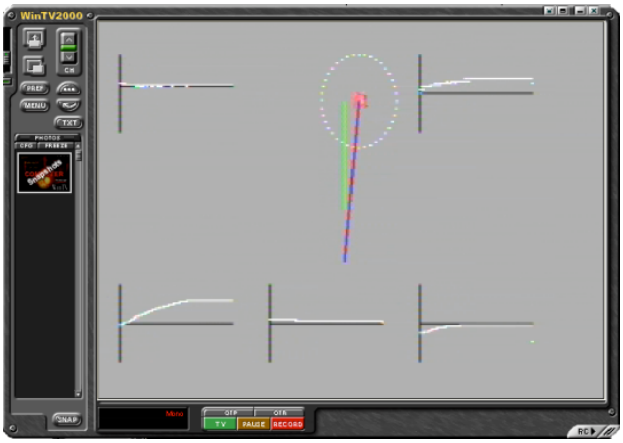


Fig. 14. Output of the DSP sent to ground control.

Fig. 15 shows the blimp tracking a tele-operated iRobot ATRV-Jr vehicle. The results of tracking figure-8 and U-shaped trajectories of the ground robot were quite satisfactory and matched quite accurately previous simulations made with the simulation setup.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we introduced an autonomous indoor blimp, with onboard computation electronics and a video camera. A hardware-in-the-loop test setup was also presented which enables accurate and fast system development and easy portability to real operation conditions.

Fig. 15. Real blimp tracking a tele-operated iRobot ATRV-Jr robot.

There are several developments that can be carried out in the future to improve the system. The electronics can be miniaturized through the development of a custom made board containing the DSP, the image coders, the DACs and the motor controllers. This would diminish the size, since a lot of unused electronics existent in the DSP evaluation board used would be eliminated. It would also lower the weight and reduce the power consumption which, in turn, would allow fewer batteries to be used.

Towards outdoor operations, we intend to endow the current blimp with more powerful motors. Regarding the navigation system, we intend to install onboard a GPS receiver, so that the blimp can follow a set of predefined waypoints.

ACKNOWLEDGMENTS

The authors acknowledge the support, by their Pluriannual fundings, of the Institute for Systems and Robotics and the Instituto de Telecomunicações at Instituto Superior Técnico.

REFERENCES

[1] T. Fukao, K. Fujitani and T.Kanade, "An Autonomous Blimp for a Surveillance System", Proceedings of the 2003 IEE/RSJ Intl. Conference on intelligent Robots and Systems, Las Vegas, Nevada. October 2003

[2] E. Hygounenc, I.-K. Jung, P. Soueres, S. Lacroix, "The Autonomous Blimp Project of LAAS-CNRS: Achievements in Flight Control and Terrain Mapping", International Journal of Robotics Research, Vol. 23, No. 4-5, 473-511, 2004

[3] J. Ko, D. J. Klein, D. Fox, D. Haehnel, "Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp", IEEE International Conference on Robotics and Automation, pp. 742-747, 2007

[4] H. Zhang, J. Ostrowsky, "Visual Servoing with Dynamics: Control of an Unmanned Blimp", IEEE International Conference on Robotics and Automation, Michigan, 1999

[5] A. Elfes, S. Bueno, M. Bergerman, J. Ramos, "A Semi-Autonomous Robotic Airship for Environmental Monitoring Missions", IEEE International Conference on Robotics and Automation, 1998

[6] S. Zwaan, A. Bernardino, J. Santos-Victor, "Vision Based Station Keeping and Docking for an Aerial Blimp". Proc. of the IEEE/RSJ international Conference on Intelligent Robots and Systems, 2000.

[7] P. Lima, M. Isabel Ribeiro, Luís Custódio, José Santos-Victor, "The RESCUE Project – Cooperative Navigation for Rescue Robots", Proc. of ASER'03 - 1st International Workshop on Advances in Service Robotics, March 13-15, 2003 – Bardolino, Italy.

[8] Jijun Wang, "USARSim – A Game-based Simulation of the NIST Reference Arenas", Carnegie Mellon Technical Report.

[9] R. J. Schalkoff, Digital Image Processing and Computer Vision. New York: Wiley, 1989

[10] T. Fossen, Marine Control Systems Guidance, Navigation, and the Control of Ships, Rigs and Underwater Vehicles. Trondheim: Marine Cybernetics, 2002.

Real-time Teaching of Industrial Robots Using a Synchronised Stereoscopic Vision System

Paulo Malheiros, Paulo Costa, António Paulo Moreira and José Carlos Lopes

***Abstract*— In this paper a method to control and teach industrial robots in real-time by human demonstration is presented. This system uses high-intensity visual markers that make it sufficiently robust for industrial environments not requiring special lighting. An automated camera calibration method was implemented which enables any non-skilled user a quick and easy configuration of the system.**

The teaching of the robot is achieved by recording the detected points and replaying them to the manipulator, therefore this is a "teaching-by-showing" method.

This system can work with any kind of industrial manipulator capable of receiving remote positioning commands.

I. INTRODUCTION

Industrial robots are programmable multifunctional mechanical devices designed to move material, parts, tools, or specialised devices through variable programmed motions to perform a variety of tasks. Robots are generally used to perform unsafe, hazardous, highly repetitive, and unpleasant tasks. Most robots are set up for an operation by the teach-and-repeat technique. In this mode, a trained operator (programmer) typically uses a portable control device (a teach pendant) to teach a robot its task manually [1].

The required flexibility in manufacturing implies that both machine control and equipment interfaces need to be easy to change for new application scenarios [2]. Robot programming is not easy for novice operators and the cost of training them is often unaffordable especially for small companies. Thus low-cost and labour-saving robot teaching is greatly to be desired [3].

Consequently, new methodologies for programming robots quickly, safely and intuitively are desired. To this purpose, interactive programming interfaces that allow non-skilled users to program robots have been developed over the years. These developments require higher levels of abstraction and in some way tend to lead to machines that understand human-like instructions [4].

A simple real-time "teaching-by-showing" method has been developed for conventional industrial robots, shown in Fig. 1. This application focuses on improving efficiency by enhancing human-robot interaction for task generation. Although several developments have been done over the years, this technology is still far from industrial applications since previous approaches require special equipment (data gloves, position sensors, etc.) and specific environment condition (controlled lighting, large teaching equipment, etc.).

Paulo Malheiros is a PhD Candidate, Paulo Costa and António Paulo Moreira are Assistant Professors with the Department of Electrical and Computer Engineering, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias s/n, 4200 465 Porto, Portugal paulo.malheiros@fe.up.pt

José Carlos Lopes is an Associate Professor with the Department of Chemical Engineering, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias s/n, 4200 465 Porto, Portugal

Fig. 1. Teaching system with industrial manipulator

The presented method uses image processing synchronised with high intensity markers (e.g. LEDs). These markers are so bright that their light outshines the surrounding environment making them easy to detect using regular cameras. The synchronisation with the cameras allows the markers to be only active during the image acquisition (with a few milliseconds duration) which makes its apparent light very dim not interfering with human vision.

The markers are attached to the user's tools (e.g. paint spray gun) and using a stereoscopic vision system, shown in Fig. 2, it is possible to register the path in space and make the robot manipulator mimic the same movements. Three or more markers allow the detection of the six degrees of freedom of the tool so all the robot's axis are correctly controlled.

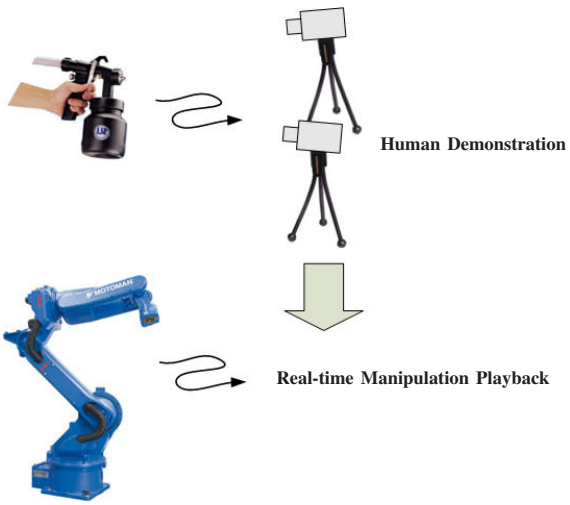


Fig. 2. Manipulator moving simultaneously with user's movement

The system can work in real-time because the markers are easily identified and classified, therefore the user can instantly see the resulting path in the manipulator. The teaching process is achieved easily by replaying the previously recorded sequence.

This system is ideal to work as a remote control which can be useful in hazardous and unsafe conditions, like controlling the space station robotic arm or moving radioactive material.

II. TEACHING SYSTEM

For the developed system we set the following assumptions:

- Uses common and non-expensive equipment.
- Non-obtrusive sensors so that users do their natural movements.
- Works in any industrial environment.
- Easy implementation for non-skilled users.

This system consists of two Firewire industrial cameras, one laptop, a synchronisation system and a set of LEDs. The cameras are two FireWire CCD Bayer industrial cameras with 640x480 resolution and external trigger that allows the synchronisation of their acquisition through an input clock. This synchronisation signal is controlled by a generic board with an 8-bit microcontroller which also triggers the LED markers. The synchronisation for the markers can be a radio or infrared signal thus reducing the cabling which can cause inconvenience with the user. In Fig. 2 is represented the use of this system with an industrial manipulator in a painting application.

An image processing application was developed in Object Pascal using Lazarus and using the 5dpo Component Library [6] which is an Open Source set of components developed by the authors for image acquisition. The application can also position the markers in a OpenGL 3D virtual world using GLScene components.

The markers are standard high-power LEDs which have approximately 4 Watt and come in several colours. The use of different colours allows the immediate classification of several markers by coloured image processing application. In Fig. 3 is represented an high-powered led easily detected by the camera which reduces the time needed to process the image, this is essential in real-time applications. Even reducing the camera aperture the marker is easily detectable while the surrounding environment fades away.

Having detected the the tool position this system can order any kind of industrial manipulator (e.g. XY Table) with a remote control interface. A Motoman HP6 Series industrial robot was used together with a Motoman NX100 Controller for this paper, shown in Fig. 1. The controller has a *Remote* mode that receives the control instructions sent by the PC through an Ethernet connection.

This system is easily installed and with little time and effort is fully operational. For this we need:

1) Install the cameras and the markers in the control tool.
2) Calibrate the cameras stereoscopy using the previous tool.
3) Fix the system scale and distortion using known world points.

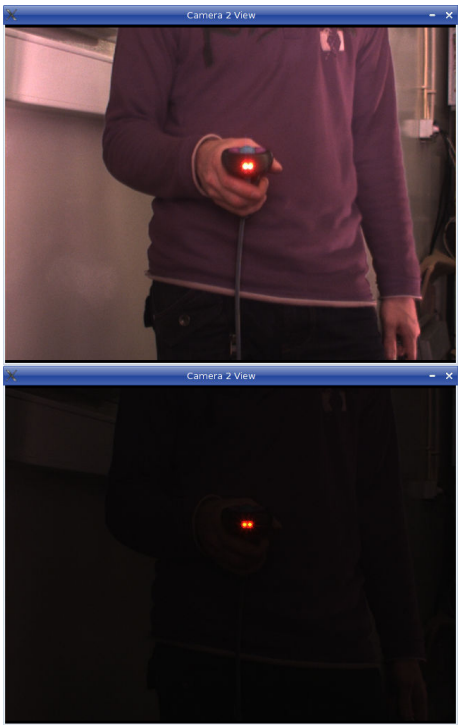


Fig. 3. LED detection in an uncontrolled environment. Only marker visible when camera's aperture is closed

4) Translate and Rotate the world coordinates to make them equal to the robot coordinates.
5) Reconstruct the tool position from the measured markers and playback in the robot.

In five steps we implement a high-precision and real-time teaching system for industrial robots.

III. CAMERAS CALIBRATION

Once the cameras are positioned in a new installation there is the need to mathematically model the imaging system. This is done using the *Fundamental Matrix* that encapsulates the epipolar geometry of the imaging configuration. We used the Normalised Eight-point Algorithm which is the simplest method of computing the fundamental matrix, involving no more than the construction and (least-squares) solution of a set of linear equations [7]. Finally it's possible to estimate the 3D position of any given point in a pair of images having the fundamental matrix determined.

We illustrate the steps taken to test the principle and possibilities of this technology. It was demonstrated using a single marker to control the position of the manipulator. Position and orientation are to be tested in the future as well as precision studies.

A. Notation

In this paper the following notation is followed:

- The vectors are represented by lowercase bold letters (e.g. $\mathbf{u}$) when it refers to 2D space in homogeneous coordinates and are thought of as being column vectors unless explicitly transposed. Uppercase bold letters when it refers to 3D space (e.g. $\mathbf{T}$) homogeneous coordinates.
- The matrices are represented by uppercase fixed size letters (e.g. A).

B. Camera Intrinsic Parameters

The first step was to determine and correct the cameras barrel distortion using a known pattern (Fig. 4). Once the second order barrel distortion and centre of the image were determined all the images were treated has not having any kind of radial distortion in the following stages.

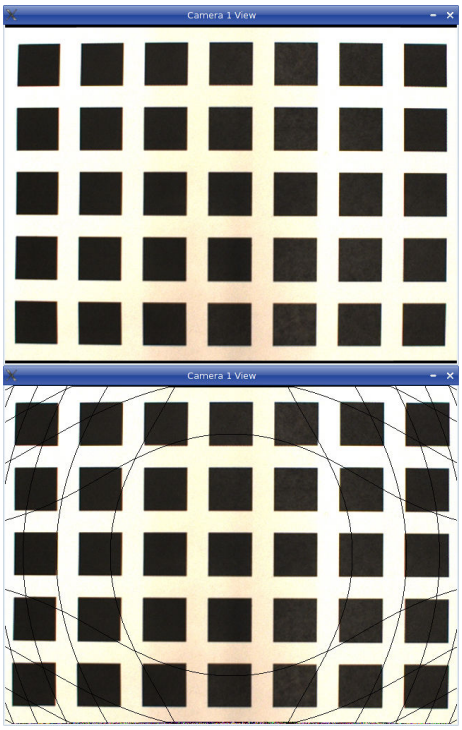


Fig. 4. Pattern used for determining the intrinsic parameters. Sample barrel distortion compensation

The inverse radial distortion function is the mapping from the distorted point $\mathbf{p}_d$ to the undistorted point $\mathbf{p}_u$. It can be concluded from the location of the point of sharp projection $\mathbf{p}$ that the radial distortion increases with the radius r. Thus, the inverse radial distortion function $f(r_d)$ can be approximated and parametrised by the following Taylor expansion [8]:

$$r_u = r_d + r_d \sum_{i=0}^{\infty} \kappa_i r_d^{i-1} \tag{1}$$

with

$$r_u = \sqrt{x_u^2 + y_u^2} \quad \text{and} \quad r_d = \sqrt{x_d^2 + y_d^2}$$

it follows that

$$x_u = x_d + x_d \sum_{i=0}^{\infty} \kappa_i r_d^{i-1} \tag{2}$$

$$y_u = y_d + y_d \sum_{i=0}^{\infty} \kappa_i r_d^{i-1} \tag{3}$$

For these tests only κ_3 was taken into account since practical tests have shown that this makes a good radial correction. The parameter κ_3 has the dominant influence on the kind of radial lens distortion. If $\kappa_3 > 0$, a barrel distortion and if $\kappa_3 < 0$, a pincushion distortion is compensated by f(rd). Thus, we simplify Eq. 2 and Eq. 3 to:

$$x_u = x_d + x_d \kappa_3 r_d^2 \tag{4}$$

$$y_u = y_d + y_d \kappa_3 r_d^2 \tag{5}$$

Finally the intrinsic matrix K (also called *camera calibration matrix*) maps the normalised image coordinates to the retinal image coordinates [10].

$$K = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Using the same pattern we were able to determine the focal length f. The horizontal and vertical scale factors, k_u and k_v, are the inverse of the size of the CCD pixel. u_0 and v_0 the intersection between the optical axis and the retinal plane.

C. Normalised Eight-point Algorithm

Binocular vision consists in using two cameras to view a point in the space $\mathbf{U}$. Fig. 5 shows the optical centers $\mathbf{C}_1$ and $\mathbf{C}_2$, $\mathbf{u}$ and $\mathbf{u}'$ are the images of $\mathbf{U}$ on the retinal planes Π_1 and Π_2.

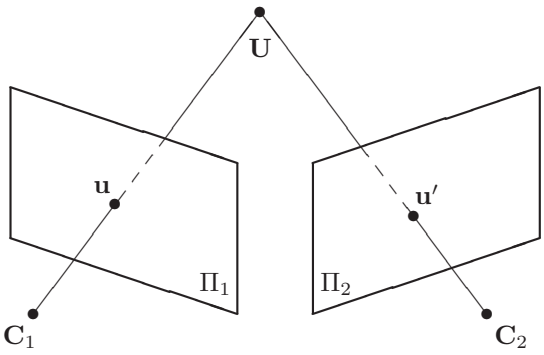


Fig. 5. Binocular vision

The fundamental matrix is a 3×3 matrix which relates corresponding points in stereo images defined by the equation

$$\mathbf{u}'^T F \mathbf{u} = 0 \tag{7}$$

for any pair of matching points $\mathbf{u}' \leftrightarrow \mathbf{u}$ in two images. Given sufficiently many point matches $\mathbf{u}_i' \leftrightarrow \mathbf{u}_i$ (at least eight) this Equation 7 can be used to compute the unknown matrix F [9]. Specifically, the equation corresponding to a pair of points $\mathbf{u} = (u, v, 1)^T$ and $\mathbf{u}' = (u', v', 1)^T$ in will be

$$\begin{aligned} &uu'F_{11} + uv'F_{21} + uF_{31} + vu'F_{12} + vv'F_{22} + \\ &vF_{32} + u'F_{13} + v'F_{23} + F_{33} = 0 \end{aligned} \tag{8}$$

The row of the equation matrix may be represented as a vector

$$(uu', uv', u, vu', vv', v, u', v', 1) \tag{9}$$

From all the point matches, we obtain a set of linear equations of the form

$$A\mathbf{f} = 0 \tag{10}$$

where $\mathbf{f}$ is a nine-vector containing the entries of the matrix F, and A is the equation matrix. The fundamental

matrix F, and hence the solution vector $\mathbf{f}$ is defined only up to an unknown scale.

The key to success with the eight-point algorithm is proper careful normalisation of the input data before constructing the equations to solve. A simple transformation (translation and scaling) of the points in the image before formulating the linear equations leads to an enormous improvement in the conditioning of the problem and hence in the stability of the result. The added complexity of the algorithm necessary to do this transformation is insignificant.

The suggested normalisation is a translation and scaling of each image so that the centroid of the reference points is at the origin of the coordinates and the RMS distance of the points from the origin is equal to $\sqrt{2}$ [7].

The fundamental matrix F is then obtained denormalising the previous result.

IV. 3D Reconstruction

Any camera can be now positioned in the world using their rotation matrix R and translation vector $\mathbf{T}$ determined from the fundamental matrix. The main property of the camera model is that the relationship between the world coordinates and the pixel coordinates is linear projective. This relationship remains linear regardless the choice of both coordinates [10]. Thus, the 3D point $\mathbf{U}$ and its camera projection $\mathbf{u}$ are related by

$$s\mathbf{u} = P\mathbf{U} \tag{11}$$

where s is a scale factor called *depth* and P is a 3×4 matrix called the *perspective projection matrix*.

To simplify we can consider the first camera as being in the origin then $(R, \mathbf{T})$ is the displacement from the first camera to the second, under the pinhole model, we have the following two equations

$$\begin{aligned} s\mathbf{u} &= K[I \mid \mathbf{0}]\mathbf{U} \\ s'\mathbf{u}' &= K'[R \mid \mathbf{T}]\mathbf{U}' \end{aligned} \tag{12}$$

where K and K' are the intrinsic, or calibration matrices of both cameras [11] determined previously. A 3D point is the intersection of two rays passing through the optical centers of each camera, and corresponding image points $(\mathbf{u}', \mathbf{u})$. In Fig. 6 is represented a set of reconstructed points used during the calibration. It represents the tool movement in space which is basically a random movement to result in random points.

The determination of the scale factor s was done placing a marker in the industrial manipulator and making this move a precise distance. The relation between the real distance and the measured distance gives this scale factor.

A 3D point is the intersection of two rays passing through the optical centers $\mathbf{C}_1$ and $\mathbf{C}_2$, and corresponding image points $(\mathbf{u}, \mathbf{u}')$. The three points $\mathbf{u}$, $\mathbf{u}'$ and $\mathbf{U}$ form a triangle. Thus, the final task is only to resolve reconstruct the 3D point through triangulation.

V. Path Detection

The demonstration of the technology of controlling and teaching industrial manipulators was the main purpose of the paper. The tests were made using only a single marker so the presented results are only of three degrees of freedom.

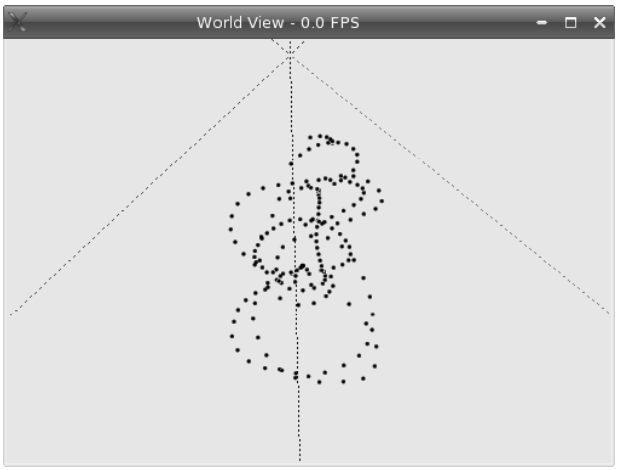


Fig. 6. Reconstructed calibration points

A. Real-time Playback

The reconstructed system are centred on the principal camera, in Fig. 6 the origin is away from calibration points which is the exact position of the camera looking at the points. Ideally the tool coordinates should be the same as the ones in the manipulator so this system can translate and rotate its coordinates.

The coordinates translation is done showing a central point with the tool, followed by the rotation of the axis showing two points placed in two of the robots axis. Fig. 7 shows the previous calibration points with the manipulator coordinates, the calibration was done in the middle of our working environment.

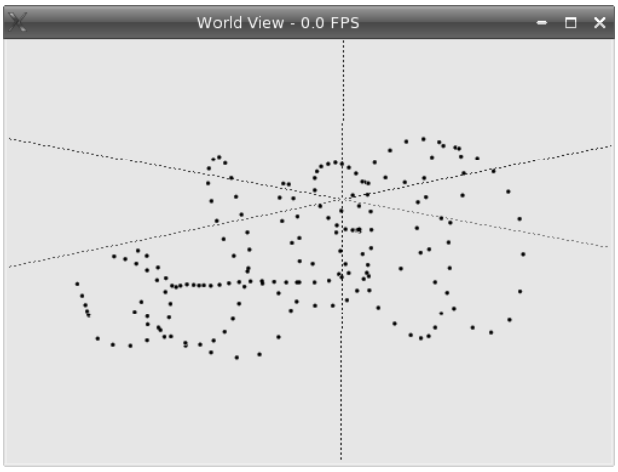


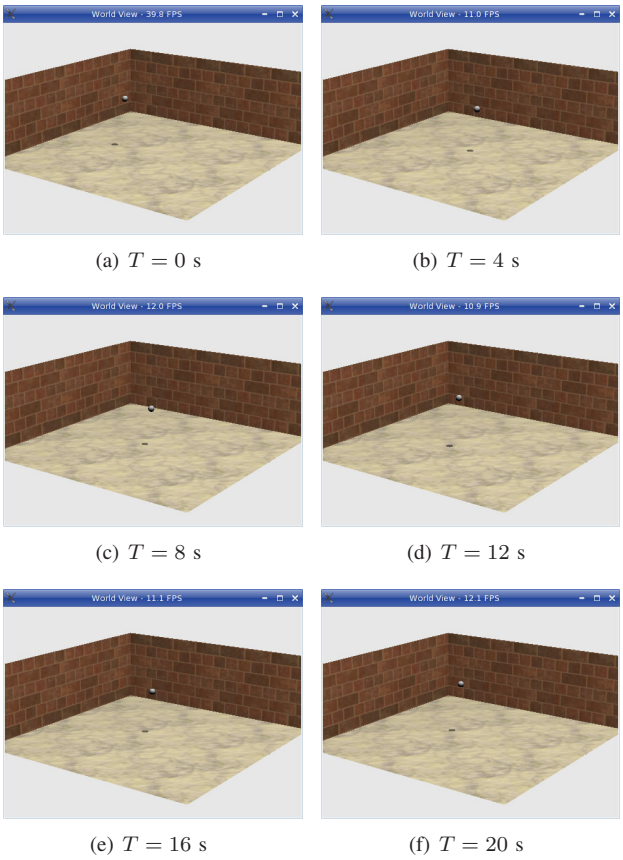
Fig. 7. Reconstructed calibration points with corrected coordinates

Having the the world coordinates correctly set this system also allows the definition of the working field, this increases the safety in the working environment. Any point outside this working area is not updated in the industrial robot.

Fig. 8 shows the virtual world with the detected marker in real-time. This 3D environment is an excellent tool for visually debugging the state of our measured markers [12].

B. Replay movement

The developed software allowed the recording of any shown trajectory. These trajectories could be replayed any



(a) $T = 0$ s (b) $T = 4$ s

(c) $T = 8$ s (d) $T = 12$ s

(e) $T = 16$ s (f) $T = 20$ s

Fig. 8. Sequence virtualisation of the measured marker

time into the manipulator, hence making this a simple teaching system.

These recorded points can be used in the future to find the best trajectory fitting into lines and curves. This will result in less instructions for the manipulator to execute, thus a faster and smoother movement.

C. Robot communication

At this point any industrial manipulator with a remote control mode can be used with this system. The Motoman controller can receive several types of commands, these can be movement instruction, control commands, memory access, etc.

For this paper the controller basically receives the 3D coordinates of the tool subsequently guiding the manipulator to that same coordinate. Once that position is reached the controller resends a new coordinate s[12]. The coordinates updating rate depends on the speed the manipulator takes to complete its current movement.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper a simple real-time teaching method for industrial robots by human demonstration was successfully implemented. The system proved to be extremely robust to variations in the environment light making it ideal for real-life applications.

The human operator demonstrated the manipulation of a tool for a stereoscopic vision system, and can see the path mimicked by the robot in real-time. Replaying the extracted points turns this into an immediate teaching machine. The real-time playback is also extremely useful for managing large or dangerous objects.

The use of easily detectable high-intensity and coloured light markers reduced the complexity of using this system by a non-skilled operator.

This system was completely built with standard components which reduces implementation cost, therefore enhancing the reconfigurability of manufacturing systems.

B. Future Works

For this paper a single marker was used to demonstrate the principle, therefore one could only control the tools position. Integrating three or more markers in the tool it will be possible to position and orientate the industrial robot's head.

The development of a path planning method will improve the robot's teaching phase by reducing the number of instructions executed by the robot.

Several precision studies will be done in the future in order to correct more precisely the projective distortions originated from the binocular vision.

A possible application for this technology in virtual reality is also being studied.

VII. ACKNOWLEDGMENTS

Paulo Malheiros acknowledges the Portuguese Science and Technology Foundation (FCT) for his PhD grant.

REFERENCES

[1] Osha Technical Manual, "Section IV: Chapter 4, Industrial Robots and Robot System Safety", http://www.osha.gov/dts/osta/otm/otm_toc.html, 2009.

[2] Robertz, Sven Gestegård and Henriksson, Roger and Nilsson, Klas and Blomdell, Anders and Tarasov, Ivan, "Using real-time Java for industrial robot control", *in JTRES '07: Proceedings of the 5th international workshop on Java technologies for real-time and embedded systems*, 2007.

[3] Yusuke MAEDA, Nanako ISHIDO, Haruka KIKUCHI and Tamio ARAI, "Teaching of Grasp/Graspless Manipulation for Industrial Robots by Human Demonstration", *in Proc. of 2002 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 1523-1528, 2002.

[4] J. Norberto Pires, Germano Veiga and Ricardo Araújo, "Programming-by-demonstration in the coworker scenario for SMEs", *in Emerald Industrial Robot*, 2008

[5] The Imaging Source, http://www.theimagingsource.com/, 2009

[6] 5dpo Component Library, http://wiki.lazarus.freepascal.org/5dpo, 2009

[7] Hartley, R. I. and Zisserman, A., *Multiple View Geometry in Computer Vision*, Second Ed., Cambridge University Press, 2004.

[8] Thorsten Thormählen, Hellward Broszio and Ingolf Wassermann, *ROBUST LINE-BASED CALIBRATION OF LENS DISTORTION FROM A SINGLE VIEW*

[9] Hartley, R. I., "In Defense of the Eight-Point Algorithm", *in Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 1997.

[10] Rimon Elias and Robert Laganière, "Projective Geometry for Three-Dimensional Computer Vision", *paper in Viva Research Lab.*

[11] Zhengyou Zhang, "A new multistage approach to motion and structure estimation by gradually enforcing geometric constrains" *in Lecture Notes in Comuter Science 1352*, Computer Vision - ACCV'98, Third Asian Conference on Computer Vi- sion, volume II, pages 567-574, 1998

[12] Real-time teaching videos, http://tinyurl.com/sincrovision, http://www.fe.up.pt/∼robotic/, 2009

This page is left blank intentionally

Light 3D Mapping for Search and Rescue Operations

Luís Lemos, Luís Paulo Reis

***Abstract*—Accurate construction of the map of a disaster area is the most critical step in allowing an autonomous robot to perform any task in an initially unknown scenario. Without an accurate map, the robot will not be able to successfully execute some elementary tasks like navigating. Some of the equipments currently used to produce maps are expensive, big and heavy while others may require a substantial processing overhead to find distances to obstacles in order to be able to build the map. This paper presents a light and simple 3D mapping methodology to achieve accurate 3D maps of disaster scenarios using a robot equipped with a few inexpensive sensors. The prototype of the robot and corresponding algorithms were tested in a simulated environment using the USARSim platform and a modified cut-down version of the P2AT simulated robot. The results obtained confirm that this methodology is suitable for 3D mapping.**

I. INTRODUCTION

In search and rescue performed by autonomous robots, the quality of the map used by the robot is of most importance. Not only does it serve the purpose of navigation, but can also be used to find victims and aid the rescue teams orientate themselves in the disaster area, among other things.

Initially, practically all maps were 2D for some reason or another. In some current applications, there are still 2D maps being used because the height is not necessary, like in an automated vacuum cleaner. For the majority of the real life applications, namely disaster areas, 2D might not hold enough information neither for the rescue crew nor the robot itself. With 3D mapping, potential victims can, possibly, be identified right from the mapping with little processing overhead involved.

Nowadays, there has been an increased usage of 3D mapping but some of the maps are produced using big, heavy, energy consuming sensors like the SICK LMS200 Laser Measurement Sensor (Fig. 1) [1] when compared to a simple IR sensor. Other sensors, like cameras, may require considerably more powerful processing units to process the measurements for distances in real time. These characteristics impose restrictions on the robot's size, autonomy and reach. Thus, in this work, there was an attempt to reduce the size of the robot and increase its autonomy and reach by using simple small sized sensors and still obtain accurate maps. Disaster scenarios were our choice for testing ground since they are the most complex scenarios that one can have due to the existence of many, mainly small, objects that may not be debris and need to be mapped.

Luís Lemos is with DEI-FEUP – Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, PORTUGAL (e-mail: luis.lemos@fe.up.pt).

Luís Paulo Reis is with DEI-FEUP – Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto PORTUGAL and LIACC – Laboratório de Inteligência Artificial e Ciência de Computadores da Universidade do Porto (NIAD&R – Núcleo de Inteligência Artificial Distribuída e Robótica) (e-mail: lpreis@fe.up.pt).

II. PROPOSAL

The 3D mapping methodology proposed includes the definition, implementation and testing of a robotic platform. The 3D mapping methodology must be able to generate accurate 3D maps of a disaster area and will be described in detail later in this document, explaining all the motives that led to that specific choice, which in turn impose restrictions on the chosen robotic platform. The robotic platform must use few simple sensors and the lightest computational algorithms possible. To demonstrate it, a robot was created in the simulation platform USARSim. It was also implemented a software robotic agent that emulates the robot's hardware architecture, implements the robot's artificial intelligence and controls the robot's body in USARSim.

The maps generated by the robot are 3D point clouds and provide a probability of occupancy for each of its points. To maximize accuracy, the probability of occupancy of each point was a combination of the measures of the sensors using sensor fusion methodologies. There were two sensor fusion methodologies used. In the first methodology, a simple dumb method was used in which the points are only marked as occupied or free, 100% or 0% for occupied or free points respectively, and the update of an existing point is the mean value of the old value with the new one. In the other sensor fusion methodology used, for each new measurement of a sensor, the points on the map were updated using probabilistic models based in the Bayesian method discussed later in this document.

In order to prove the accuracy of the methodologies, the maps generated using realistic sensors' models had the probabilities of all its points statistically compared against maps generated using ideal sensors' models by calculating the absolute error mean and the standard deviation of the absolute error. The most accurate maps are the ones that have all these values closer to zero since the difference between probabilities for each point will be smaller.

III. 3D MAPPING SENSOR TECHNOLOGY

Search and rescue robots use a large set of sensors to perform innumerus tasks like collision detection, mapping and localization. In this work, our interest falls specifically on sensors used for 3D mapping and sensors that could potentially be used for that purpose. Table I presents a list of the most relevant characteristics of some sensors belonging to the group of range sensors that are used for mapping, and Fig. 1. presents the respective sensors.

TABLE I
RANGE SENSORS' CHARACTERISTICS

Sensor		SICK LMS200	HOKUYO URG-04LX	Mesa Imaging SR4000
Dimensions (cm)	Length	15.6	5	6.5
	Width	15.5	5	6.5
	Height	21	7	6.8
Weight(g)		4500	160	470
Max Range(m)		80	4	5
View angle (º)	Horizontal	180	240	43.6
	Vertical	0	0	34.6
Range Precision (m)		0.01	0.01	0.01
Angle Precision (º)		0.25	0.36	0.23
Approximate Price (€)		7500	1850	7050

Fig. 1. SICK LMS200 (left), HOKUYO URG-04LX (centre) and Mesa Imaging SR4000 (right).

From Table I, we observe that the prices of the sensors are relatively high or very high. Also, apart from the SR4000, the sensors do not have a vertical angle which means that to enable measurements to be performed in a third dimension, the sensor must be mounted in a tilt or pan unit.

The Mesa Imaging SR4000 [9] is similar to a camera but instead of acquiring an optical image, each of its pixels measures the distance to the nearest non-transparent object. The result will be a map like the one presented in Fig. 2. where smaller distances correspond to lighter colours.

Fig. 2. An office chair (left) and as viewed by a SR4000 sensor (right).

Other methods of generating 3D maps exist, that use sensors that do not belong to the group of range sensors. One of those groups is the group of the image sensors. One of the methods that uses an image sensor is the stereo pair of cameras. This method is relatively complex and demands a great deal of computing power to be performed in real time with negligible delay. Fig. 3 shows the view from a stereo pair of cameras.

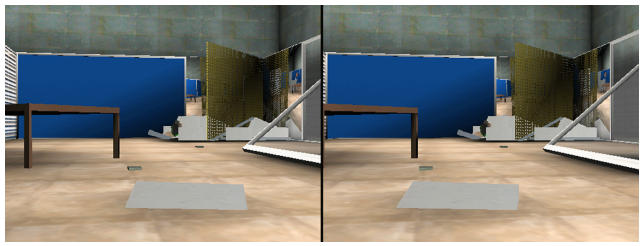


Fig. 3. View from a stereo pair of cameras.

Performing measurements from a pair of stereo images involves sophisticated algorithms of image processing to detect and match the objects from both images and to extrapolate the distances [8].

Currently, there are techniques of capturing the surroundings using just one camera. One of those techniques is by using an omnidirectional camera, like the one presented in Fig. 4. . It is basically a camera pointed at a mirror, usually hyperbolic, which increases the camera field of view and allows it to have a 360º horizontal by, almost, 180º vertical view of the surroundings.

Fig. 4. An omnidirectional camera sensor.

A view from an omnidirectional camera perspective is presented in Fig. 5. .Notice that the black circle at the centre of the image is the camera and that the objects at the edges of the image are stretched.

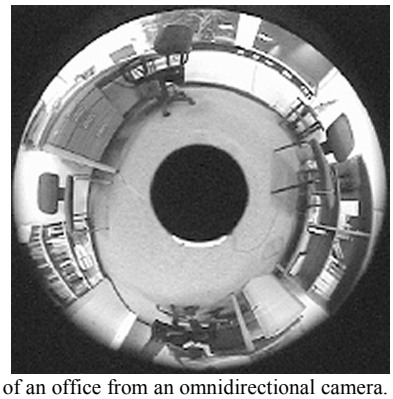


Fig. 5. View of an office from an omnidirectional camera.

As can be deduced from Fig. 5., to process the image, it is required the application of an algorithm that performs the inverse of the distortion of the image, which can represent a significant overhead of processing, depending

on the type of mirror and on the order of the equation chosen to approximate the effects of the mirror. Although relatively more complex to process and calibrate than the stereo pair of cameras, mainly because of the distortion of the mirror, a single omnidirectional camera can be used to measure distances to objects [10].

Still in the group of vision technology, and also using just one camera, is the fish-eye lens. This type of lens is shown on Fig. 6. (left) along with a photo taken with it (right).

Fig. 6. Fish-eye lenses (left) and a view through one (right).

Much like the omnidirectional camera, images taken using this technique are also more complex to process than images taken with the stereo pair of cameras due to the distortion of the lenses.

Returning to the group of range sensors, there are two basic sensors that, nowadays, are not usually used to measure distances but to detect if a collision is imminent. Those sensors are the sonar and the infra-red. Small dimensions, lightweight and low power consumption are some of the characteristics these sensors possess. Although they lack precision for great distances, specially in harsh conditions, it is not too affected at close range.

IV. Simulation Environment

To ease the implementation of this work, it was decided to implement it under a simulated environment. Our choice fell on the USARSim simulation platform which is the simulation platform used in the Virtual Robots Competition of the RoboCup Rescue [2]. The simulation platform is composed of a physics simulator, a graphics simulator and a robotics simulator.

The physics and graphics simulation is performed by Unreal Tournament 2004 (UT2004) from EPIC Games [3], one popular multi player 3D first person shooter that uses a rigid body physics simulator called Karma [4][5]. The UT2004 game engine also provides an Unreal Virtual Machine [6] which allows a user to create programs to interact with the game in a language called UnrealScript.

The robotics simulator is named USARSim and it is an UnrealScript application that runs on top of UT2004 to provide APIs, robot and scenario parts, sensors and actuators in order to facilitate the implementation and interaction with the robot, and the implementation of scenarios. Its combination with UT2004 results in a realistic simulator and the measurements returned by it are comparable in quality to those a real robot would return in a real environment under certain conditions like it is explained in [6].

V. Mapping Strategy

There are some characteristics that are desirable in a search and rescue robot. Accuracy of the maps is just one of those characteristics. Small dimensions, light weight and long range of action are other of the desirable characteristics since they will allow the robot to reach tight places, finding victims that a larger robot might miss, and imply a longer battery life using a small sized battery.

In order to reduce the size of the robot, the amount and size of the sensors and the size of the battery and actuators must be reduced. Reducing the size of the sensors and actuators also implies that the power consumed by them will potentially be smaller since the power consumption tends to diminish with the reduction in size.

In order to have the fewest sensors as possible, the best method is for the sensors to have some mobility instead of being statically placed on the robot. By installing them in a moving platform, one single sensor can replace several others and still return equivalent measurements. With an increased mobility allied to a sweeping tactic, it is possible for that single sensor to return even more measurements than all the others statically placed, making it possible to obtain a 3D map of part of the world.

By optimizing the placement of the sensor sweeping platform in the robot and carefully choosing the sensors that are installed in it, we can discard most of the other sensors like touch sensors or sonars.

The strategy adopted, that was thought to comply with the characteristics described earlier, was to use only one sensor of each kind, mount them in a Pan-Tilt-Zoom (PTZ) platform and place that platform in the frontal part of the robot with no other part of the robot in the way so that the PTZ can perform sweeps of, at least, half a sphere.

To produce the most accurate maps a sensor fusion algorithm should be used to combine all sensor measures in order to take all the individual errors of the sensors and minimize the final error.

VI. Implemented Architecture for Testing

A modified cut-down version of the P2AT, shown in Fig. 7. , was used for testing purposes since I lack the knowledge of 3D image editing to implement my own robot. The P2AT was striped of all its sensors except the ground truth, one sonar and one IR. The ground truth sensor remained in the physical centre of the robot's body and the two sensors left (IR and sonar) were moved to the PTZ platform. The PTZ base was placed in a lowered position in the frontal part of the robot so that the ground in front of the robot stayed in the sensors line of sight when the PTZ is pointing down. Also, due to that, the PTZ was not able to perform sweeps of half a sphere.

Fig. 7. The modified P2AT robot.

VII. Sensors and Sensor Models

A. In USARSim

In USARSim, the IR sensor is modelled as a straight line that starts at the sensor and ends when it intersects the first non transparent object. It then returns the length of the line affected by an error which in turn depends of the length of the line.

The sonar is modelled by many straight lines starting at the sensor, projecting themselves away from the sensor, inside its cone of action, and ending in the first object (transparent or not). After one pass of measurements, the length of smallest line is returned. Similarly to the IR sensor, the value returned by the sonar sensor is also affected by an error. It also has an angle that defines the sonar's cone of action. Table II presents the sensors characteristics used for this work.

TABLE II
Sensors characteristics used for this work.

Sensor name	Sensor property	Property value
Sonar	Maximum range	5
	Maximum error	0.05
	Angle of cone	0.0873
IR	Maximum range	5
	Maximum error	0.05

B. In Robotic Software Agent

Each measurement from the IR or the sonar sensors will be converted into a line or cone of points, respectively, were each one of those points will have a probability of occupancy associated with it and depends on the sensor's model of uncertainty. Keep in mind that a line can be understood as a special case of a cone where the angle of the cone is zero. This approach allows the same model of uncertainty to be used with both types of sensors.

The model of uncertainty used in this work will be the one represented in Fig. 8. [7] and it divides the sensor's action range in three regions. Region I is the region where the points are probably occupied. Region II is the region where the points are probably empty. Region III is the region where it is unknown if the points are occupied or empty.

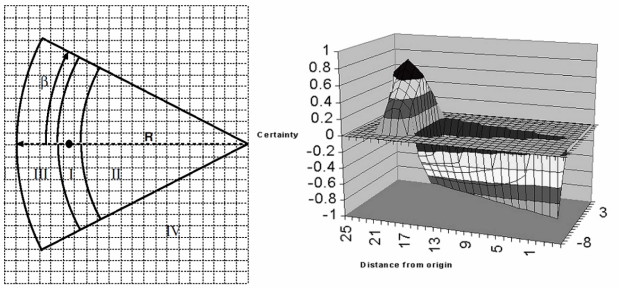


Fig. 8. Representation in 2D (left) and 3D (right) of the sensor model of uncertainty extracted from [7].

If a sensor measures the distance A and the sensor model states that, for that distance, the maximum error is E, then the limit that separates Region I from Region II is at a distance of A-E from the sensor, and the limit that separates Region I from Region III is A+E.

The probability of each point will be calculated using one of two methods. The first is the dumb method in which the sensor model presented will be applied to each measure from the sensor but the error will always be zero, meaning that Region I will be constituted solely by the points that present themselves at the measured distance and will be given the probability of 100% occupied. All other points will be given the probability of 0% occupied. If any of those points already exists in memory, their probabilities will be updated by performing the mean value of the currently given probability with their old value of probability of occupancy.

For the second method, the Bayesian method, the formulas used for the calculation of the probabilities are (1) to (3) and belong to regions I to III of Fig. 8., respectively.

$$P(occupied) = \frac{\frac{R-r}{R} + \frac{\beta-\alpha}{\beta}}{2} \times max_{occupied} \quad (1)$$

$$P(occupied) = 1 - \frac{\frac{R-r}{R} + \frac{\beta-\alpha}{\beta}}{2} \quad (2)$$

$$P(occupied) = unknown\,;\, do\,nothing \quad (3)$$

Where R is the sensor's maximum range, r is the measured distance from the sensor to the point being calculated, β is the beam half angle, α is the angle between the line that connects the centre of the sensor to the point being calculated with the line that defines the centre of the cone, and maxoccupied is a pre-defined value that defines the assumption that a point is already occupied before any measurements are made. In the special case of the IR sensor, since there is no β nor α, the term $(\beta-\alpha)/\beta$ in the above formulas is zero.

After calculating the probability of a measured point

using formulas (1) to (3), the probability of each point in the world is updated using formula (4)

$$P_{world} = \frac{P \times P_{-1}}{P \times P_{-1} + (1.0 - P) \times (1.0 - P_{-1})} \quad (4)$$

Where P is the probability of the measured point and P_{-1} is the previous probability of that point in the world. If there is no previous point (P_{-1}) then maxoccupied is used instead.

VIII. Post-Process for Result Analysis

Because most points will not match, due to errors in measurements using non-ideal sensors, a method was devised to create the missing points in a manner that the calculated points would not diverge much from the values of probability they would have if they had been measured. That method is to calculate the value of a missing point by performing a weighted mean of the closest points surrounding him. Formula 5 presents the equation used to calculate the weighted mean.

$$M = \frac{\sum_{i=1}^{n} w_i * v_i}{\sum_{i=1}^{n} w_i} \quad (5)$$

Where M is the mean value resulting from the calculation, n is the number of points, vi is the value of the current point and wi is the weight attributed to the current point. In this formula, the weight attributed to a point was calculated as being the distance of the current point to the missing point, subtracted to the maximum distance of all points being used to the missing point, as presented in Formula 6.

$$w_i = max(d_1, d_{2,\ldots}, d_n) - d_i \quad (6)$$

Performing the subtraction to the maximum ensures that points further away from the missing point have smaller influence on the result.

IX. Statistical Comparison

In order to test the accuracy of the measures, a set of points is statistically compared with a reference set of points through the comparison of the mean of the absolute error of the probability and the standard deviation of the absolute error of the probability. The closer to zero these values are, the more accurate the measured set of points is.

For each point in the target point cloud, the same point is retrieved from the reference point cloud and the absolute value of the difference of the probabilities is calculated, obtaining the absolute error of the probability. After calculating the absolute error, the mean value and the standard deviation value of the absolute error can be calculated using the usual formulas for the mean and standard deviation. By performing these calculations for all the intended point clouds, one can compare the values of the different means of the absolute error and the different standard deviations of the absolute error to have an idea of how accurate is each one of the point clouds relative to the other.

X. Reference Point Clouds

The required reference set of points was generated by performing measures of the scenario with ideal sensors, meaning sensors without noise. After being measured, the reference set of points was filtered to keep only those points that had a probability of occupancy greater than 90%. Since the sonar does not return good enough results to be usable as a reference, only the IR measurements were used. After having the set of points, the probabilities of the remaining, unfiltered, points were updated to read 100% occupancy. Fig. 9. presents the final appearance of a reference set of points for a position of the robot in the USARSim Yellow arena and Fig. 10 presents that position.

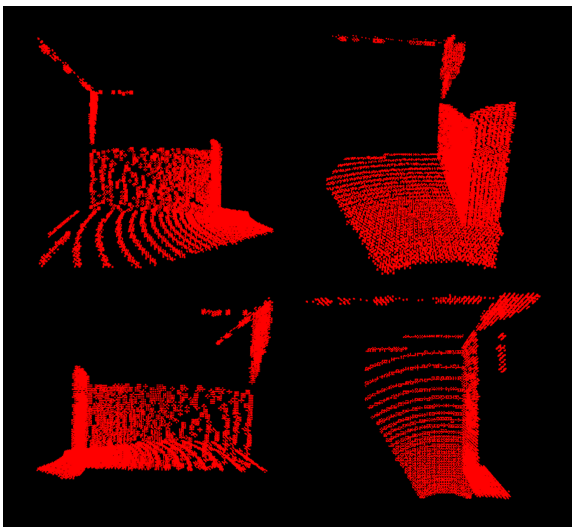


Fig. 9. IR reference point cloud after post-processed as seen from the left (upper left), front (upper right), right (lower left) and up (lower right).

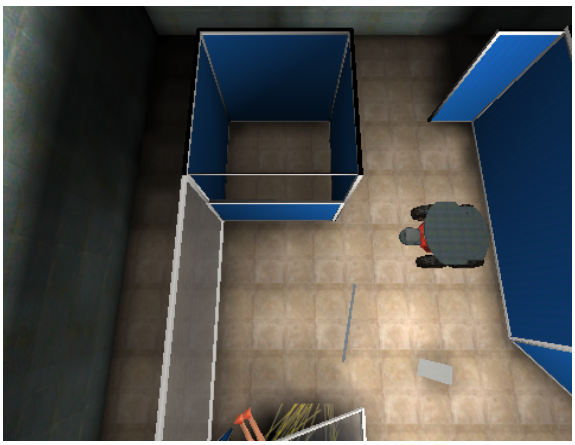


Fig. 10. One of the positions used for testing.

XI. Results analysis

Having the set of references, the actual experiments with non-ideal sensors were performed. After obtaining the measurements, the point clouds were filtered to remove all

probabilities below 90% because the remaining 10% of the points are the most interesting ones since they represent the actual obstacles. Fig. 11. presents the IR measures belonging to the Bayesian method.

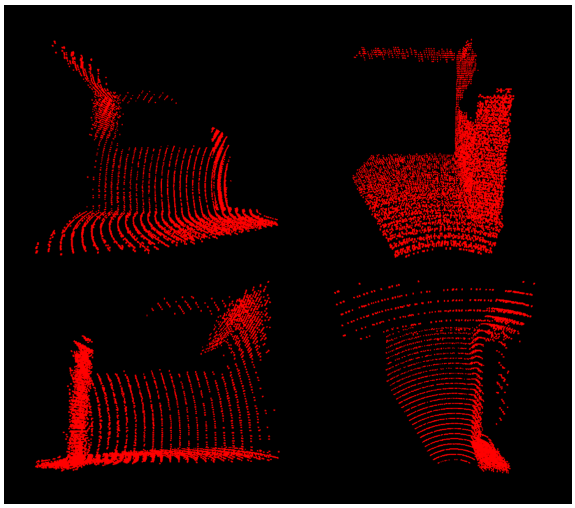


Fig. 11. - IR point cloud obtained using the Bayesian method as seen from the left (upper left), front (upper right), right (lower left) and up (lower right).

The resulting sets were statistically compared with the reference point clouds and some of their corresponding statistics are presented in Table III.

TABLE III
STATISTICAL COMPARISON

	Absolute Error Mean			Absolute Error Standard Deviation		
Algorithms	IR	Sonar	Fusion	IR	Sonar	Fusion
Dumb	0,152	0,306	0,267	0,192	0,262	0,237
Bayesian	0,144	0,281	0,243	0,190	0,253	0,226

XII. COMPARE RESULTS

As expected, the sonar yields worst accuracy than the IR, probably due to the angle of the cone being so large, disallowing it from having good accuracy. As a consequence of the sonar's accuracy and the fusion methodology implemented, the fusion of both IR and sonar sensor measures also yields worst accuracy than the IR alone. These results should not discourage the usage of sensor fusion since, with it, it is possible to obtain extra information, like the presence of transparent surfaces, that is not possible to obtain with the IR alone. In fact, the analysis of the results indicates that the sonar has a standard deviation smaller than the IR for scenarios where the majority of the objects is closer. This means that, although the mean of the absolute error is larger, the maximum deviation a measure can have is smaller.

XIII. CONCLUSIONS AND FUTURE WORK

Due to the generation of an accurate map being so important, investigators are always experimenting new sensors or new algorithms that are able to improve the accuracy of the generated maps. Sensors are getting smaller, lighter, more energy efficient and more accurate. Some sensors already return a 3D map of scenario in front of it without the intervention of any moving part, like the SR4000 described earlier.

In this document, a methodology for mapping is described that generates accurate 3D maps of a disaster scenario. It is composed of a methodology to obtain measurements, a hardware platform and a group of algorithms that, in conjunction, complement each other. A list of constraints aids in the optimization of the robot to achieve its full potential.

The most basic range sensors are used and it is proven by the results that their performance and accuracy is good although the speed of generation of the map is a little slow because of the amount of points required to measure and the scanning method adopted. Anyway, this could be minimized by having more than one robot performing measurements in different parts of the scenario and sharing each others maps.

There is still some work to be done in the future to make this work usable in a real search and rescue scenario. One of the tasks of the future work is the implementation of an object detection algorithm in order to enable the robot to store objects in memory instead of points, possibly saving memory since an object might be defined by a large set of points.

ACKNOWLEDGMENTS

This work has been supported by project ACORD – Adaptative Coordination of Robotic Teams (FCT/PTDC/EIA/70695/2006).

REFERENCES

[1] Stahn, R.; Heiserich, G.; Stopp, A., Laser Scanner-Based Navigation for Commercial Vehicles, Intelligent Vehicles Symposium, 2007 IEEE, 13-15 June 2007 Page(s):969 - 974

[2] Virtual Robots Competition of the RoboCup Rescue , online, available at: http://www.robocuprescue.org/virtualsim.html [consulted on December 2008]

[3] Epic Games , online, available at: http://www.epicgames.com/ [consulted on December 2008]

[4] Unreal Engine 2, online, available at: http://www.unrealtechnology.com/html/technology/ue2.shtml [consulted on December 2008]

[5] MathEngine Karma User Guide , online, available at: udn.epicgames.com/Two/rsrc/Two/KarmaReference/KarmaUserGuide.pdf [consulted on December 2008]

[6] Carpin, S., Lewis, M., Wang, J., Balakirsky, S., Scrapper, C.: USARSim: a robot simulator for research and education. IEEE International Conference on Robotics and Automation (2007)

[7] Murphy R. : Introduction to AI Robotics. The MIT Press, Massachusetts (2000)

[8] Iocchi, L., Konolige, K., Bajracharya, M.: Visually Realistic Mapping of a Planar Environment with Stereo. Proc. of Seventh International Symposium on Experimental Robotics (2000)

[9] SR4000 sensor, online, available at: http://www.mesa-imaging.ch/prodview4k.php [consulted on December 2008]

[10] Millnert, O., Goedemé, T., Tuytelaars, T., Van Gool, L.: Range Determination For Mobile Robots Using One Omnidirectional Camera. International Conf. on Informatics in Control, Automation and Robotics, ICINCO 2006, 1-5 August 2006, Setubal, Portugal.

European Land Robotic Trial 2008 (ELROB) - A Realistic Benchmark for Outdoor Robotics

Bernd Brueggemann, Timo Röhling, Hans-Ludwig Wolf and Frank E. Schneider

***Abstract*— The European Land Robotic Trial (ELROB), which was held for the third time in 2008, is designed to compare unmanned ground vehicles in realistic outdoor tasks. It addresses the need to create a benchmark that can reproducibly compare and evaluate different robot systems. While robot trials like the DARPA Grand Challenge or the RoboCup have proven to be adequate benchmarks to compare robots systems in specific scenarios, the ELROB provides benchmarking in a wide range of tasks which are oriented at prospective use-cases in both military and civil applications.**

In this paper we describe the ELROB 2008, the rationale behind the scenario design and how the trial has been implemented. Further, we present the results which illustrate the remaining gap between requirements and abilities.

I. INTRODUCTION

The European Land Robotic Trial (ELROB) is designed to demonstrate and compare the capabilities of unmanned systems in realistic scenarios and terrains. It was invented by the European Robotics Community and organised by FGAN [9]. The trial is held annually, alternating between a military and civilian focus. The first military ELROB (M-ELROB 2006) was performed at the military school in Hammelburg, Germany. In 2007 a civilian ELROB (C-ELROB) took place in Monte Ceneri, Switzerland. The latest ELROB, the M-ELROB 2008, was held again at the military school in Hammelburg. The aim of each ELROB is to get a deep insight into the field of ground robotics by testing existing solutions in practical trials. These trials

- are conducted with a focus on short-term realisable robot systems,
- are explicitly designed to assess current technology to solve real world problems at hand, and
- are an opportunity to bring together users, researchers and industry to build a community.

The ELROB scenarios do not limit themselves to the abilities of today's robots, but focus on realistic missions designed by prospective users in a demanding environment. The challenges in 2008 have become much harder than in previous ELROBs, reflecting the new requirements on ground robotics defined by the German Army. Thus, it was expected from the beginning that not all participants would be able to complete the scenarios.

This paper should give an insight both into the designing of the different trials as into the performance of the participants. The remainder of this paper is organized as follows: We present some reasoning why a trial is an adequate benchmark for robots, and why benchmarks are important to the robotics community. In section II the tracks and tasks for the participants are described. Section III briefly presents the participants and their robots. In section IV the results of the ELROB are discussed. The paper closes with our conclusions and future work.

B. Brueggemann, T. Röhling, H.-L. Wolf and F.E. Schneider are with the Research Group "Unmanned Systems", Research Institute for Communication, Information Processing and Ergonomics Research Establishment for Applied Science (FGAN), 53343 Wachtberg, Germany {brueggemann,roehling,wolf,schneider}@fgan.de

A. Related Work

Generally it is a difficult task to compare different published approaches in the field of robotics[1]. Thus robot competitions are recognized as valuable benchmarks for real robot systems [2]. Several different competitions were held in the last years. Two of the largest and best-known competitions are the RoboCup [4] and the DARPA Grand Challenge [3], which are also recognized outside the robotics community.

While the RoboCup is currently targeted at indoor robots, the the DARPA Grand Challenge aims to test and compare driverless cars. It started in 2004 with the rather simple task of following a 241 km long path, defined by several thousand GPS waypoints. Due to the difficult terrain and some teething problems, no participant was able to solve this task. In 2005, the task remained basically unchanged, and four participants successfully completed the race. In 2007, the DARPA Grand Challenge modified its goals from driving autonomously on difficult terrain to interacting with other vehicles in an urban scenario. Again this challenge could be solved by three teams.

The ELROB is somehow comparable to the DARPA Grand Challenge in its attempt to gauge the functionality of outdoor robots. However, the ELROB presents a variety of tasks instead of a single scenario, and generally puts less emphasis on competitive rankings. Thus, the participating teams are encouraged to explore the limits of their systems.

II. TRACKS AND TRIALS

The chosen area for ELROB 2008 lies within the training facility of the military school in Hammelburg. Its size is of about 9 square kilometres. The accessible roads have different qualities, ranging from well paved to heavy dirt roads. The environment is predominantly woody (see Fig 1).

The different tracks on site were chosen to test specific aspects of robot deployment. Some challenges were common to all tracks, others were specific to certain scnearios. In preparation for the trials, every track was tested with respect to

- accessibility of the roads,
- GPS reception, and
- radio reception between track and control station.

By selecting areas with an elevation profile that does not support continuous radio communication from the control station, a certain level of autonomy was enforced. Thus,

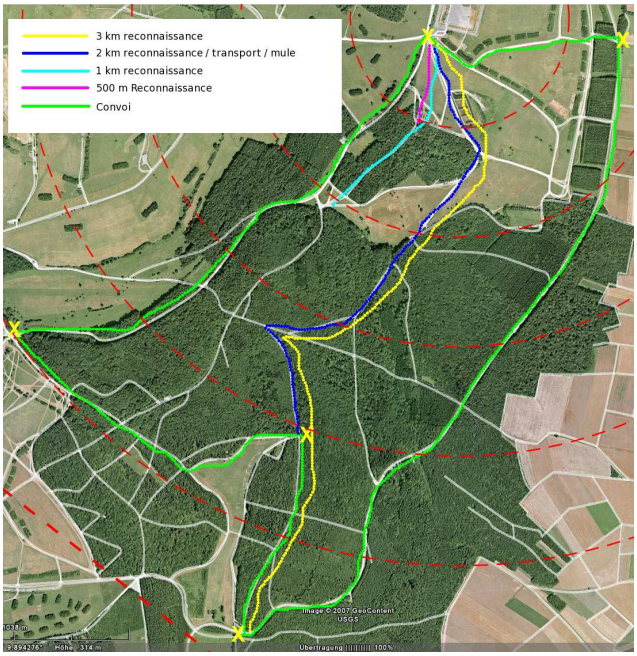


Fig. 1. Overview of the different tracks

it was deliberately made very difficult or even impossible to complete the missions in a purely remote-operated way. In figure 1 one can get an overall impression of the whole area. The different colours mark the different tracks of the missions. The camp-site which had to be explored in the reconnaissance mission is located at the uppermost yellow cross. This was also the starting point for the transport missions. The point farthest away from that, the lowermost yellow cross, is about 3000 metres away. In the following subsections, each mission and track is briefly described.

A. Reconnaissance Mission (Day/Night)

The objective of this mission was to approach and explore a camp-site. The participants had the choice to start either 500 metres, 1000 metres or 3000 metres away from the destination. The latter two starting points were located in the woods and had no direct line of sight to the camp. Upon entering the camp-site, the robot had to find several marked vehicles, take a picture of them, and acquire their GPS coordinates. The positions were to be marked on a map. Additionally, if the robot had started 500 metres away from the camp, it had to return to the starting point. The time limit was 45 minutes for all participants starting 500 metres away, and 60 minutes otherwise. The tracks were different for each starting point, however the last 500 metres were always identical. The participants were taken to their respective starting point by transport, but received no map and were not told their location.

The area for the 500 metres can be seen in Fig. 2. `500m1`, `500m2`, and `500m3` were possible starting points for the participants who decided to start at the 500m distance. Between them and the camp-site there was open area with rather long grass. Participants who decided to try the longer distances, were expected to arrive from the bottom right (direction south-east). Because of the abundantly covered area (waist-deep grass) the surface was not easy to figure out. Especially trenches and big stones

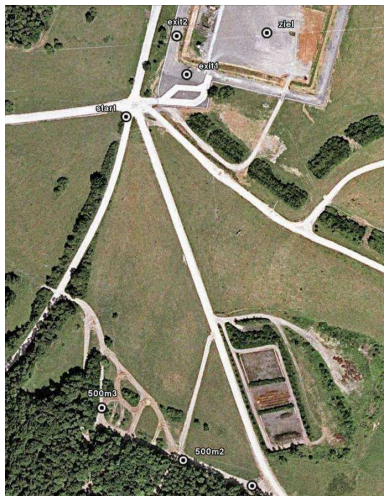


Fig. 2. Close view of the last 500 metres of the reconaissance mission

were hidden by the grass. The participants knew of the existence of such obstacles, but not their location.

Because the task should be accomplished as realistically as possible, the approach should be stealthy. Most of the roads were declared as mined, so the robots were not allowed to use them. Only a small part of the road near the camp was permitted to be used, because there was no other way to enter the site.

The mark `Ziel` was given to the participants as GPS waypoint. The labels `exit1` and `exit2` mark the exits from the camp-site, which were unknown to the participants in advance. The camp-site itself was surrounded by a meshed wire fence with only two entrances, a security entrance and a gate. Both entrances were big enough to let every robot pass through.

B. Convoy Transport Mission

For this mission, a convoy of at least two vehicles had to be formed, and at most one vehicle was allowed to be manned. The convoy had to drive a distance of about 20 km. While on track, the convoy encountered dynamic as well as static obstacles. The path was defined by a minimal set of GPS waypoints which were quite far from each other, so the robot could not just drive straight lines between the waypoints but had to navigate along the road. There were two blockades on the track, which made a detour necessary. The roads were part of the local testing grounds for trucks, usually gravelled and leading mainly through the forest. They were not marked, so there was no clear distinction from the surrounding terrain.

C. Mule Transport Mission

In this mission, the participants were asked to establish a shuttle service between two points approximately 2 km apart. The participants were allowed to accompany their robot on the first run between those two points. After that, the robot should repeatedly shuttle in between. The only information for the participants was the position of the turning point. The goal was to travel the way between the two points as often as possible within 60 minutes.

The track was similar to the one in the reconnaissance mission, with dirt roads, wooded areas and some open

spaces. The straight connection between starting point and turning point was not the best choice with respect to obstacles and difficulty of the ground. As there were several possible ways, the robot could select the one most suitable to its own abilities.

D. Security Mission

A camp-site environment should be monitored by one or more robots. Possible intruders had to be found and reported. This was to be done by taking a picture and acquiring the GPS position of the intruder. The entire security mission lasted 30 minutes for each participant.

During the task, the robots had to stay inside the camp-site which was partly surrounded by mesh wire fence. The robots were unable to overlook the complete camp from any single point of view, so they had to move around to detect the intruders. The camp consisted of three different sections:

- Free area at the east part
- Barracks
- Area in front of the barracks

The free area in the east of the camp was surrounded by mounds on three sides. The border of the fourth side was built by the front of the barracks and some containers. The ground consisted of short grass and grit. Most challenging was the barracks area with its chessboard-like building positions and the adverse lighting conditions that could confuse automatic picture analysing algorithms. Nine short pathways connected the free area in the east part of the camp with the area in front of the barracks. This was a relatively uncluttered area with six bigger buildings on it. Those buildings were at the far west end of the area. There was enough space between them to clearly see what was going on. This area was covered with grit.

III. PARTICIPANTS

The ELROB 2008 participants were both from industry and universities, 14 teams in total. We will briefly introduce each team's robot in this section, additional information (if published) can be found in the cited literature.

The *University of Siegen* sent their team with the robot *AMOR* into the trials. AMOR is a modified quad equipped with laser line scanners, PMD cameras and a stereo camera system. It is able to drive autonomously, e.g. following a person or finding given waypoints[5]. The team from *University of Hannover* entered the trial with the robot called *Hanna*. Based on an off-the-shelf transport car, it is equipped with 3D laser range scanners and cameras. It is capable of driving autonomously, for example to find given waypoints. The *University of Kaiserslautern* attended with their Robust Autonomous Vehicle for Off-road Navigation, in short *RAVON*. It is able to move fully autonomously, driven by a behaviour-based control system. It uses three laser range scanners, cameras and several additional sensors like GPS or a magnetic field sensor[6]. The *University of Oulu* brought their robot *Mörri* with them. Mörri is a six wheel robot and was the smallest ground robot in the field of participants. Mörri is build for heavy terrain and large payload. It is controlled semi-autonomously with an operator as guide. The Team *MuCAR* from the BW-University of Munich developed and operated the robot *MuCAR-3*. It is a modified Volkswagen Touareg with extended autonomous capabilities. Its primary sensor is a camera system that resembles the human vision system. The *Jacobs Robot Team* from Jacobs University Bremen attended with the robot *Rugbot*. It is a small track robot designed to pass difficult environment. It can be equipped with various sensors, but relies primarily on laser range scanners and ultrasonic sensors [7].

Team *Base10* brought their robot *Gecko TRS* with them. It is a four wheel vehicle of about 3000 kg. Its speciality is its high manoeuvrability, because of its four separate steerable wheels. *Diehl* started with their robot *CANGARU*, a four wheel car, which is capable of carrying one human. CANGARU, short for Compact, autonomously navigating ground unmanned robotic unit, is equipped with a 360 degree CCD and an infra-red camera. *AirRobot* was one of two teams which entered the trial with an aerial robot only. The *AR100-B* is a quad rotor flying vehicle that is controlled completely manually. Thanks to its self-stabilizing abilities it is easy to handle. *The French Team* is a cooperation of THALES and ECA. They announced three different systems to take part in ELROB. First the *Rtrooper*, a large six wheel robot, second the *PRM*, a mini UGV on track and equipped with flippers, and last the *SPY'Arrow*, a light-weight UAV. *Wiesel2 digital* and *Trobot* were the robots presented by *Rheinmetall Defence*. The first one is a modified Wiesel2 track platform from the German Army. The second one is a eight wheel off-road vehicle. Both robots were mainly manual-controlled. The company *Telerob* presented the abilities of their robot *teleMAX*. It is a track robot with flippers and a robotic arm. It is equipped with several cameras and able to climb stairs. The *EyeRobot* from *TNO* provides a special interface to its controller. It sends a 3D video and 3D audio stream to the operator. With a special helmet, the controller sees and hears what the robot does [8]. The *SR-H3* is an aerial vehicle developed by *Siralab*. This aircraft is a basically a flying wing with an autopilot to reach given waypoints. Its range is about 15 km and it can remain airborne for about one hour.

IV. RESULTS

A. Evaluation

The evaluation concentrated on parameters which were clear to distinguish and to measure. The parameters and weights measured for each trial can be seen in table I. For each mission the following parameters were considered:

- *Reconnaissance Mission day/night*
  degree of autonomy, number of found objects, total time
- *Security Mission*
  degree of autonomy, number of detected and checked intruders, time until detection of first intruder
- *Transport Mission Convoy*
  degree of autonomy, average speed, time needed for the road block bypass
- *Transport Mission Mule*
  degree of autonomy, average speed, successfully driven distance

The offical ranking could also be downloaded at [10].

TABLE I

WEIGTH OF EACH MEASURED PARAMETER FOR EACH MISSION

Mission	autonomy	detected objects /intruders	running time	speed	bypass blockade	turning points reached	distance	time until detection
Reconnaissance Day 1000m	10000	1000	100	-	-	-	-	
Reconnaissance Day 500m	100	10	1	-	-	-	-	
Reconnaissance Night 3000m	10000	1000	100	-	-	-	-	
Reconnaissance Night 500m	100	10	1	-	-	-	-	
Convoi Mission	100	-	-	10	1	-	-	
Mule Mission	100	-	-	-	-	10	1	
Security Mission	4	3.5	-	-	-	-	-	2.5

B. Reconnaissance Mission (Day)

UGVs do have the possibility to gain information about a situation while staying low on the ground and remaining undetected. Their ability to infiltrate deep into infrastructure and view objects from a short distance are key advantages in contrast to UAVs. Having these facts in mind, participating UGVs and UAVs had to face following tasks:

- Results of exploration must be obtained as fast as possible (best is real-time). So mobility and speed are important
- Results of exploration have to be exact and widespread to give a good insight into the situation. Therefore a good picture resolution together with exact positions are important
- As robots have to be reusable, detection by third parties has to be avoided. So small, quiet robots and the use of passive sensors are preferable

No participant could present a complete success. For example only the UAV from AirRobot was able to retreat from the camp in time. Every other robot had to be removed from the mission after time was up. Further only two systems, the AirRobot and the robot of Rheinmetall, had acquired the exact position of some targets. From eleven robots altogether three robots were dropped out, two because of problems with communication and the Rtrooper because of an accident. The robot had broken its wheel suspension while driving into a trench at high speed.

All participants started from the 1000m distance or the 500m distance to the camp-site. While every robot had to find the same amount of marked objects inside the camp, the locations of these objects changed for each run. The ranking system was built to ensure that participants starting from a longer distance outranked those starting from a shorter distance. This accounts for the large gap in the point rankings. The robots from the Universities Hannover and Kaiserslautern had driven with a high degree of autonomy. Due to this fact, they were the leading teams in this mission. Base10 and Diehl are placed second because they chose to try the long distance (see table II).

TABLE II

RESULTS: RECONNAISSANCE MISSION

Team	System	Points	Distance
1. University of Kaiserslautern, DE	RAVON	10100.00	1000m
2. University of Hannover, DE	Hanna	300.00	1000m
3. Base 10, DE	Gecko	150.00	1000m
3. Diehl, DE	CANGARU	150.00	1000m
5. University of Oulu, FIN	Mörri	Failed	1000m
6. AirRobot, DE	AR100-B	111.00	500m
7. Rheinmetall Defence, DE	Wiesel2	93.19	500m
8. TNO, NLD	EyeRobot	88.19	500m
8. University of Siegen, DE	AMOR	88.19	500m
10. French Team, FR	RTrooper	Failed	500m
10. Telerob, DE	teleMAX	Failed	500m

C. Reconnaissance Mission (Night)

The track and the rules were the same as in the day mission. Every robot that had reached the camp in daylight could participate in the night trial. The Wiesel2 from Rheinmetall had reached the camp and discovered some objects, but not within the time limit. In total, only four robots were qualified, but additional robots were allowed to start in the night mission without being evaluated. Telerob and Rheinmetall were willing to do that. The results of the night run can be seen in table III.

The circumstances were difficult in all respects except the weather, which was a mild and calm summer night. Again no participant could solve the entire mission. Only two teams were able to enter the camp (University of Kaiserslautern and AirRobot). Additionally, AirRobot was able to detect some objects but without any GPS position information.

University of Hannover was the only participant willing to try the 3000m distance, assuring the first place if the robot would not break during the run. Although the communication was lost several times, the robot could keep driving. Shortly before time ended, the robot stopped after driving almost 2400 metres, mostly semi-autonomously.

The robot of the University of Kaiserslautern drove into the camp fully autonomously, but did not explore the objectives, because that could not be done with the current implementation. Still, the presented level of autonomy was very impressive.

The other systems were remotely operated, so they could not gain any points for autonomous operation. The UAV AR100 (AirRobot) showed at least a semi-autonomous flight back to starting position. Altogether, the flight of the UAV demonstrated that such systems can operate very stealthy at night. The performance was mostly dedicated to the operator, because of his high level of experience with this type of robot.

TABLE III
RESULTS: RECONNAISSANCE MISSION NIGHT

Team	System	Points	Distance
1. University of Hannover, DE	Hanna	10100.00	3000m
2. University of Kaiserslautern, DE	RAVON	100.83	500m
3. AirRobot, DE	AR100-B	2.05	500m
4. University of Siegen, DE	AMOR	1.66	500m

TABLE IV
RESULTS: CONVOY TRANSPORT MISSION

Team	System	Points	Rank
Base10 DE	Gecko	DNS	
Diehl, DE	CANGARU	74.55	3.
University of Hannover, DE	Hanna	111.00	1.
University of Siegen, DE	AMOR	27.82	4.
BW-University Munich, DE	MuCAR-3	84.88	2.

TABLE V
RESULTS: MULE TRANSPORT MISSION

Team	System	Points	Rank
Base10 DE	Gecko	3.10	5.
Diehl, DE	CANGARU	2.90	6.
Rheinmetall Defence, DE	TROBOT	Failed	
University of Hannover, DE	Hanna	25.90	2.
University of Kaiserslautern, DE	RAVON	100.60	1.
University of Oulu, FIN	Mörri	3.35	4.
University of Siegen, DE	AMOR	13.5	3.

D. Convoy Transport Mission

Beside the demanding track length of about 20km, the main challenge in this mission was to bypass the road blockade. Additionally the participants had to face other difficulties like changing road conditions from paved road to dirt and back. Although the weather was fine and dry, only the larger robots with some off-road abilities were able to participate.

Because of the long distance, a high average speed was necessary. As one manned vehicle was allowed in the convoy, most teams decided to let one manned vehicle lead and their robot follow. All participants started with exactly two vehicles, avoiding the additional complexity of longer convoys. Five participants applied for this mission, but no robot fulfilled the mission entirely (see table IV for results).

The driven distance (without detour) was between 250m (Bw-University of Munich) and almost 10km (University of Hannover). The average speed ranged from 4 km/h to 7 km/h. This shows that the ability to follow a vehicle autonomously in difficult terrain is still far from stable. Surely there is much more progress in driving autonomously in cities. Especially the Bw-University of Munich has been very successful in this task for many years. In spite of that, the MuCAR-3 lost contact to the leader vehicle within 250m due to the bumpiness of the track and the limited vertical field of view of the tracking sensor, proving that this scenario is much more demanding than the pursuit on a paved road.

Once more the combination of a series production vehicle with high quality sensors delivered good results; the University of Hannover was able to drive almost one round of 10km including the (remotely controlled) avoidance of the road block.

The Bw-University of Munich ended up on second place because of their high level of autonomy. The mileage was no criterion, because when planning the track, the jury assumed that everyone would complete at least one round of 10 km. MuCAR-3 was allowed to retry the scenario and then showed its abilities in autonomous driving. Diehl would have placed second if their leader vehicle had not lost orientation and lead the robot into truly impassable terrain. The robot of the University of Siegen often lost connection to the leader and therefore only a very low average speed was possible.

E. Mule Transport Mission

This mission presented two main challenges. In the first part of the mission, the robot had to figure out how to move from one point to another in an unknown environment. The robot had to find the best route to shuttle between those two points, which was not necessarily the shortest path. In the second part of the mission, the discovered path had to be driven as often as possible.

One of the seven participants dropped out right in the beginning: Trobot from Rheinmetall broke at the starting point. The distance made by the other robots ranged from 0.8 km (Diehl) to 2 km (Uni Siegen). Only the robot of the University of Siegen actually reached the turning point. However, on its way back, with about 13 minutes of time remaining, it had to be stopped by emergency button after 50 metres because of uncontrolled direction changes towards the forest. So the autonomous functions could not be presented very well and therefore were not easy to evaluate. University of Kaiserslautern won by a clear margin because their robot already showed a very impressive autonomous driving during the first part of the mission. The other participants did the first part of the mission remote-operated (BASE10, Diehl and Oulu) or by following a leading person (University of Siegen), which was permitted by the rules.

Like the team Kaiserslautern, the University of Hannover chose to let their robot act almost completely autonomously. That is the reason for their second place. Additionally they achieved the second longest distance (1.8 km), with the robot being in close range of the turning point when time was up. The University of Siegen earned the third place because they travelled the longest distance (see table V).

F. Security Mission

This mission was not about actively securing a military camp. In such a situation an approach with stationary sensors, dogs, and security personal could be much more appropriate. The intent of this scenario is to secure mobile facilities or close security gaps within large military areas (like airports). Important is, beside the abilities a mobile sensor platform can offer, that the UGV can follow and identify persons (and vehicles).

To keep the runs of the robots comparable, each participant faced the same number of intruders. One intruder

TABLE VI
RESULTS: SECURITY MISSION

Team	System	Points	Rank
AirRobot, DE	AR100-B	2.00	5.
Diehl, DE	CANGARU	4.06	3.
French Team, FR	PRM	2.00	5.
Siralab, IT	SR-H3	2.00	5.
Telerob, DE	teleMAX	7.88	2.
TNO, NLD	EyeRobot	2.00	5.
University of Bremen, DE	Rugbot	2.00	5.
University of Oulu, FIN	Mörri	8.00	1.
University of Siegen, DE	AMOR	3.3	4.

had a passport to legitimate himself. All other intruders could not identify themselves. When an intruder was encountered by a robot, the person was cooperative and did not try to escape. Important for evaluation was the number of detected intruders and their corresponding positions. Other criteria included the time until the first intruder was detected and the level of autonomy. The environment was too cluttered to be reliably secured by a single robot. The combination of an UAV to overlook the site from above and an UGV to follow and stop intruders was an obvious solution and was employed by two participants. In total, nine teams participated (for results, see table VI):

The University of Bremen used a "Mikado" for this mission. All UAVs had big trouble with the choppy wind. Diehl could only run one robot out of two. The French team wanted to start with their UGV and an UAV. But due to the weather conditions the use of the "SpyArrow", a very fragile Styrofoam glider, was not possible. The UAV from Siralab started outside the camp and flew about 8 minutes on a pre-programmed path. Telerob used two robots: the well known tele-operated Telemax and another one equipped with a complex environment recognition provided by the University of Freiburg. The Jakobs University of Bremen used two UGV, called RugBots. The Universities of Oulu and Siegen, and TNO used one remote-operated robot each. TNO lost communication during the run. The University of Siegen used an additional UAV.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The purpose of ELROB was not to get an overview over technological possibilities but to test outdoor ground robots in real world scenarios without regard to limitations of the robots. Thus, the scenarios were to show the gap between desired and possible applications for today's robots. As could be expected, not every participant could cope with the designed missions. So the results were not unexpected and definitely not disappointing. In retrospect, two main problems could be singled out:

- reliable Hardware, including reliable communication
- innovative autonomous software controller

It was noticeable that while the industry generally had hardware in excellent quality available, they lacked the innovative autonomous control algorithms developed by the university teams. On the other hand, the University teams had most of their problems due to their restrained hardware budget and the required trade-off between functionality and cost. The combination from the robots used by industry and the control algorithms of university might achieve much better results.

B. Future Works

From the 15th to the 18th of June 2009 the second civilian ELROB will take place in Oulu, Finnland. It is titled with "Robotics in security domains, fire brigades, civil protection, and disaster control". So the missions will be designed having typical scenarios of those fields of applications in mind. And again the trials will be designed to present scenarios as close to real world applications as possible.

REFERENCES

[1] Angel P. Del Pobil, "Why do We Need Benchmarks in Robotics Research?",In *Proceedings IROS-2006 Workshop on Benchmarks in Robotics Research*, Beijing, October 2006
[2] S. Behnke, "Robot Competitions - Ideal Benchmarks for Robotics Research", In *Proceedings IROS-2006 Workshop on Benchmarks in Robotics Research*, Beijing, October 2006.
[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The Robot that Won the DARPA Grand Challenge" , *Journal of Field Robotics*, Vol. 23, No. 9, June, 2006, pp. 661-692.
[4] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, Hitoshi Matsubara, "RoboCup: A Challenge Problem for AI", *AI Magazine* 1997
[5] W. Seemann and K.-D. Kuhnert, "Design and realisation of the highly modular and robust autonomous mobile outdoor robot amor", In *The 13th IASTED International Conference on Robotics and Applications*, Wrzburg, Germany, August 29-31, 2007.
[6] M. Proetzsch and K. Berns and T. Schuele and K. Schneider, "Formal Verification of Safety Behaviours of the Outdoor Robot RAVON",*Fourth International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Angers, France , INSTICC Press, 2007 pp. 157-164
[7] Kaustubh Pathak, Andreas Birk, Soeren Schwertfeger, Ivan Delchef, and Stefan Markov "Fully Autonomous Operations of a Jacobs Rugbot in the RoboCup Rescue Robot League 2006", *International Workshop on Safety, Security, and Rescue Robotics (SSRR)*,IEEE Press, 2007.
[8] van der Mark, W., van den Heuvel, J.C.,Groen, "Stereo based Obstacle Detection with Uncertainty in Rough Terrain", *Intelligent Vehicles Symposium, 2007 IEEE*, Istanbul, 2007, pp. 1005-1012.
[9] Research Institute for Applied Sciences *http://www.fgan.de/fgan/fgan_en.html*
[10] Civilian and Military ELROB website *http://www.elrob.org/*

The e-puck, a Robot Designed for Education in Engineering

Francesco Mondada[1], Michael Bonani[1], Xavier Raemy[2], James Pugh[2], Christopher Cianci[2], Adam Klaptocz[3], Stéphane Magnenat[1], Jean-Christophe Zufferey[3], Dario Floreano[3], Alcherio Martinoli[2]

***Abstract*— Mobile robots have the potential to become the ideal tool to teach a broad range of engineering disciplines. Indeed, mobile robots are getting increasingly complex and accessible. They embed elements from diverse fields such as mechanics, digital electronics, automatic control, signal processing, embedded programming, and energy management. Moreover, they are attractive for students which increases their motivation to learn. However, the requirements of an effective education tool bring new constraints to robotics. This article presents the e-puck robot design, which specifically targets engineering education at university level. Thanks to its particular design, the e-puck can be used in a large spectrum of teaching activities, not strictly related to robotics. Through a systematic evaluation by the students, we show that the e-puck fits this purpose and is appreciated by 90 percent of a large sample of students.**

I. Introduction

Mobile robots are both fascinating objects and the result of the fusion of multiple competences. This fascination leads to the organization of plenty of robotics contests worldwide annually [16]. From an engineering point of view, the design and control of mobile robots requires skills in many disciplines such as mechanics, electronics, energy management, computer science, signal processing, and automatic control. The combination of these two aspects (fascination and inter-disciplinarity) makes mobile robots an excellent educational platform that enables students to address a broad range of engineering fields.

This paper presents, for the first time, the design approach resulting in the *e-puck*[1], an educational desktop mobile robot developed at the École Polytechnique Fédérale de Lausanne (EPFL) for a broad exploitation in teaching activities. The main objectives of this development were:

- The use of a common platform in all EPFL courses related to mobile robotics, to replace the different robots previously in use.
- The use of a mobile robot in non-robotic courses, for instance signal processing, automatic control, and embedded programming, in order to propose more project-based exercises.
- The introduction of mobile robots earlier in the curriculum, which implies the deployment in larger classes (50 to 100 students).

II. Existing robots for education

A wide range of mobile robots are available on the market. In this section we survey the subset of them that we think to be relevant as educational platforms.

This work was supported by the École Polytechnique Fédérale de Lausanne, Switzerland (EPFL, http://www.epfl.ch) in the framework of a FIFO project (Fond pour l'Innovation dans la Formation). All authors are associated with EPFL, in the following laboratories: 1. Laboratoire de Systèmes Robotiques, 2. Distributed Intelligent Systems and Algorithms Laboratory, 3. Laboratory of Intelligent Systems.

[1]e-puck: http://www.e-puck.org

Fig. 1. The e-puck robot.

The Khepera II from K-Team is a redesign of the original Khepera robot [15]. With the same size of the original Khepera, it is compatible with its extensions, software, and scientific tools. The Khepera II is interesting because its size allows to use it on a desktop. It is expensive (around 1500 € for a basic configuration) but is known for being reliable and well supported.

The Hemisson from K-Team is a cheap platform (225 €) with a diameter of 120 mm. It only provides a limited computational power and few sensors in its basic configuration, but is extensible. It is a robust platform well suited for beginners.

The IdMind's circular GT kit is a similar platform slightly cheaper (210 €) with a diameter of 150 mm. It has less standard extensions than Hemisson, but has more I/O available ports to connect self-made extensions. It is more suited for experimentation using custom self-made extensions.

Even cheaper (175 €), the platform Bot'n Roll is representative of a set of simple robots with few sensors that are excellent starting kits for beginners. This and the previous kit improve their accessibility by providing graphic programming environments.

The Lego Mindstorms RCX was the first robotic platform from the Lego company. The RCX is built around a small 8 bit processor and can manage only 3 inputs and 3 outputs (typically DC motors without encoder); but the combination with the Lego bricks makes it a fantastic tool to discover new robots shapes. The RCX has been replaced in 2006 by the Lego Mindstorms NXT. This newer version is equipped with more advanced sensors, including color and sound, and can drive motors equipped with encoders. It is a clear reference in the field because of its good computational power, its flexibility and interesting price (260 €).

The Palm Pilot Robot Kit (PPRK) is a commercially available platform[2] from Carnegie Mellon University combining a mobile base and a personnal digital assistant (PDA), originally a Palm Pilot. The PDA provides the computational power and the user interface and controls the sensors and the actuators through a serial connection with a PIC processor. The result is a compact omnidirectional platform with three distance sensors (in its basic configuration for 250 €). Furthermore, the availability and the maturity of Palm development tools makes this platform an interesting starting kit.

The Cye platform[3] is a medium-size robot (40×28×13 cm) equipped with special wheels that ensure a good odometry. Its price is around 540 €. Designed for indoor domestic environments, Cye can carry extensions such as a vacuum cleaner and can navigate in indoor environments.

The Khepera III from K-Team is a research oriented platform much larger than the Khepera II (120 mm of diameter). It is adaptable to specific research requirements through extensions, for instance the korebot board which provides an XScale processor. Flexible, efficient, and powerful with respect to its size, this robot is also quite expensive as an educational robot (around 2000 € in basic configuration).

The ER1 from Evolution Robotics[4] is a simple aluminum frame kit supporting a laptop (not included) and equipped with wheels. The laptop provides the computational hardware, which improves the performance / cost ratio of the kit (the basic configuration costs around 230 €). The motor wheels controller has some free inputs/outputs but provides limited computational power. This low-cost kit comes with a sophisticated but expensive software environment for navigation and vision.

The KHR-1 from Kondo [5] has been the first humanoid robot with good mobility capabilities (17 DOF) for a price under 1000 €.

The Pioneer 3 (P3) is the latest version of the Pioneer robot by ActivMedia. It is a large (44×38×22 cm) solid platform on which the user can install custom processors, sensors, and actuators. The AmigoBot of the same company is a cheaper version (1550 €) of the same concept.

The Garcia from Acroname[6] is a small robot frame (25×18×10 cm) designed to be controlled by a companion XScale board. The size of Garcia makes it suitable for experiments in compact environments. Its price is around 1360 € in the basic configuration.

Robotino from Festo[7] is a modern mobile robotic platform for education. Robotino runs a real-time Linux kernel and is built around industrial standards. These features make this robot powerful in term of computational power but also expensive (about 4500 €). Robotino is well suited for technical schools that want to approach technical problems using robotics.

Roomba Create from iRobot is an educational/research version of the roomba vacuum cleaner. Devoided of the cleaning module, this platform provides low cost mobility (100 €). Its sensors, designed for vacuum cleaning tasks, offer a good support for reactive navigation. The limited internal processor, dedicated to the low-level robot control, is programmable by simple scripts. Any advanced programming or supplementary I/O requires an additional main processor.

[2]PPRK: http://www.cs.cmu.edu/~pprk/
[3]Cye: http://www.personalrobots.com
[4]Evolution Robotics: http://www.evolution.com
[5]Kondo: http://www.kondo-robot.com
[6]Acroname: http://www.acroname.com
[7]Festo: www.festo-didactic.com

III. Robot design for engineering education

Most of the aforementioned products are exclusively either efficient mobile robots or good educational tools. However, being both implies the following criteria:

- *Desktop size*. A robot that can evolve on the desk near the computer improves drastically the student efficiency during experimentation. We consider that for a good mobility, the experimentation space should be 10 times the diameter of the robot. On a table, this implies a robot diameter smaller than 80 mm.
- *Wide range of possibilities from an engineering and educational point of view*. To exploit this tool in various fields of education such as signal processing, automatic control, embedded programming, or distributed intelligent systems design, the robot should provide a wide set of functionalities in its basic version.
- *User friendly.* The user interface has to be simple, efficient, and intuitive. This is an important point for the acceptance of the system by the students.
- *Low cost.* The broad introduction in engineering classes requires a large number of robots. Knowing that the budget of many schools is constant or decreasing, this is only feasible by reducing the cost of an individual robot.
- *Open information.* This robot has to be shared among professors, laboratories, schools and universities. An open source hardware/software development model is an effective way to achieve this goal.

None of the platforms available on the market is respecting these criteria. Most robots are large and thus need to operate on the floor. The smallest robots are either expensive or have limited functionalities. Very few are open source.

This motivated us to create the e-puck robot in summer 2004. We tested a first set of prototypes with students during the 2004–2005 academic year. Based on this experience, we redesigned the robot and produced the final version in summer 2005 (Fig. 1). The following sections present the robot design and a student evaluation based on its use during four semesters from 2005 to 2007.

IV. The e-puck desktop mobile robot

We based the design of the robot on the first two aforementioned criteria: desktop size and flexibility. Combined, these two constraints imply the miniaturization of a complex system. To achieve a low price, we opted for the use of cheap components and mass production manufacturing techniques. We took special care to make the robot as user-friendly and interactive as possible, in order to ensure that it would be well received by students.

In this design process, a central aspect is the choice of the robot features. This particular choice is one of the

innovations of the e-puck design. The sensors, actuators, and interfaces of the e-puck are representatives of a wide range of devices one can find in several engineering sub-domains:

- sensors in different modalities: audio, visual, distances to objects, gravity,
- input devices with different bandwidths from 10 Hz to 10 MHz (Figure 2, left),
- actuators with different actions on the environment (Figure 2, right),
- wired and wireless communication devices,
- two types of processors: general purpose and DSP.

By exploiting this large set of possibilities, a teacher can present and the student can practice a broad set of engineering fields and skills.

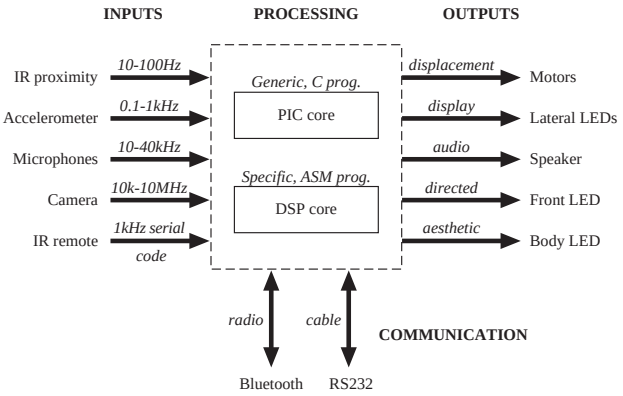


Fig. 2. The e-puck allows exploration of different engineering topics, control options, and signal bandwidths.

For applications where the basic features are not sufficient, the e-puck can be extended with specific hardware. We provide embedded software consisting of a library and several demo applications. We also provide an open source simulator and a monitoring tool to run on a desktop computer. In this section we present the detailed hardware and software design choices.

A. e-puck hardware (basic configuration)

1) Microcontroller: The electronic structure of the e-puck (Fig. 3) is built around a Microchip dsPIC microcontroller. This microcontroller complies with the educational criteria of flexibility because it embeds both a 16 bit processor with a 16 entry register file and a digital signal processor (DSP) unit. This CPU runs at 64 MHz and provides 16 MIPS of peak processing power. The instruction set is mostly orthogonal[8] and rich; in particular, it contains multiply-acccumulate and hardware-repeat instructions suitable to drive the DSP unit, for instance to efficiently compute scalar products and fast fourier transforms. Finally, this processor is supported by a custom tailored version of the GCC C compiler. For the e-puck, we chose a microcontroller version with 8 kB of RAM and 144 kB of flash memory.

[8]An instruction set is *orthogonal* if any instruction can use data of any type via any addressing mode (http://en.wikipedia.org/wiki/Orthogonal_instruction_set)

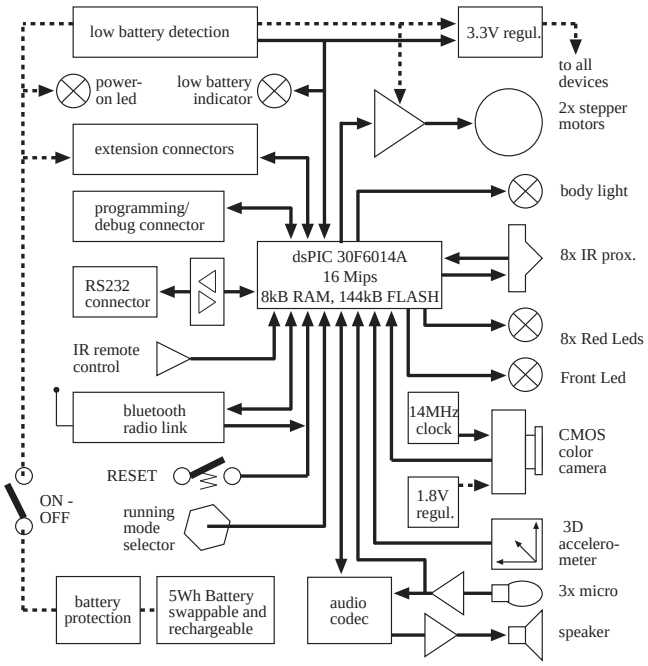


Fig. 3. The outline of the electronic of the e-puck.

2) Sensors and actuators: To ensure a broad range of experimentation possibilities, the e-puck contains various sensors covering different modalities:

- Eight infrared (IR) proximity sensors placed around the body measure the closeness of obstacles or the intensity of the ambient infrared light. These are typical sensors for simple navigation in cluttered environments.
- A 3D accelerometer provides the acceleration vector of the e-puck. This vector can be used to measure the inclination of the e-puck and the acceleration produced by its own movement. It can also detect collisions and if the robot falls. This sensor is rarely included in miniature low-cost mobile robots. We decided to include it because it allows a rich set of experiments.
- Three microphones capture sound. Multiple microphones allow the e-puck to localize the source of the sound by triangulation. The bandwidth of this signal is much larger than the one of the accelerometer or of the infrared sensors, making the microphones, because of their larger computational demands, the ideal tools to learn how to use the DSP unit.
- A color CMOS camera with a resolution of 640×480 pixels in front of the e-puck enables experimentation in vision. Only a sub-part of the image can be grabbed: the size of acquisition is limited by the memory size of the dsPIC and the rate is limited by its processing power. Any format of sub-image is acceptable, providing these two constraints are fulfilled. For instance, the e-puck can grab a color image of 40×40 pixels at 4 frames per second; the frame rate is doubled in gray-scale. This limitation shows to the students the impact of high bandwidth sensors such as cameras.

The e-puck provides the following actuators:

- Two stepper motors. They control the movement of

the wheels with a resolution of 1000 steps per wheel revolution.
- A speaker, connected to an audio codec. Combined with the microphones, the speaker can create a communication network with the ability to detect the direction of peers. It is also an excellent output device for human interaction.
- Eight red light emitting diodes (LED) placed all around the e-puck. These LEDs are covered by a translucent plastic and the e-puck can modulate their intensities. They provide a visual interface with the user; furthermore, another e-puck can observe them with its camera which allows mutual visual interactions.
- A set of green LEDs placed in the transparent body. By lighting the body, they improve the interactions with the user.
- A red front LED placed beside the camera. This LED generates a focused beam that projects a red spot on objects in front of the e-puck. Combined with the camera, this spot allows distant measurements at longer range than the infrared proximity sensors.

3) User interface: The e-puck also contains several devices to interact with the user and to communicate with other equipments:

- Two LEDs show the status of the battery: One indicates whether the robot is powered on, while the other indicates a low battery condition.
- A connector to interface to an in-circuit debugger, to program the flash memory and to debug code.
- An infrared remote control receiver, to control the e-puck with standard television remote controls.
- A classic RS232 serial interface to communicate with a desktop computer.
- A Bluetooth radio link to connect to a desktop computer or to communicate with up to 7 other e-pucks.
- A reset button.
- A 16 positions rotary switch to specify a 4 bit number, which can be used, for instance, to select among pre-programmed behaviors or parameters.

4) Mechanics: The robot has a diameter of 75 mm and a height which depends on the connected extensions. The mechanical structure of the e-puck consists of injected plastic parts. We have chosen this manufacturing technique because it reduces the unit price of the robot for sufficient quantities. The robot structure is simple, being made of only four injected plastic parts: the main body, the light ring, and the two wheels (Fig. 4). The main body is the core of the mechanical structure and encloses the battery. The user can extract the battery from the bottom of the e-puck. The two motors are simply laterally screwed onto the main body, with the wheels directly attached to the motor axis. The main printed circuit board (PCB), containing most of the electronics, is screwed on top of the main body. A light diffusion ring and a default extension board are mounted over this main PCB; the user can replace the default extension board with application specific boards, as illustrated by some examples in the next section. All mechanical parts are transparent and allow to observe all components.

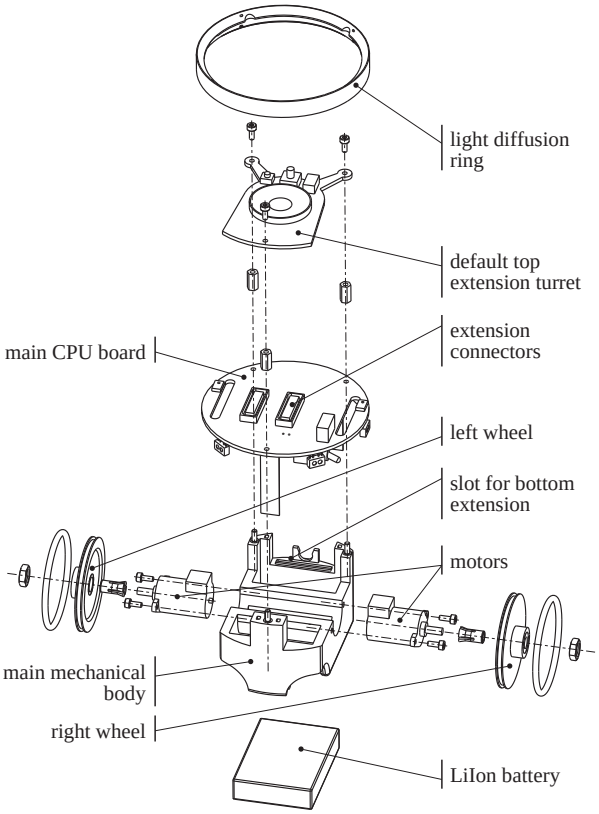


Fig. 4. The mechanical structure of the e-puck in an exploded view.

B. e-puck extensions

To address the needs of specific teaching scenarios that require different mechatronics, the user can connect physical extensions to provide additional sensors, actuators, or computational power. Extensions physically connect through an extension bus which routes a two wires interprocessor communication bus (I^2C) as well as the connections to most sensors.

There are three physically different types of extensions: “top”, “bottom”, and “sandwich”.

Top extensions do not allow other extensions above them. The basic e-puck includes a default extension board of this type which provides the speaker, the 16 positions rotary switch, the infrared remote control receiver, the RS232 connector, and a reset button (Fig. 1, right). There are several other extensions of this type available, for instance:

- A rotating scanner. It is equipped with infrared triangulation distance sensors with a range of 40 cm (Fig. 5, left). For exercises involving robot localization, the short range (2–3 cm) of the proximity sensors available on the basic version of the e-puck is not sufficient.
- A turret with three linear cameras. It provides a very large field of view for measuring optical flow [8], [21] (Fig. 5, right) still requiring few memory.

Bottom extensions are internal PCBs, enclosed in a vertical slot in the front part of the main body. These extensions are close to the ground and are connected to the main PCB by an I^2C bus and a power connection. For instance, an extension of this type provides three analog sensors which measure the ground color (Fig. 6, left top).

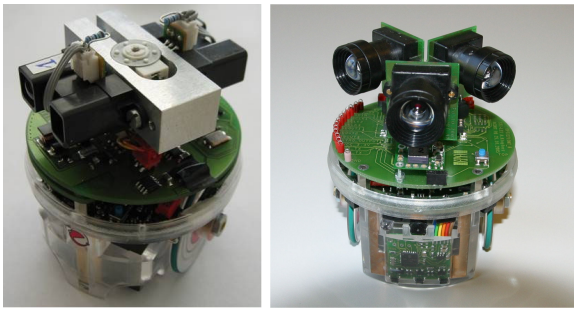


Fig. 5. Two "top" extensions for the e-puck robot: an infrared distance scanner (left) and a large field of view linear camera (right).

Fig. 6. Left top: A bottom extension to measure the ground color. Left bottom: a "sandwich" extension implementing a Zigbee radio link. Right: A complete visual communication system consisting of two "sandwich" extensions.

Extensions can also be "sandwich" boards that replicate the connector on their top, for instance:

- a Zigbee radio link with adjustable radio communication range does not require mechanical access to the top of the e-puck [4] (Fig. 6, left bottom);
- a color RGB LED ring (Fig. 6 right);
- an omnidirectional vision turret with onboard microcontroller dedicated to vision (Fig. 6 right)

Several extensions of this type can be stacked, and a top turret placed above (Fig. 6 right).

C. e-puck embedded software

To develop software for the e-puck, we provide standard components: a bootloader to program the e-puck over Bluetooth, a low-level library to drive the hardware, and a monitor to communicate with a desktop computer. Within the community of e-puck users, standard software modules are also developed. For instance a Player server is developed by VeRLab[9]. These components are all released under the e-puck open source license.

1) Bootloader: To program and debug a microcontroller, one typically uses an in-circuit debugger, which is a specific piece of hardware. To reduce cost and remove this requirement, the e-puck comes with a bootloader that allows the user to re-program the flash of the microcontroller through the Bluetooth or the serial port. At boot, the bootloader listens a small amount of time for activity on these ports, and if none is detected, launches the user application. During this time, the user has the opportunity to send a special command in order to flash a new program.

2) Low-level library: To facilitate the use of the e-puck hardware, which requires specific code with precise timings, we provide a collection of functions called the "Low-Level Library". This library contains functions such as "move the right motor at that speed", "read 40×40 pixel image", or "send this message through Bluetooth". It is statically linked with the user application at compile time.

3) BTcom protocol: When developing for a robot, it is often useful to be able to control it from a desktop computer. To that end, the e-puck comes with a monitor implementing a remote control protocol through the Bluetooth or the serial port, called the "BTcom protocol". This protocol provides full remote control of the e-puck, allowing the desktop computer to set the speed of the motors, read the image of the camera, specify the state of the LEDs, read the accelerometer, and so on. This allows the user to develop applications on a desktop computer, in a comfortable environment with a rich set of development tools. Moreover, with this approach, applications can exceed the computational capabilities of the dsPIC. The same strategy can be used to control the e-puck from a simulator, such as Enki or Webots.

D. e-puck simulation

Several simulators support the e-puck. Among them, we use Webots and Enki. Webots [13] is commercial and supports three-dimensional physics through the ODE[10] library. Enki[11] is open source and provides fast 2D physics, which, for instance, makes it suitable for evolutionary robotics experiments [7].

E. Availability

All our software, design, and production documents are available under an open source hardware/software license, meaning that anyone can use and modify them as long as they comply with the license.

Currently, two companies (GCtronic and AAI Japan) produce the e-puck and about ten companies distribute it under the term of the open hardware license. A growing community is using the e-puck both for research and education (see for instance [1], [20], [18], [12], [11], [5], [2], [10], [14], [9], [17]).

The production price of the e-puck basic version is around 250 €. The selling price by most companies is around 550 €.

[9]Player driver for the e-puck robot: http://code.google.com/p/epuck-player-driver/

[10]Open Dynamics Engine: http://www.ode.org

[11]Enki: http://home.gna.org/enki/

V. Engineering education using the e-puck robot at EPFL

At EPFL, several courses exploit the e-puck robot as experimentation platform. In particular we organize practical exercices in the following teaching areas:

- *Signal processing*. We use the e-puck to explore signal processing on sound, which is an optimal context for education because of the signal low frequency, low propagation speed, ease of generation, acquisition, and direct perception by humans. In our embedded programming course we explore signal processing applications during two sessions of five hours each. We analyze the sound using a fast fourier transform (FFT), with a close attention to its efficient implementation on the DSP unit. In each session, we first present the theory; then the students verify it on MATLAB with data acquired from the robot, and finally they implement it on the dsPIC of the e-puck.
- *Automatic control*. In our embedded programming course, we apply automatic control concepts on the e-puck. The goal of the five hours session is to build a regulator that controls the e-puck and maintains a constant distance with respect to a visual target (Fig. 7, left). The input of a proportional-integral regulator is the estimated distance to the target, extracted from the camera looking at a vertical black stripe on a white background (Fig. 7, right). The output of the regulator is assigned to the speed of both wheels, reducing the problem to one degree of freedom.

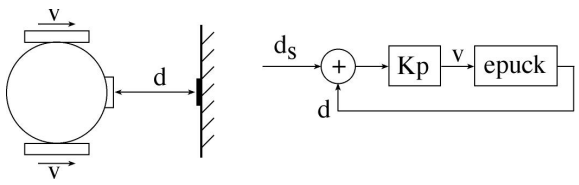


Fig. 7. Model of the problem (left) and scheme of the regulator (right) for the e-puck automatic control practical.

- *Behaviour-based robotics*. In a robotics course, we use the e-puck to provide the students with a better understanding of the subsumption architecture [3], its advantages and its shortcomings. The goal of the exercise is to implement a behavior-based controller for a simplified industrial application consisting of the transport of goods throughout a warehouse. The e-puck (the carrier) has to follow a line painted on the ground (the track), while avoiding obstacles and going back to the line after disruption. To that end, the e-puck is equipped with the ground sensor extension (see Section IV-B and Fig. 6, left top).
- *Distributed intelligent systems*. Our course on distributed intelligent systems (formerly focusing on swarm intelligence) includes weekly laboratory exercises in which the students use a combination of real e-puck robots and realistic simulations using Webots [13] to test and verify the topics and theories presented in lectures. This help the students to assimilate theoretical concepts around multi-robot coordination and networking, and to understand the difficulties of implementing them. This also enhances their awareness of the differences between various types of implementation levels; for example realistic simulations and real experiments.
- *Position estimation and path finding of a mobile robot*. In a 10 hours practical session we use the e-puck to explore robot localization and path finding. These techniques are active fields of research and begin to see deployment in industry [6]. They are usually implemented using expensive sensors such as laser range finder [19], which, because of limited education budgets, often restricts their teaching to simulation. Yet simulation is not always sufficient to fully understand what is critical for correct functionality on physical robots. The e-puck, with the distance scanner extension (see Section IV-B and Fig. 5, left), allows students to explore these techniques in a physical setup.

VI. Evaluation of e-puck by the students

Our embedded programming course using the e-puck has been evaluated on a regular basis since the introduction of the e-puck in 2005. The goal of this course is the understanding and practice of embedded programming. Most of the course focuses on C programming, with links to assembler and some concepts of C++. Practicals explore signal processing and control problems. The students who attend this course are not specialized in robotics. All the programming exercises of this course are performed using the e-puck as a motivation tool.

Over the first three years of e-puck use, students were asked to give feedback about the use of the e-puck to illustrate the concepts of the course. Fig. 8 summarizes the results of this analysis and shows that more than 90 percent of the students agree that the e-puck is a good tool to illustrate the concepts of the course. The score has slightly improved over the years, with the fine-tuning of the course.

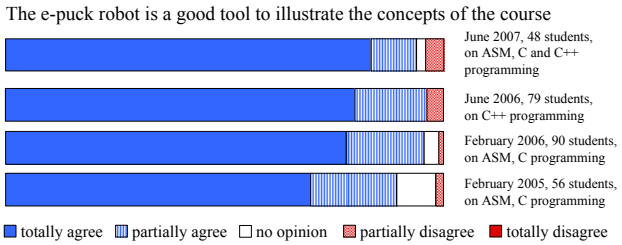


Fig. 8. Comparison between the evaluations of the e-puck robot as teaching tool.

When asked about the quality of the e-puck, students gave the feedback shown in Fig. 9. These results show that the latest version of e-puck is considered as performing well by more than 90 percent of the students. We observe a clear improvement since the first version evaluated in February 2005.

VII. Conclusion

The e-puck robot is an innovative education tool for engineering in a broad sense. For its size and price, it is a complex systems that we exploit to teach a wide range of topics. By integrating the latest technologies in a compact

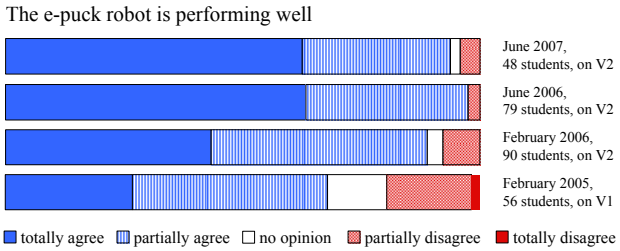


Fig. 9. Comparison between the evaluation of the quality of the e-puck robot and its on-board software, following its evolution over the years and versions.

design, it repositions robotics in the engineering education landscape. More than 200 students over 3 years have validated this concept and shown their satisfaction using the e-puck. The open-source nature of this robot improves the quality of the support to the students by providing full access to knowledge at every level. For teachers this simplifies maintenance and opens new experimentation possibilities. Finally, the rapid diffusion of the e-puck in the research community shows its versatility as a scientific experimentation tool.

VIII. Acknowledgment

We thank EPFL for funding the development of the e-puck robot through its Funding Program for Teaching and Learning. We also thank the Pedagogical Research and Support team of EPFL for the detailed evaluation of the e-puck educational performances. We finally thank our students for their constructive feedback.

References

[1] Alberto Acerbi, Davide Marocco, and Stefano Nolfi. Social facilitation on the development of foraging behaviors in a population of autonomous robots. *Advances in Artificial Life*, 4648:625–634, 2007.

[2] Saeed Amizadeh, Majid Nili Ahmadabadi, Babak N. Araabi, and Roland Siegwart. A bayesian approach to conceptualization using reinforcement learning. *Advanced intelligent mechatronics, 2007 ieee/asme international conference on*, pages 1–7, 2007.

[3] Rodney A. Brooks. *Cambrian Intelligence*. MIT Press, 1999.

[4] Christopher M. Cianci, Xavier Raemy, Jim Pugh, and Alcherio Martinoli. Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics. In *Simulation of Adaptive Behavior (SAB-2006), Swarm Robotics Workshop*, pages 103–115, 2006, Lecture Notes in Computer Science (2007), vol. 4433.

[5] Roozbeh Daneshvar, Abdolhossein Sadeghi Marascht, Hossein Aminaiee, and Caro Lucas. *A Quantitative Investigation into Distribution of Memory and Learning in Multi Agent Systems with Implicit Communications*, volume 4850, pages 124–133. Springer, 2007.

[6] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics and Automation Magazine, IEEE*, 13(2):99–110, June 2006.

[7] Dario Floreano, Sara Mitri, Stéphane Magnenat, and Laurent Keller. Evolutionary Conditions for the Emergence of Communication in Robots. *Current Biology*, 17:514–519, 2007.

[8] N. Franceschini, J. M. Pichon, C. Blanes, and J. M. Brady. From insect vision to robot vision [and discussion]. *Philosophical Transactions: Biological Sciences*, 337(1281):283–294, 1992.

[9] O. Gigliotta and S. Nolfi. Formation of spatial representations in evolving autonomous robots. *Artificial Life, 2007. ALIFE '07. IEEE Symposium on*, pages 171–178, 2007.

[10] Babak N. Araabi Hadi Firouzi, Majid Nili Ahmadabadi. A probabilistic reinforcement-based approach to conceptualization. *International Journal of Intelligent Technology*, 3:48–55, 2008.

[11] Daisuke Kurabayashi Herianto, Toshiki Sakakibara. Artificial pheromone system using rfid for navigation of autonomous robots. *Journal of Bionic Engineering*, 4:245–253, 2007.

[12] Mattias Jacobsson, Sara Ljungblad, Johan Bodin, Jeffrey Knurek, and Lars Erik Holmquist. Glowbots: robots that evolve relationships. In *SIGGRAPH '07: ACM SIGGRAPH 2007 emerging technologies*, page 7, New York, NY, USA, 2007. ACM.

[13] Olivier Michel. Webots: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42, 2004.

[14] Yasser F. O. Mohammad and Toyoaki Nishida. *TalkBack: Feedback from a Miniature Robot*, pages 357–366. Springer, 2007.

[15] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. In T. Yoshikawa and F. Miyazaki, editors, *Proceedings of the Third International Symposium on Simulation on Experimental Robotics (ISER-93)*, volume 200 of *Lecture Notes in Control and Information Sciences*, pages 501–513. Springer, 1993.

[16] R.R. Murphy. "competing" for a robotics education. *Robotics and Automation Magazine, IEEE*, 8(2):44–55, Jun 2001.

[17] F. Rastegar and M.N. Ahmadabadi. Grounding abstraction in sensory experience. *Advanced intelligent mechatronics, 2007 ieee/asme international conference on*, pages 1–8, 2007.

[18] P. Roduit, A. Martinoli, and J. Jacot. A quantitative method for comparing trajectories of mobile robots using point distribution models. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2441–2448, 2007.

[19] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. MIT Press, 2004.

[20] Vlad M. Trifa, Christopher M. Cianci, and Dominique Guinard. Dynamic Control of a Robotic Swarm using a Service-Oriented Architecture. In *13th International Symposium on Artificial Life and Robotics (AROB 2008)*, 2008.

[21] Jean-Christophe Zufferey and Dario Floreano. Toward 30-gram Autonomous Indoor Aircraft: Vision-based Obstacle Avoidance and Altitude Control. In *IEEE International Conference on Robotics and Automation (ICRA'2005)*, pages 2594– 2599, 2005.

This page is left blank intentionally

robotica 2009
festival nacional de robótica

Hexapod Walking as Emergent Reaction to Externally Acting Forces

Adam El Sayed Auf, Nico Dudek and Erik Maehle

Abstract— **Insect-like walking with a six-legged robot involving 18 degrees of freedom is a fascinating as well as challenging task in terms of controlling and coordinating the 18 joints. Especially the control of those legs touching the ground, thus being mechanically coupled, is complex. External forces acting to the walker's joints contain the information for each joint where and how far to move. This paper presents a decentralized controller approach measuring externally acting forces in each joint and combining active compliance with a step performing reflex. Local communication between the six legs as well as individual complying of the selected joints is used to achieve walking as an emergent reaction to the externally acting forces. This approach is based on an organic computing architecture and is implemented and tested on a six-legged walking machine.**

I. INTRODUCTION

Observing insects moving with a surprising facility through extremely unstructured environments like branchwood, leaves, and under branches can make a robotic engineer still jealous. Although great six-legged walking machines were built [1][2], insects still outclass their robotic half siblings with their locomotion abilities. Insects are able to handle obstacles like large gaps or disturbances like losing foothold contact on slippery ground. They even compensate strong changes in their own body geometry like the loss of a leg. In robotic science these capabilities in unstructured environments are still a challenge. This fact leads to an increasing curiosity for biological principles. For solving these challenges an internal accurate calculation by the insect's neurons seems hardly probable. Biological studies expose a decentralized reflex-based control system in strong combination with the organism's anatomical characteristics such as elasticity and stiffness of muscles and joints. The later component is at least as important as the controller and has been disregarded for a long time. The deeper scientists probe into the biological coherences of animal locomotion the more important the anatomical features and qualities become. The muscles' elasticity seems to play an important part in terms of walking stability as well as compensating external forces acting on the walker's body. In particular the walking machine's legs touching the ground at the same time produce forces acting on the robot's body, which seem to be compensated in a biological organism. This compensation corresponds to the accurate calculation of a closed kinematic loop built up from the ground touching legs. Introducing a positive feedback concept [3][4] in the walker's joints as successfully presented in two versions of the Walknet simulation [5][6] may contribute to the improvement of walking robots. Positive feedback or active compliance has been implemented to improve walking in the DLR-Crawler [7].

This work contributes to the topic of legged locomotion by combining active compliance with a reflex based system to produce the complex behaviour of six-legged walking. Here, six-legged walking is a reaction to the trigger of active forces. The global reaction arises from local joint reflexes. While a positive feedback concept was successfully introduced in a hexapod simulation in [5][6], in this work a similar concept using a reduced set of parameters is implemented and tested on a real robot. In the following section the robotic platform OSCAR (Organic Self Configuring and Adapting Robot) is introduced. Subsequent, the decentralized Organic Robot Control Architecture (ORCA) [8][9] particularly its walk controller modules are described. In section IV the active compliance approach embedded in the ORCA concept is elucidated in detail. In section V the reflex based walking reaction to external forces is explained. Finally, section VI shows the experimental results of the introduced active compliance approach.

Institute of Computer Engineering, University of Lübeck, Ratzeburger Allee 160, D-23538 Lübeck, Germany. Corresponding author: Adam El Sayed Auf, email: elsayedauf@iti.uni-luebeck.de

II. ROBOTIC PLATFORM OSCAR

The robotic platform OSCAR (Organic Self Configuring and Adapting Robot) is a hexapod walking machine (Fig. 1) with 18 degrees of freedom. Each leg consists of three joints and their linking segments. The joints are denoted with alpha, beta and gamma, where the leg's protraction and retraction is performed by the alpha joint, the elevation and depression by the beta joint and the extension and the flexion by the gamma joint. Table 1 gives more detailed information about the robot's technical data. The legs are attached at the walker's round symmetric body in an angle of 60°. Each joint represented by a servo motor is equipped with an Open Servo Module offering the parameters goal position, real position, and current power consumption. In addition each leg has a binary ground contact sensor. The servo motors are connected via I^2C bus. The control software runs either on a personal computer or on an ASUS eeePC 4G 701 that can be carried

Fig. 1. The six-legged robotic platform OSCAR

TABLE I
ROBOTIC TECHNICAL GEOMETRIC DATES

Segment Length	
proximal link	$30\,mm$
medial link	$60\,mm$
distal link	$120\,mm$
Joint Motion Range	
protraction/retraction (alfa joint)	$\pm 90^{\circ}$
elevation/depression (beta joint)	$-60/+90^{\circ}$
extension/flexion (gamma joint)	$-60/+120^{\circ}$
Type of Servo Motor	**Servo motor controllers**
HightTech HS-985MG	OpenServo Module

by the robot. A DIOLAN USB to I^2C adapter connects the PC with the servo motor's I^2C bus. Two batteries ensure the robot's internal power supply. In this work a stationary PC instead of the mentioned eeePC is used.

III. ORGANIC ROBOT CONTROL ARCHITECTURE

The ORCA (Organic Robot Control Architecture) concept describes a decentralized modular controller inspired by organic computing principles. It is used in this work for controlling the six-legged walker OSCAR. The architecture is based on two types of units: the Organic Control Units (OCUs) and the Basic Control Units (BCUs). While the OCUs are supervising units observing and monitoring other OCUs or BCUs, detecting anomalies and reacting to detected failures, the BCUs implement tasks from controlling servos through to simple and higher behaviours.

An OCU is a module observing selected other modules like OCUs or BCUs by monitoring their signals. The observed BCU's signals could be e.g. a servo motor's position signal or a servo motor's current consumption signal. In defined states the signal's default values can be concluded to a default "health signal" by the OCU. The OCU health signal generation as well as the health signal itself can also be monitored by another OCU. Hence, specific health surveillance for single signals as well as an unspecific surveillance for health signals generation and module functionality arise from the interacting OCUs.

The BCUs as characterised above, are hardware control performing modular units for any kind of task. In this work each servo motor of the robotic platform OSCAR is controlled by a BCU sending goal positions and moving speeds to the servo motor hardware and receiving real position and current power consumption. Other BCUs calculate the appropriate signals or implement coordination rules as well as trajectories or simple reflexes.

OCUs as well as BCUs have the capability of communicating locally or to selected Units. In contrast to the OCUs, BCUs communicate only to other BCUs. OCUs are allowed to communicate with both other OCUs and BCUs. This modular and decentralized concert of simple BCUs and observing OCUs is put together to a total system controlling the robots locomotion.

In this work active compliance in the alpha joints, step reflexes and also trajectory calculation is implemented as BCU for each joint communicating with the appropriate joints of the neighbouring legs. OCU implementations are neglected in this work.

IV. ACTIVE COMPLIANT UNIT

The active compliant units are implemented as BCUs for each of the 18 joints. They are all based on the same structure. Each BCU has to measure the force acting on its associated joint and to decide whether its joint needs to comply or not. The complying decision depends on the according joint's predefined working area. Inside the predefined area the joint complies where after leaving the area a reflex is triggered correcting the joint's angle back into its working area.

A. FORCE MEASURING

The OpenServo module provides the current power consumption as averaged power usage at the time of reading in a servo module's own entity. It is taken advantage of the correlation between current power consumption in the OpenServo module and the force acting on the appropriate servo motor. An empirical experiment showed a linear correlation between torque and the measured consumption of power. Therefore, fifteen torque values from $0.1\,Nm$ to $0.7\,Nm$ have been used to define a servo motor's torque power correlation. This correlation is used to define thresholds for the complying decision.

B. ACTIVE COMPLYING

Active compliance is implemented in this work as simple control circuit updating the current servo motor position depending on the difference between the alpha goal position α_g at time step $\alpha_{g,t}$ and current position. A detected force exceeding the predefined threshold acting onto the servo motor leads to a significant difference between goal position and real position, here donated as $\triangle\alpha_{g,t}$. The servo motor is pressing against the acting force. With an appropriate chosen threshold the acting force is preventing the servomotor from reaching its goal position. The emerged difference in position is corrected actively by the active compliance control loop. The control loop initiates a positive feedback to the acting force by moving the current position actively towards the real position in the next time step:

$$\alpha_{g,t+1} = \alpha_{g,t} + \triangle\alpha_{g,t} \tag{1}$$

The servo motor's active compliance reduces the servo motor's current power consumption. This active compliance is biologically inspired by a reflex found in stick insect's gamma joint assisting external forces acting on the joint [10].

C. COMPLYING DECISION

The decision of compliance is based on a simple power consumption threshold for each of the appropriate joints. An acting force on the joint exceeding the predefined threshold introduces the joint compliance after saving the position the force was detected in. The saved position is donated as initial goal position $\alpha_{g,t}^{init}$. If an acting force is detected further on after complying and the threshold is still exceeded the next compliance step will be decided. Here, the decision function $D(I_{\alpha,t})$ is dependent on the normalised measure $I_{\alpha,t}$ measuring the current power consumption of the alpha joint at time step t. The threshold is defined as 1. A result of $I_{\alpha,t} > 1$ leads to the active compliance activation.

$$D(I_{\alpha,t}) = \begin{cases} \alpha_{g,t+1} = \alpha_{g,t} + \triangle\alpha_{g,t} & : \quad I_{\alpha,t} > 1 \\ \alpha_{g,t+1} = \alpha_{g,t}^{init} & : \quad I_{\alpha,t} \leq 1 \end{cases} \quad (2)$$

The decision function D is a mapping from a normalised measure onto an angle. The decision process is run cyclically. In case of no force detecting or threshold overstepping, the decision process moves the servo motor back into the position saved at the first force detection $\alpha_{g,t}^{init}$ as mentioned above.

D. STEP REFLEX

Reflexes are fast uniform reactions triggered by defined stimuli. During locomotion reflexes can compensate occurring disturbances like the human patella reflex triggered by a stretch sensory receptor helping humans to walk without consciously thinking about each single step. The biological concept supporting animals in their locomotion system can be transferred to mobile legged robots to improve their behaviours and enhancing their ability of adaptation.

The above explained active compliance can be regarded as a reflex reacting to an acting force stimulus. It leads to a joint's steady movement into the same direction dependent to an acting external force. To ensure keeping a joint in its optimal scope a work area is predefined by a maximum and a minimum angle. A joint exceeding the maximum angle or deceeding the minimum angle has to be moved back into its optimal scope. This movement is implemented for the alpha joint as a reflex performing a step back into the predefined alpha joint's work area. Here $J(A_{\alpha,t})$ is the judgement function where $A_{\alpha,t}$ is a normalised measure for the joint's working area. The verification whether the alpha joint is still in its working area ($A_{\alpha,t} > 1$) or not ($A_{\alpha,t} \leq 1$) decides between the two reactions: performing step reflex denoted as $R(\alpha, \beta, \gamma)$ and performing the compliance decision $D(I_{\alpha,t}), \beta, \gamma$:

$$J(A_{\alpha,t}) = \begin{cases} R(\alpha, \beta, \gamma) & : \quad A_{\alpha,t} > 1 \\ D(I_{\alpha,t}), \beta, \gamma & : \quad A_{\alpha,t} \leq 1 \end{cases} \quad (3)$$

The function J represents a mapping from a single angle onto an agle triple. The angle triple represents all three angles of one leg. Leaving the predefined working area represents a stimulus like a stretch sensory receptor triggering the step reflex performing a swing phase of the leg, lifting up the according leg with the beta and gamma joint, moving it back into the work area with the alpha joint and putting it down again with the beta and gamma joint. This movement is a fast predefined and therefore uniform reaction to a stimulus involving each of the leg's three joints. Thus, a detected stimulus in the alpha joint activates a goal position update in the three appropriate joints of the same leg. Stepping back into a leg's optimal scope supports the stability in standing during complying external acting forces.

V. REFLEX BASED WALKING

Walking corresponds to the coordination of all involved legs as well as controlling the leg's two phases: the swing and the stance phase. While in its swing phase a leg lifts up and moves in respect to the robot's walking direction to the front to touch down on the ground again, its distal end keeps ground contact while the leg pushes the robot's body into its walking direction during stance phase. A decentralized implementation of the coordination for a predefined swing and stance phase based on local communication for the OSCAR platform is explained in detail in [11]. In this work an approach based on the above described active compliance and reflex implementation is presented leading to a walking behaviour exclusively caused by externally acting forces.

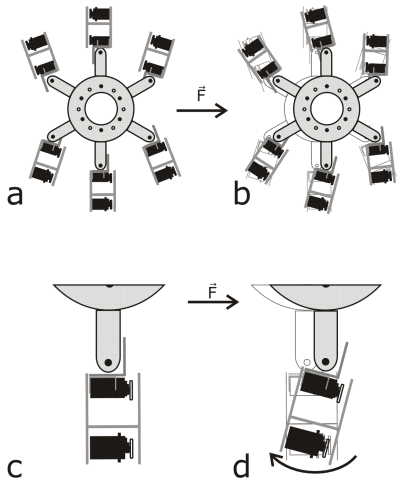


Fig. 2. Schematic top view illustration of OSCAR's reaction to an external acting force $\vec{F}$. a) shows the robot's initial position for walking and an acting force $\vec{F}$. b) result of the acting force pictured over the initial posture. c) and d) are close ups for the middle leg illustrating the effect of active compliance in the alpha joint.

A. PASSIVE STANCE PHASE

A standing leg as well as a leg performing the above described active compliance can be regarded equivalent to a leg performing a stance phase. In contrast to a regular stance phase the reflex based stance phase is not actively pushing the walker's body forward but an acting force pushing the robot's body forward is triggering the active compliance reflex, moving the leg backwards in respect to the walking machine's body. Thus, the active compliance stance phase is more passive in terms of being a result of an active external force than an active movement. Figure 2 a) shows a schematic top view of the six-legged walker illustrating an external force $\vec{F}$, acting on the robot's corpus. In Figure 2 b) the reaction resulting from the active compliance in each of the robot's alpha joints is presented. The illustrations c) and d) of Fig. 2 depict a close up of one alpha joint's reaction to the external force.

B. ACTIVE SWING PHASE

The above described step reflex is similar to a predefined swing phase. One difference between a regular swing phase and an alpha joint step reflex lies in the shorter duration of the step reflex based on the faster reflex reaction. The work area limits represent the corresponding posterior extreme position (PEP) the swing phase in normal walking starts from. This and other coordinating influences have been found in stick insects [12][13]. Another difference is given by the fact that the step reflex can be triggered at any time at one of the two working area limits. Thus, the step reflex reacts in both directions where in normal walking

a forward direction is given and the swing phase is only acting forwards.

The step reflex is a critical reaction concerning the robot's stable standing because the leg lifts up from the ground and ends its support carrying part of the robot's body weight. For this reason, a stable standing has to be ensured by avoiding neighbouring legs to lift up at the same time. supposing the active compliance reflex represents a stance phase and the step reflex matches a swing phase the leg coordination can be implemented exactly as the coordination rule presented in [11]. When the step reflex trigger is active each leg decides locally based on its neighbouring legs' stance and swing state information to stay in its stance phase or to switch into swing phase. In case, one of its neighbouring legs is performing a step reflex, the leg BCU decides not to perform its own step reflex. The working area can be temporally extended to allow the leg continuing its movement. In the event neither one of its neighbouring legs is performing the step reflex it executes its own step reflex. After finishing a step reflex the leg is monitoring again the acting forces triggering the active compliance reflex. This combination of reflexes and leg coordination based on local rules can lead to an overall behaviour of the robot adapting to an external force.

VI. EXPERIMENTAL RESULTS

The described active compliance approach and the reflex activation have been tested in several experiments. The three reactions: active compliance as reaction to an acting force, moving back to the initial position after a force decrease, and the step reflex activation have been tested separately. Also the overall reaction of the six-legged walking machine combining all single reactions has been tested and explored.

A. ACTIVE COMPLIANCE WITHOUT A STEP REFLEX

The force adapting active compliance in combination with the goal position update after a force drop has been tested in a single alpha joint. The alpha joint is set here in its default position, thus, its maximum working area is $\pm 90°$. The upper angle area limit is defined at about $45°$. A force changing the alpha joint's angle from its goal value towards the upper angle limit increases the joint's power consumption.

The power consumption threshold for the alpha joint is defined at $2.5\,\%$ of the maximum servo motor's power consumption. Figure 3 shows two charts: an alpha joint time course of the current consumption in $\%$ of the servo motor's maximum current consumption on the top and a time course of the alpha joint's angle on the bottom. At time $1600\,ms$ the alpha joint's current consumption rises above the defined threshold of $2.5\,\%$. The alpha joints angle is corrected towards the direction, the force is acting to. From time $2000\,ms$ to $3200\,ms$ the current consumption decreases but stays above the current consumption threshold of $2.5\,\%$. The joint's angle is corrected the whole time and rises briefly below the upper angel limit. At time $3200\,ms$ the current consumption falls below the current consumption threshold. The reaction of moving the joint back into its default position is activated and the alpha joint angle is corrected. For moving the alpha joint more current is needed than holding a position, thus, the current consumption rises again during the movement bringing the joint back to its default position. The current consumption signal is ignored during the time the reflex is performed.

The experiment shows the alpha joint's active compliance reaction to an external force. Thus, the angle's working area limit was not exceeded before the external force has disappeared and the alpha servo moves back to its default position.

B. ACTIVE COMPLIANCE WITH A STEP REFLEX

The force adapting active compliance in combination with a step reflex has also been tested in a single alpha joint. Except of the angle's working area limit the same setup as described above is used. In this experiment the angle's upper area limit is defined at about $+25°$. Figure 4 is structured as Figure 3 and shows the alpha joint's angle time course as well as the alpha joint's current consumption time course. At time $1375\,ms$ the current consumption exceeds the threshold of $2.5\,\%$ in the upper chart. The alpha joint increases its angle value towards the upper area limit in the bottom chart. The current consumption stays above the threshold in the upper chart and the alpha angle is corrected continuously towards its upper area limit. At $5125\,ms$ the joint's angle exceeds its predefined working area limit.

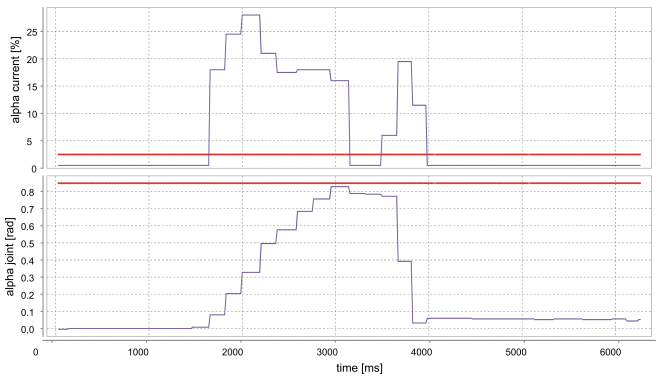


Fig. 3. Alpha joint's current consumption in $\%$ of a servo motor's maximum current consumption over time in milliseconds in the top chart. Current consumption threshold at $2.5\,\%$. Alpha joint's angle in radian measure over time with the upper angle scope limit of about 0.8 radian measure.

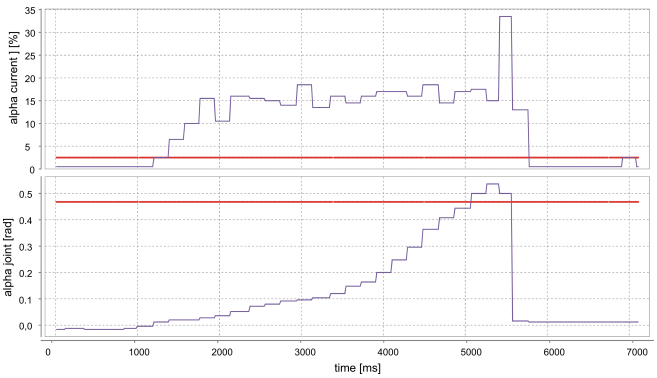


Fig. 4. Alpha joint's current consumption in $\%$ of a servo motor's maximum current consumption over time in milliseconds in the top chart. Current consumption threshold at $2.5\,\%$. Alpha joint's angle in radian measure over time with the upper angle scope limit of about 0.48 radian measure.

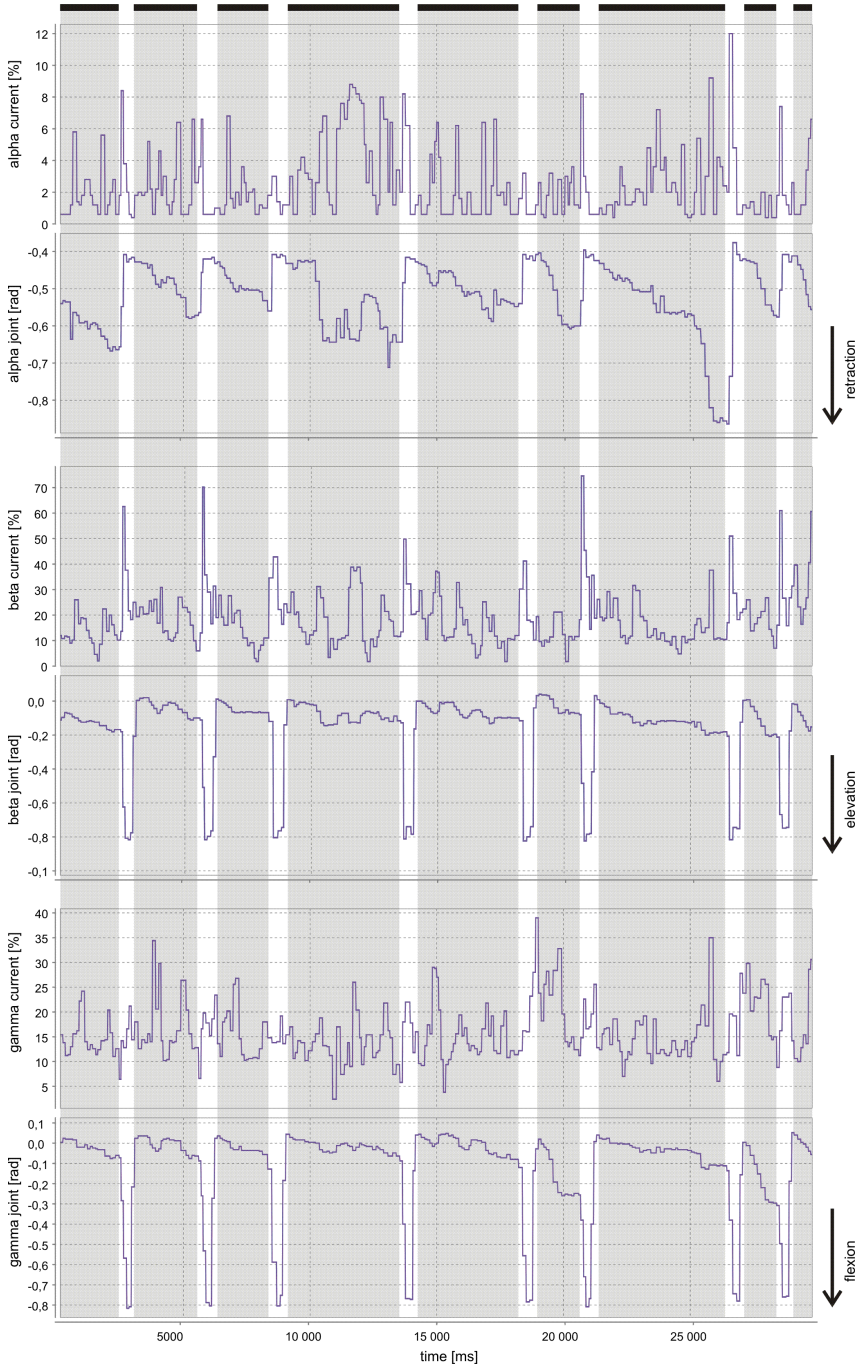


Fig. 5. Current consumption in % of a servo motor's maximum current consumption and joint's angle in radian measure over time in milliseconds for the joints alpha, beta and gamma from top to bottom of the left front leg. Grey background indicates actively complying stance phase and white background step reflex swing phase. At the top of all charts ground contact is pictured as black bars.

At $5500\,ms$ the step reflex is performed and the alpha joint moves back to its default position. During this movement the current consumption rises again and falls after finishing the movement blow the current consumption threshold in the upper chart. The here described result shows the active compliance in the alpha joint until the angle's working area is exceeded and a step reflex is performed. The alpha joint is moved by an external force towards its upper area boarder. Thus, the current consumption rises, exceeds its threshold and the active compliance is activated. The external force stays while the alpha joint is complying and is getting closer to its upper limit. After exceeding the upper angle limit the active compliance is stopped and the step reflex is activated.

C. WALKING AS EMERGENT REACTION

To explore walking as emergent reaction the robot is put on a flat ground, all legs in their default position. For walking depending on the leg its default position is shifted like indicated in the schematic top view in Figure 2. For that reason, an acting external force from the robot's front or from its back is better distributed to all alpha servos. The robot is pulled by a rope to its front until the alpha joints perform their active compliance. After the force pulling the rope with the robot disappears without the alpha joints performing a step reflex, the robot moves back into its initial posture.

A steady force pulling the robot's body by the rope leads to a switch from active compliance to a step reflex in the alpha joints. The in section V described coordination rule prevents an unstable state by avoiding neighbouring legs to perform the step reflex at the same time. In Figure 5 the alpha, beta and gamma joints' time courses for current consumption and angles for the robot's left front leg are plotted over $30\,s$. For each joint its angle in radian measure as well as its current consumption in $\%$ of the maximal servo motor current consumption is plotted. To illustrate the leg's different phases, the active complying stance phase is marked by a grey background and a black bar on top of the charts representing the leg's ground contact while the step reflex as swing phase is denoted with a white background.

Because of being a left front leg, the alpha joint's default position is shifted to $-23°$ $(-0.4\,rad)$. This shift is equivalent to a right hind leg shift. The working area has been reduced to $\pm 10.23°$. Thus, the lower alpha joint's angle border is defined at $-33.23°$ $(-0.58\,rad)$. At time $0\,s$ the alpha joint is already decreasing. After about two seconds the position of the alpha joint's angle is reset back to its default position. Moving back to the leg's default position is also supported by the beta and the gamma joint elevating and flexing the leg and depressing and extending it back on the ground at about time three seconds. During the $30\,s$ extract eight step reflexes are performed where the actively complying stance phases are varying in their endurance. In the stance phase from $21500\,ms$ to $26500\,ms$ the effect of the coordination rule suppressing the step reflex activation shows clearly up. The alpha joint is not allowed to lift up from the ground and the lower angle working area limit is shifted.

Figure 5 shows the experimental results of the left front leg's reaction to the external force acting to the robot's body. Due to loosing grip and slipping away the active complying stance phases in the alpha joints are varying in their angle values. Most of the other legs performed less step reflexes because of slipping towards the pulling force. In Figure 6 a step pattern of all six legs is illustrated.

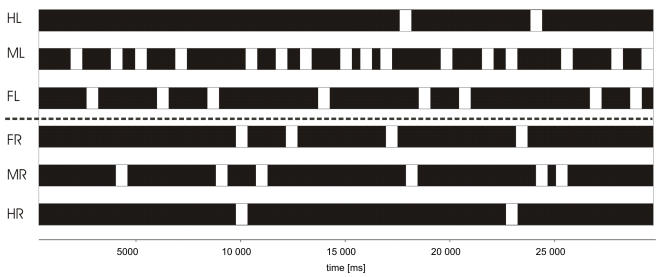


Fig. 6. Step pattern diagram of the left hind (HL), middle (ML) and front leg (FL) from top to bottom and the right front (FR), middle (MR) and hind leg (HR) over time. Black bars represent stance phases white spaces represent step reflexes.

From top to bottom the left hind leg (HL), the left middle leg (ML) and the left front leg (FL) are pictured separated by a grey dashed line from the right front, middle and hind leg (FR MR, HR). The black bars represent ground contact where the actively complying stance phase is performed. During the white spaces a step reflex is executed. While the pulling force seems to act continuously on both front and middle legs, each hind leg performs just two step reflexes. Another noticeable fact is the high frequency of the left middle leg's step reflexes.

VII. DISCUSSION

A decentralized control architecture combining active compliance in the six-legged walker's joints has been presented to achieve a hexapod walking behaviour as an emergent reaction to external forces acting on the walking machine's body. The possibility to replace an active stance phase based on complex calculations to coordinate the mechanically coupled legs by an actively complying reflex in the robot's alpha joints has been tested and explored. Although the basic functionality of the principle of walking as a complying reaction to an external force has been tested successfully the presented results bring up more questions to the topic of active compliance supported walking. Further optimisation and exploration on this approach are still required.

VIII. ACKNOWLEDGMENTS

This work was funded in part by the German Research Foundation (DFG) within priority programme 1183 under grant reference MA 1412/7-1.

REFERENCES

[1] F. Pfeiffer, H.J. Weidemann and J. Eltze, Leg Design Based on Biological Principles, *Proceedings of the 1993 IEEE Conference on Robotics & Automation*, 352-358, Atlanta, USA , 1993.

[2] B. Gassmann, K.U. Scholl and K. Berns, Locomotion of Lauron III in Rough Terrain, *International Conference on Advanced Mechatronics*, Como, Italy, 2001.

[3] M.Schilling, A. Schneider, H. Cruse, and J. Schmitz. Local control mechanisms in six-legged walking. *Proc. of IROS 2008. IEEE/RSJ Int. Conferenc*, 2655-2660, 2008.

[4] A. Schneider, H. Cruse, and J. Schmitz. Decentralized control of elastic limbs in closed kinematic chains. *The International Journal of Robotics Research*, 25(9):913-930, 2006.

[5] H. Cruse, T. Kindermann, M. Schumm, J. Dean, and J. Schmitz. Walknet - a biologically inspired network to control six-legged walking. *Neural Netw.*, 11(7-8):1435-1447, 1998.

[6] V. Dürr, J Schmitz, and H. Cruse. Behaviour-based modelling of hexapod locomotion: Linking biology and technical application. *Arthropod Structure and Development*, 33(3):237-250, 2004.

[7] M. Görner, T. Wimbck, A. Baumann, M. Fuchs, T. Bahls, M. Grebstein, Ch. Borst, J. Butterfass, and G. Hirzinger. The DLR-Crawler: A Testbed for Activley Compliant Hexapod Walking Based on the Fingers of DLR-Hand II. *Proc. of IROS 2008. IEEE/RSJ Int. Conferenc*, 1525-1531, 2008.

[8] A. El Sayed Auf, M. Litza,E. Maehle. Distributed Fault-Tolerant Robot Control Architecture Based on Organic Computing Principles. *Biologically-Inspired Collaborative Computing*, 115-124, Springer, Boston 2008.

[9] F. Mösch, M. Litza, A. El Sayed Auf, B. Jakimovski, E. Maehle, W. Brockmann. Organic Fault-Tolerant Controller for the Walking Robot OSCAR. *Proc. of the Workshop on "Dependability and Fault Tolerance" at ARCS*, VDI, 2007.

[10] C. Bartling and J. Schmitz. Reactions to disturbances of a walking leg during stance. *J. Exp. Biol.*, 203:1211-1223, 2000.

[11] A. El Sayed Auf, F. Mösch, M. Litza. How the Six-Legged Walking Machine OSCAR Handles Leg Amputations. *Proc. of the Workshop From Animals to Animals 9 (Simulation of Adaptive Behaviour - SAB'09)*, Rom, CD, From Animals to Animals 9 (Simulation of Adaptive Behaviour - SAB'09), Rom, Italy 2006.

[12] H. Cruse. What mechanisms coordinate leg movement in walking arthropods? *Trends in Neurosciences*, 13:15-21, 1990.

[13] T. Kindermann. Behavior and adaptability of a six-legged walking system with highly distributed control. *Adaptive Behavior*, 9(1):16-41, January 2002.

A Passive System Approach to Increase the Energy Efficiency in Walk Movements Based in a Realistic Simulation Environment

José L. Lima, José A. Gonçalves, Paulo G. Costa and A. Paulo Moreira

***Abstract*— This paper presents a passive system that increases the walk energy efficiency of a Humanoid robot. A passive system is applied to the simulated robot allowing the energy consumption to be reduced. The optimal parameters for the passive system depend on the joint and gait trajectories. Final results prove the benefits of the presented system apply. It was optimized thanks to a realistic simulator where the humanoid robot was modeled. The model was validated against a real robot.**

I. INTRODUCTION

Newfound research in biped robots has resulted in a variety of prototypes that resemble their biological counterparts. There are several advantages associated with legged robots: they can move in rugged terrains, they have the ability to choose optional landing points, and two legged robots are more suitable to move in a human environment. Consequently, research on biped robots is very active [1]. As humanoid robots are powered by on-board batteries, its autonomy depends on the energy consumption. The trajectory controller can also be optimized having in mind the energy consumption minimization [2] and walking gate optimization [3]. This paper addresses a passive system, that coupled to the humanoid robot joints, allows to save energy. The passive system optimal characteristics depend on each joint desired trajectory. These characteristics can be found by an optimization method. For this purpose, a realistic model for the simulator (SimTwo [4]) was developed. There are several simulators with humanoid simulation capability, like Simspark, Webots, MURoSimF, Microsoft Robotics Studio and YARP: Yet Another Robot Platform [5]. SimTwo, as a generic simulator, allows to simulate different types of robots and allows the access to the low level behaviour, such as dynamical model, friction model and servomotor model in a way that can be mapped to the real robot, with a minimal overhead. This simulator deals with robot dynamics and how it reacts for several controller strategies and styles. It is not an easy task to develop such model for the robot due to the inherent complexity of building realistic models for its physics, its sensors and actuators and their interaction with the world [6]. The paper is organized as follows: Initially, the real robot (which is the system that was modeled in the simulator) and its control architecture are described. Then, section 3 presents the developed simulator. The servo and friction models are presented. Further, section 4 presents the energy efficiency increase based in a passive system approach where optimal parameters are found. Finally, section 5 rounds up with conclusions and future work.

José L. Lima and José A. Gonçalves are with Polytechnic Institute of Bragança, {jllima,goncalves}@ipb.pt

Paulo G. Costa and A. Paulo Moreira are with the Faculty of Engineering of University of Porto, {paco,amoreira}@fe.up.pt

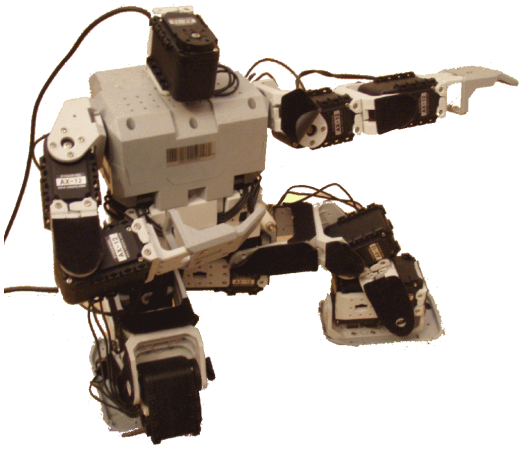


Fig. 1. Real humanoid robot

II. REAL HUMANOID

There are several humanoid robots kits available. The commercially available Bioloid robot kit, from Robotis, served as the basis for the humanoid robot and the proposed biped robot is shown in Fig. 1.

The servo motors are connected to the central processing unit (CM-5), based on the ATMega128 microcontroller, through a serial 1Mbps network. The original firmware presented in the CM-5 can be replaced in order to develop a personalized control application. Its manufacturer provides the source code making it easier to develop a new firmware. Next subsections present the physical robot in which the developed humanoid simulator was based.

A. Main Architecture

The presented humanoid robot is driven by 19 servo motors (AX-12): six per leg, three in each arm and one in the head. Three orthogonal servos set up the 3DOF (degree of freedom) hip joint. Two orthogonal servos form the 2DOF ankle joint. One servo drives the head (a vision camera holder). The shoulder is based on two orthogonal servos allowing a 2DOF joint and elbow has one servo allowing 1DOF. The total weight of the robot (without camera and onboard computer) is about 2 kg and its height is 38 cm.

B. Control Architecture

Multiple layers that run on different time scales contain behaviours of different complexity. The layer map is presented in Fig. 2.

The lowest level of this hierarchy, the control loop within the Dynamixel actuators (AX-12), has been implemented by Robotis. The servomotor is an embedded system, based on a ATMega8 microcontroller, that has an identifier ID

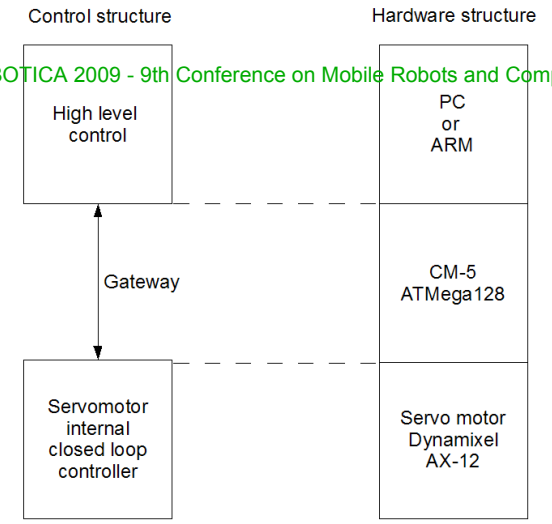


Fig. 2. Architecture levels of real robot

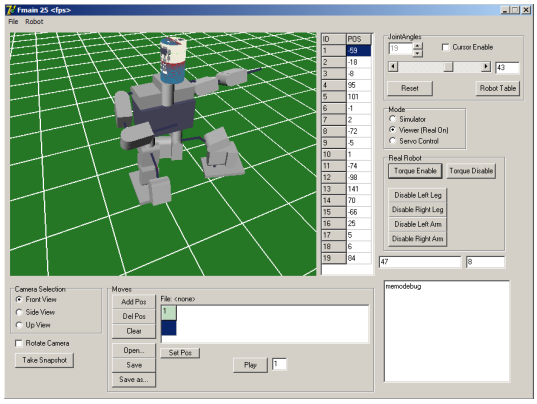


Fig. 3. Control application

and receives commands from the bus shared for all servomotors. It is mainly composed by a DC motor and a PWM driver. Each servo is able to be programmed with not only the goal position, the moving speed, the maximum torque, the temperature and voltage limits but also with the control parameters. These limitations are presented in the simulator for a faithful representation. At the next layer, the CM-5 module interface, allows for data interchange. It receives messages from the upper layer and translates them to the servos bus. Answers from servos are also translated and sent back to the upper layer. Fig. 3 shows the developed high level application that allows to control the real humanoid robot. This application is independent from the simulator.

III. Simulation Model

Design behaviour without real hardware is possible due to a physics-based simulator implementation. The physics engine is the key to make simulation useful in terms of high performance robot control [6]. The dynamic behaviour of robot (or multiple robots) is computed by the ODE Open Dynamics Engine, a free library for simulating rigid body dynamics.

A. Simulator Architecture

The simulator architecture is based on the real humanoid robot. The simulated body masses and dimensions are the same as the real one. The communication architecture in the real robot brings some limitations to control loop such as lag time. The developed simulator implements these properties and the same architecture levels of the real robot are implemented in the simulator. The simulation step is 500 μs and the controller loop period is done at 40 ms. At the lowest level, the servo motor model includes the control loop, just like the real servomotors. At the highest level, some predefined joint states are created based on several methods presented on literature: [7], [8] and [9]. It is also implemented, at the middle level, an optimized trajectory controller that allows to minimize acceleration, speed or energy consumption [2].

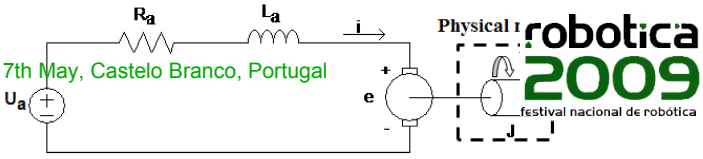


Fig. 4. Servomotor model electric scheme

B. Servomotor Model

The servomotor can be resumed to a DC motor model, presented in Fig. 4 where U_a is the converter output, R_a is the equivalent resistor, L_a is the equivalent inductance and e is the back emf voltage as expressed by (1).

The parameters of the motor can be measured directly or through some experiments: R_a is 8 Ω, L_a is 5 mH and K_s is 0.006705 $V.s/rad$. The motor can supply a T_L torque and load has a J moment of inertia that will be computed by the physical model ODE. Current i_a can be correlated with developed torque T_D through (2) and the back emf voltage can be correlated with angular speed through (3), where K_s is a motor parameter [10].

$$U_a = e + R_a i_a + L_a \frac{\partial i_a}{\partial t} \quad (1)$$

$$T_D(t) = K_s i(t) \quad (2)$$

$$e(t) = K_s \omega(t) \quad (3)$$

In fact, the real developed torque (useful) that will be applied to the load (T_L) is the developed torque subtracted by the friction torque, presented in next subsection.

C. Friction Model

Friction exists in the simulator in two cases: The foot ground interaction and the joint connectivity. The first one, is adjusted so that displacement is the same as reality. The joint friction model becomes from two ways: the static and viscous friction. The first one can be modelled as the sign function (with F_c constant) and the second one can be modelled as a linear function with slope B_v. The final friction model is shown in Fig. 5.

The F_c and B_v constants are found using simulator scanning several possible values minimizing the error with the real system during an arm fall from 90 to 0

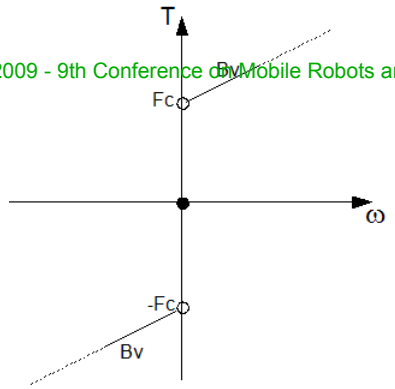


Fig. 5. Friction model: static and viscous

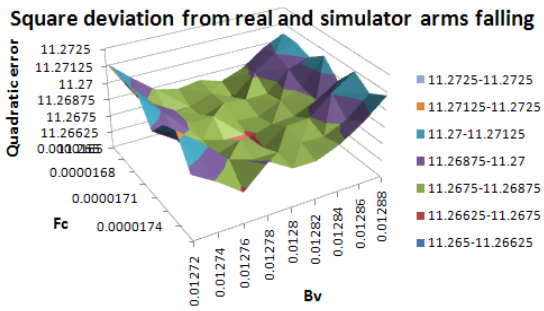


Fig. 6. Deviation from real and simulator frictions constants

$degrees$. Fig. 6 shows the surface of error between real and simulator robots.

As result, B_v=0.01278 $N.m.s/rad$ and F_c=0.0000171 $N.m$ shows the best values. These constants allows simulator to follow reality very close as presented in Fig. 7 where an arm falls from 45, 90 and 135 degrees for both robots.

D. Simulator Validation

A way to validate the humanoid simulation model is to apply the same control signal to both robots and to analyze the behavior. Predefined trajectory states, that allow robot to walk, are based on the Zero Moment Point (ZMP) method and are well described in literature [7] [8] [11]. Fig. 8 shows the sequence during walk movements for both robots (real at left and simulator at right).

It is possible to observe that both robots exhibit a very similar behavior. Furthermore, information from servomotors can be acquired with the developed control

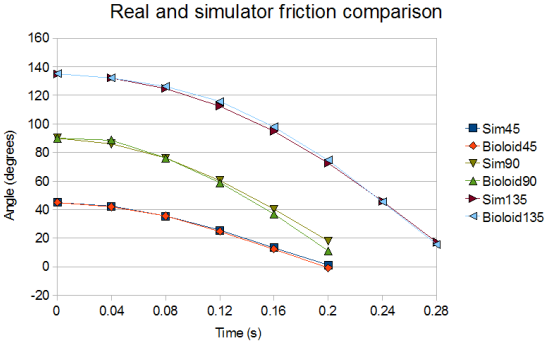


Fig. 7. Real and simulator friction comparison

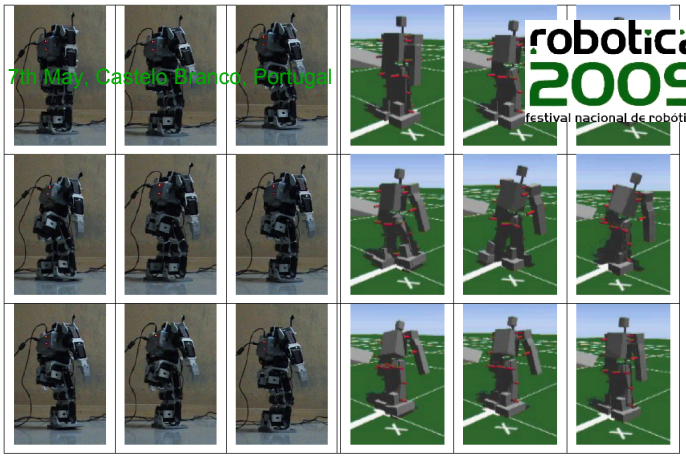


Fig. 8. Real and simulator robots walking with the same predefined gaits

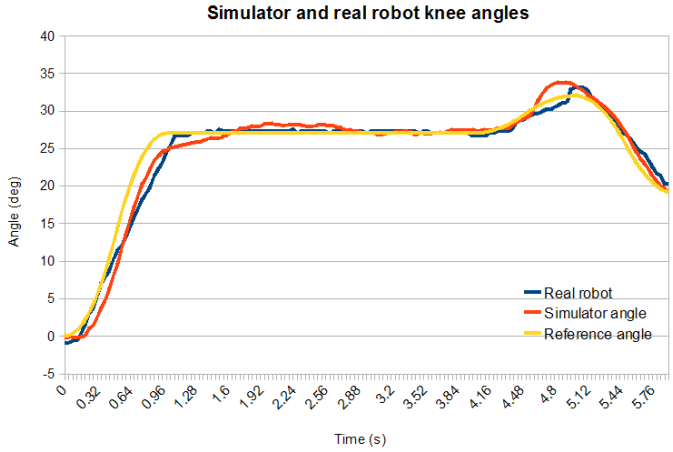


Fig. 9. Simulator and real humanoid robot knee behaviour

application and then compared with the simulator. Fig. 9 shows the knee angle of simulator and real robot for the same reference that seems to be very close. Finally, previous work presents energy consumption comparison for simulator and the real one on get up movements [12].

E. Humanoid Simplified Model

In order to get lower numerical errors during simulations, the total number of connected joints should be reduced. That problem comes from the way that the joint constraints are implemented by the ODE. For the simulation, it can be used a simplified model, presented in Fig. 10, with the same dimensions and weights of the humanoid robot, as only legs are important to this case and other joints are static. So, arms and rotational joints are dropped, resulting in three basic joints: ankle, knee and hip for each leg. The trunk is composed by an oscillating body that maintains the equilibrium during walk movements. Its length depends on the oscillating angle. This model has lower numerical errors in the simulator, as it has fewer articulations.

The presented simplified model will be used in next section.

IV. Energy Efficiency Increase

Having in mind the potential energy (E_p), presented in (4), where m is the body mass, g is the gravitational

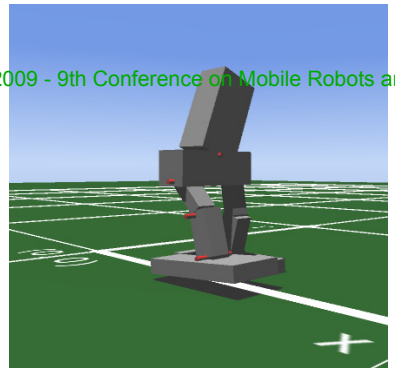


Fig. 10. Simplified model of humanoid robot

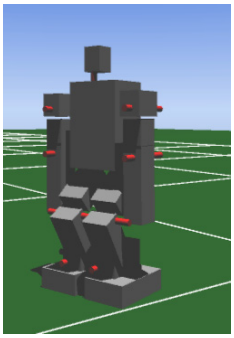


Fig. 11. Humanoid simulator in low level posture

acceleration and h is the height of the body, it can be shown that when the body goes down, it loses its energy that cannot be recovered.

$$E_p = mgh \tag{4}$$

Besides, the energy consumed to keep the robot in a low level posture (usually used during a walk movement as presented in Fig. 11) cannot be neglected due to the R_a resistor from the servomotor model (Joule effect). The power dissipated P_D, during a static pose in R_a resistor can be found in (5) where T_h the holding torque to keep the joint in the desired angle and the T_{ae} is the static friction force. This way, while robot keeps its low posture and when it rises, the consumed energy is provided only from batteries. While moving, the power consumption can be estimated in the simulator through voltage and current product and energy consumption for a movement can be computed through the power integral. This energy consumption can be used to find the optimal characteristics.

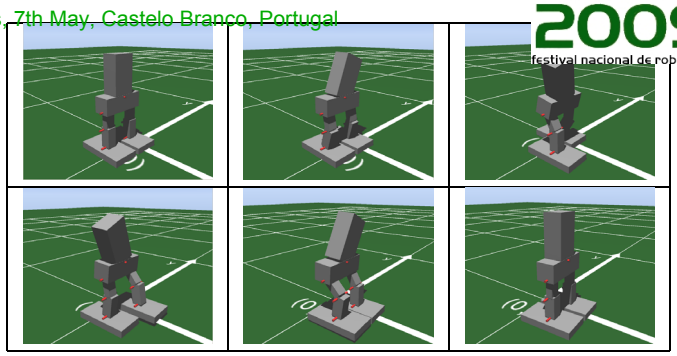
$$P_D = R_a \frac{(T_h - T_{ae})^2}{K_s^2} \tag{5}$$

Next section presents a solution that allows to decrease the power consumption, based on a passive method that increases the energy efficiency. The presented results, that validate the proposed approach, are extracted from a walk movement and based in the estimated energy consumption minimization.

A. Proposed Method

To store the potential energy and assist during low postures, an elastic element can be used. When the robot

TABLE I
SIMPLIFIED MODEL COMPLETE WALKING SNAPSH



posture rises, the elastic element releases the stored energy. This elastic element is composed by a spring and its force can be calculated through $Hooke$ law, as presented in (6), where T_m is the torque, α_m is the torsion angle and k is a spring characteristic. The best spring type is a spiral one due to the nature of the system (torsion). The zero position can be changed and an $offset$ appears.

$$T_m = k(\alpha_m - offset) \tag{6}$$

In order to analyze the energy consumption during movements, the integral of the power must be computed. The power can be calculated by the voltage and current product. The current should only be used when its signal is the same as the voltage, as the power supply and the PWM driver are not regenerative.

B. Optimal Characteristics Computation

In order to compute the optimal characteristics for each spring (added to each joint), the walk movement was implemented for the robot and energy consumption was estimated for each joint. As the spring characteristics to be found, several values for k and $offset$ are implemented and the final energy consumption function allows to find the optimal characteristics at the minimum point. The inter-dependences between parameters can be despised whereas the joints controller keep the desired angle.

Figures presented in table I displays a graphic simulation snapshots of the walking robot model. Once walk movements are not symmetric, energy consumption can be different in legs.

To know the functions surfaces, the k and $offset$ scanning graphics for the minimum energy consumption are presented in figures 12, 13, 14, 15, 16, 17 and 18 for left and right ankles, left and right knees, left and right hips and equilibrium trunk.

Table II presents, in a short way, the optimal characteristics results for each one of the seven joints presented in the simplified robot.

C. Results

As result, it is possible to remark that the use of optimal spring's characteristics in the joints allows to decrease the energy consumption. In this case, 18.8 % of the energy can be saved as presented in table III.

Fig. 19 presents a comparison bar graphic for the energy consumption for each joint, with and without spring.

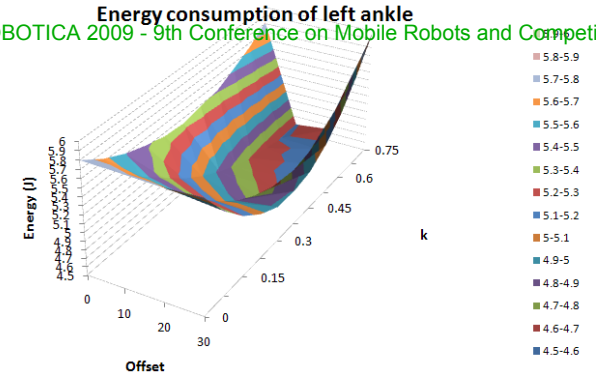


Fig. 12. Energy consumption of left ankle

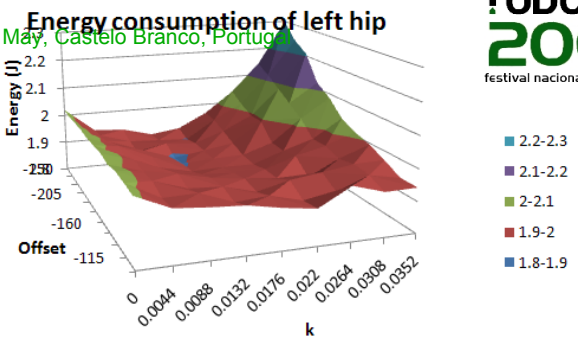


Fig. 16. Energy consumption of left hip

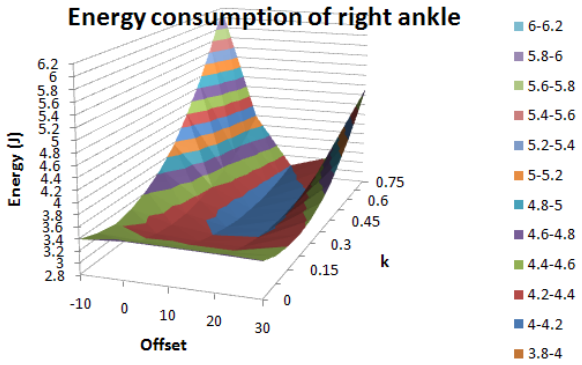


Fig. 13. Energy consumption of right ankle

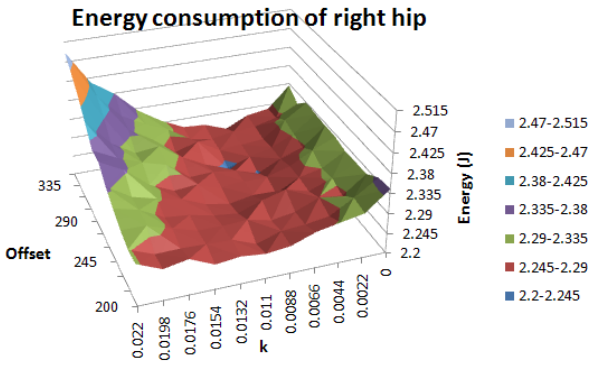


Fig. 17. Energy consumption of right hip

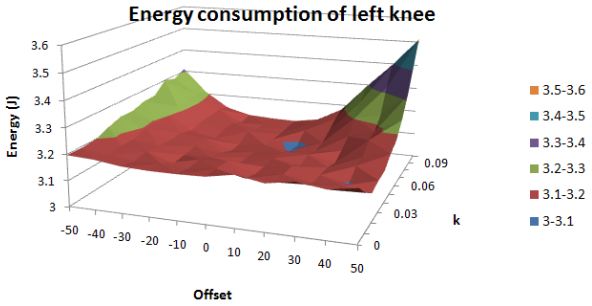


Fig. 14. Energy consumption of left knee

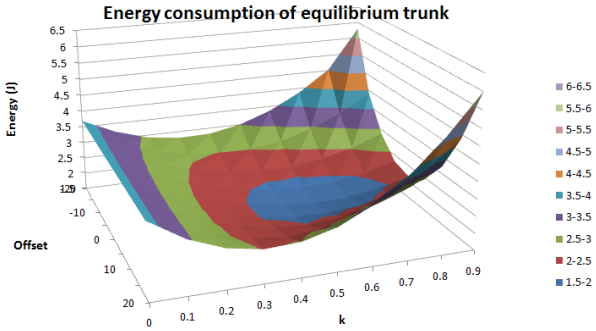


Fig. 18. Energy consumption of equilibrium trunk

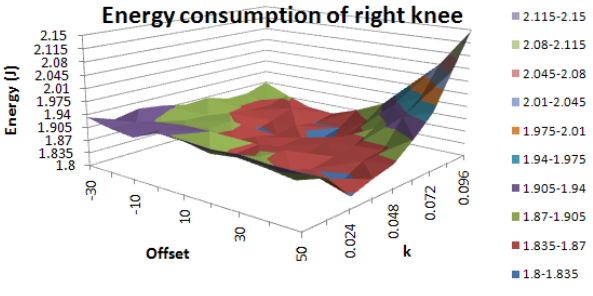


Fig. 15. Energy consumption of right knee

TABLE II
K AND $offset$ OPTIMAL VALUES

Joint	k $(N.m^{-1})$	$offset$ (deg)
Left ankle	0.525	20
Right ankle	0.525	10
Left knee	0.07	10
Right knee	0.072	10
Left hip	0.0131	-205
Right hip	0.0088	305
Eq. trunk	0.6	0

TABLE III

ENERGY CONSUMPTION IN A WALK MOVEMENT PERIOD

Joint	En w/o spr (J)	En w/ spr (J)	Gain (%)
Left ankle	5.79	4.55	21.5
Right ankle	3.42	2.84	17.2
Left knee	3.18	3.09	3.0
Right knee	1.93	1.82	5.8
Left hip	2.02	1.89	6.2
Right hip	2.33	2.24	3.9
Eq. trunk	3.70	1.74	53.08
Total En.	22.38	18.16	18.8

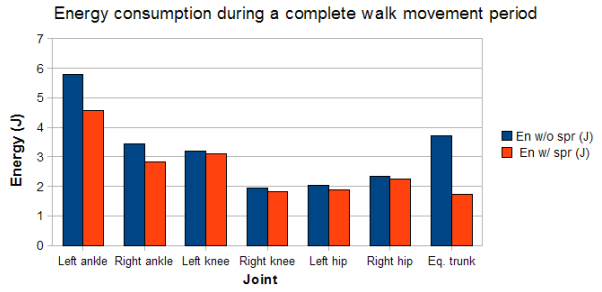


Fig. 19. Energy consumption comparison during a walk movement period

The equilibrium trunk joint is the most perceptible example. Excluding this joint, there is still a 12 % of energy that can be saved with springs. This shows that the use of the suggested springs in humanoid robotics articulations can increase the robot autonomy.

V. CONCLUSIONS

In the presented work, a simulator model of a humanoid platform was developed. Servomotor, friction and dynamic models were developed and validated. The passive system that increases energy efficiency was presented and the energy saving results were shown. The presented approach allows reducing energy consumption in 19 %. The initial results are found to be satisfactory, and improvements are currently underway to explore and enhance the capabilities of the proposed method. Henceforth, its adaptation to a real humanoid joint is the final implementation step.

REFERENCES

[1] Suzuki, T. and Ohnishi, K., Trajectory Planning of Biped Robot with Two Kinds of Inverted Pendulums, Proceedings of 12th International Power Electronics and Motion Control Conference, pp. 396-401, Portoroz, Slovenia (2006).

[2] Lima, J., Gonçalves, J., Costa, P. and Moreira, A., Humanoid robot simulation with a joint trajectory optimized controller, In Proceedings of 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, German (2008).

[3] Tang, Z., Zhou, C. and Sun, Z. Humanoid Walking Gait Optimization Using GA-Based Neural Network, Advances in Natural Computation, pp. 252-261, Springer Berlin/Heidelberg, Changsha, China (2005).

[4] http://www.fe.up.pt/˜paco/wiki/

[5] Wang, X., Lu, T. and Zhang, P., YARP: Yet Another Robot Platform, International Journal of Advanced Robotic Systems, Vol. 3, No. 1, pp. 043-048 (2006).

[6] Browning, B. and Tryzelaar, E., UberSim: A Realistic Simulation Engine for RobotSoccer, Proceedings of Autonomous Agents and Multi-Agent Systems, Melbourne, Australia (2003).

[7] Kajita, S., Morisawa, M., Harada, K., Kaneko, K., Kanehiro, F., Fujiwara, K. and Hirukawa, H., Biped walking pattern generator allowing auxiliary ZMP control, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2994-2999, Beijing, China (2006).

[8] Zhang, L., Zhou, C. and Xiong, R., A lie group for realtime zmp detection using force/torque sensor, In Pr the 11th International Conference on Climbing and Wa and the Support Technologies for Mobile Machines, pp Coimbra, Portugal (2008).

[9] Wang, X., Lu, T. and Zhang, P., State Generation Method for Humanoid Motion Planning Based on Genetic Algorithm, Journal of Humanoids, Vol. 1, No. 1, pp. 17-24, (2008).

[10] Bishop, R., The Mechatronics Handbook, CRC Press, New York (2002).

[11] Meghdari, A., Sohrabpour, S., Naderi, D., Tamaddoni, S., Jafari, F. and Salarieh, H., A Novel Method of Gait Synthesis for Bipedal Fast Locomotion, Journal of Intelligent and Robotic Systems, Volume 53, Number 2, pp. 99-202, Springer Netherlands (2008).

[12] Lima, J., Gonçalves, J., Costa, P. and Moreira, A., Realistic Behaviour Simulation of a Humanoid Robot, 8th Conference on Autonomous Robot Systems and Competitions, Aveiro, Portugal (2008).

robotica 2009
festival nacional de robótica

Orbital Obstacle Avoidance Algorithm for Reliable and On-Line Mobile Robot Navigation

Lounis Adouane
LASMEA, UBP-UMR CNRS 6602, France
Email: Lounis.Adouane@lasmea.univ-bpclermont.fr

***Abstract*— This paper proposes an orbital obstacle avoidance algorithm which permits to obtain safe and smooth robot navigation in very cluttered environments. This algorithm uses specific reference frame which gives accurate indication on robot situation. The robot knows thus if it must avoid the obstacle in clockwise or counter-clockwise direction. Moreover, it knows the moment to go into the orbit of the obstacle and the moment to go out. These orbital behaviors are performed using adaptive limit-cycle trajectories. The later with a specific conflicting situations module permit to prevent robot oscillations, local minima and dead ends. The proposed algorithm is embedded in a specific bottom-up control architecture with stability proof given according to Lyapunov synthesis. The overall proposed structure of control allows to decrease significantly the time to reach the target. In fact, according to the proposed algorithm, robot anticipates the collisions with obstacles according to smooth local trajectory modifications. A large number of simulations in different environments are performed to demonstrate the efficiency and the reliability of the proposed control architecture.**

I. Introduction

Obstacle avoidance controllers have a predominating function to achieve autonomously and safely the navigation of mobile robots in cluttered and unstructured environments. Khatib in [1] proposes a real-time obstacle avoidance approach based on the principle of artificial potential fields. He assumes that the robot actions are guided by the sum of attractive and repulsive fields. Arkin in [2] extends Khatib's approach while proposing specific schema motors for mobile robots navigation. Nevertheless, these methods suffer from the local minima problem when for instance, the sum of local gradient is null. In [3], Elnagar et al., propose to model the repulsive potential field characterizing obstacles by Maxwell's equations which have the merit to completely eliminate the local minima problem. Fuzzy control is widely used to perform robust obstacle avoidance [4], [5]. This formalism allows to integrate several linguistic rules to avoid dead ends or local minima [6]. Unfortunately, its lacks of stability demonstration of the applied control laws. Another interesting approach, based on a reflex behavior reaction, uses the Deformable Virtual Zone (DVZ) concept, in which a robot kinematic dependent risk zone surrounds the robot [7]. If an obstacle is detected, it will deform the DVZ and the approach consists to minimize this deformation by modifying the control vector. An interesting overview of other obstacle avoidance methods is accurately given in [8].

Nevertheless, the obstacle avoidance controller is only a part of the different functions which must constitute an overall control architecture for navigation tasks. One part of the literature in this domain considers that the robot is fully actuated with no control bound and focuses the attention on path planning. Voronoï diagrams and visibility graphs [9] or navigation functions [10] are among these roadmap-based methods. However, the other part of the literature considers that to control a robot with safety, flexibility and reliability, it is essential to accurately take into account: robot's structural constraints (e.g., nonholonomy); avoid command discontinuities and set-point jerk, etc. Nevertheless, even in this method, there are two schools of thought, one uses the notion of planning and re-planning to reach the target, e.g., [11] and [12] and the other more reactive (without planning) like in [13], [14] or [15]. Our proposed control architecture is linked to this last approach. Therefore, where the stability of robot control is rigourously demonstrated and the overall robot behavior is constructed with modular and bottom-up approach [16]. The proposed on-line obstacle avoidance algorithm uses specific orbital trajectories given by limit-cycle differential equations [17], [18]. The proofs of controllers stability are given using Lyapunov functions. The proposed algorithm provides also several mechanisms to prevent oscillations, local minima and dead end robot situations.

The rest of the paper is organized as follows. Section II gives the specification of the task to achieve. The details of the proposed control architecture are given in section III. It presents the model of the considered robot and the implemented elementary controllers laws. Section IV gives in details the proposed obstacle avoidance algorithm whereas section V introduces the conflicting situations management module. Section VI is devoted to the description and analysis of simulation results. This paper ends with some conclusions and further work.

II. Navigation in presence of obstacles

The objective of the navigation task in an unstructured environment is to lead the robot towards one target while avoiding statical and dynamical obstacles. One supposes in the setup that obstacles and the robot are surrounded by bounding cylindrical boxes with respectively R_O and R_R radii [19]. The target to reach is also characterized by a circle of R_T radius. Several perceptions are also necessary for the robot navigation (cf. Figure 1):

- D_{ROi} distance between the robot and the obstacle "i",

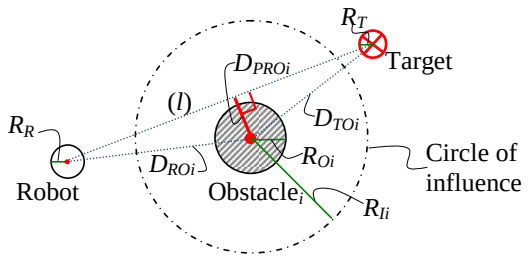


Fig. 1. The used perceptions for mobile robot navigation

- D_{PROi} perpendicular distance between the line (l) and the obstacle "i",
- D_{TOi} distance between the target and the obstacle "i".

For each detected obstacle we define a *circle of influence* (cf. Figure 1) with a radius of $R_{Ii} = R_R + R_{Oi} + Margin$. *Margin* corresponds to a safety tolerance which includes: perception incertitude, control reliability and accuracy, etc.

III. Control Architecture

The proposed structure of control (cf. Figure 2) aims to manage the interactions between elementary controllers while guaranteing the stability of the overall control as proposed in [15]. Its objective is also to obtain safe, smooth and fast robot navigation. It will permit for example to an autonomous application of travelers transportation [20] to have more comfortable displacements of the passengers. The specific blocks composing this control are detailed below.

A. Hierarchical action selection

Most reactive approaches activate the obstacle avoidance controller only when the robot is close to an obstacle (i.e. $D_{ROi} \leq R_{Ii}$) (cf. Figure 3(a)) [2], [21], [22], etc. In contrast, the proposed algorithm 1 activates the obstacle avoidance controller as soon as it exists at least one obstacle that can obstruct the future robot movement toward the target (i.e. $D_{PROi} \leq R_{Ii}$, cf. Figure 1). Thus, while anticipating the activation of obstacle avoidance controller (cf. Figure 3(b)), Algorithm 1 permits to decrease the time to reach the target, especially in very cluttered environments (cf. Section VI).

The proposed control architecture uses a hierarchical action selection mechanism to manage the switch between two or even more controllers. Obstacle avoidance strategy is integrated in a more global control architecture unlike what is proposed in [24]. Otherwise, the controller activations are achieved in a reactive way as in [23] or [16].

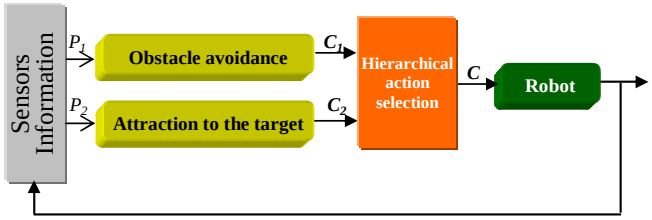
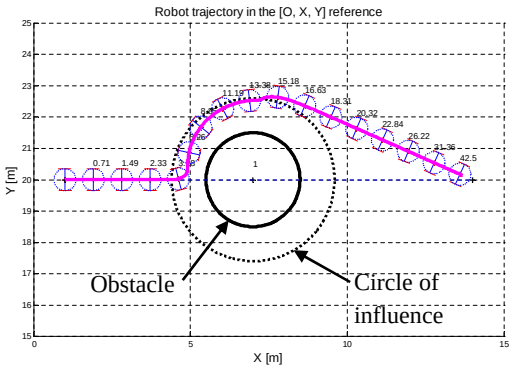
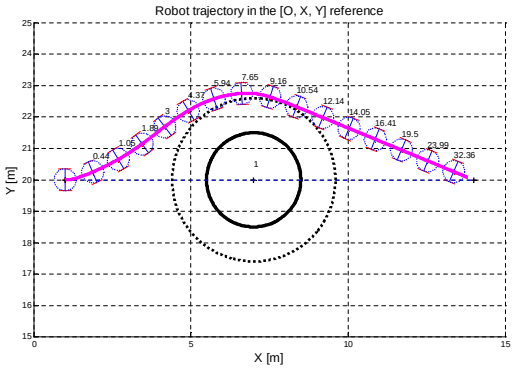


Fig. 2. Control architecture for mobile robot navigation



(a) Without orbital algorithm



(b) With orbital algorithm

Fig. 3. Robot trajectory while anticipating or not the obstacle collision

if *It exists at least one constrained obstacle* {*i.e.*, $D_{PROi} \leq R_{Ii}$ *(cf. Figure 1)* } **then**
  Activate obstacle avoidance controller
else
  Activate the attraction to the target controller
end

Algorithm 1: Hierarchical action selection

B. Elementary controllers

Each controller composing the control architecture (cf. Figure 2) is characterized by a stable nominal law. These laws are synthesized according to Lyapunov theorem. We will present here only some details about the stability demonstration of the used laws. More details are given in [25]. Before describing each elementary controller, let's show the used kinematic robot model (cf. Figure 4):

$$\dot{\xi} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l_2\cos\theta - l_1\sin\theta \\ \sin\theta & -l_2\sin\theta + l_1\cos\theta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} \quad (1)$$

with:

- x, y, θ: configuration state of the unicycle at the point "P_t" of abscissa and ordinate (l_1, l_2) according to the mobile reference frame (X_m, Y_m),
- v: linear velocity of the robot at the point "P_t",
- w: angular velocity of the robot at the point "P_t".

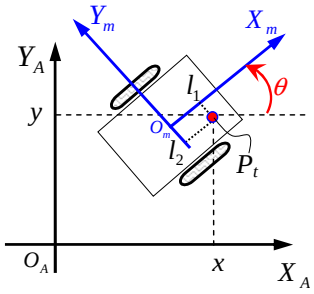


Fig. 4. Robot configuration in a cartesian reference frame

1) Attraction to the target controller: This controller guides the robot toward the target which is represented by a circle of (x_T, y_T) center and of R_T radius (cf. Figure 1). The used control law is a control of position at the point P_t = (l_1, 0) (cf. Figure 4). As we consider a circular target with R_T radius, therefore, to guarantee that the center of robot axis reaches the target with asymptotical convergence, l_1 must be $\leq R_T$ (cf. Figure 4).

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & -l_1 \sin\theta \\ \sin\theta & l_1 \cos\theta \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix} = M \begin{pmatrix} v \\ w \end{pmatrix} \quad (2)$$

with M invertible matrix.

The errors of position are: $\begin{cases} e_x = x - x_T \\ e_y = y - y_T \end{cases}$

The position of the target is invariable according to the absolute reference frame (cf. Figure 6) $\Rightarrow \begin{cases} \dot{e}_x = \dot{x} \\ \dot{e}_y = \dot{y} \end{cases}$

Classical techniques of linear system stabilization can be used to asymptotically stabilize the error to zero [26]. We use a simple proportional controller which is given by:

$$\begin{pmatrix} v \\ w \end{pmatrix} = -KM^{-1} \begin{pmatrix} e_x \\ e_y \end{pmatrix} \quad (3)$$

with $K > 0$. Let's consider the following Lyapunov function

$$V_1 = \tfrac{1}{2} d^2 \quad (4)$$

with $d = \sqrt{e_x^2 + e_y^2}$ (distance robot-target).

Therefore, to guarantee the asymptotical stability of the proposed controller, $\dot{V}_1$ must be strictly negative definite, so, $d\dot{d} < 0$, what is easily proven as long as $d \neq 0$.

2) Obstacle avoidance controller: To perform the obstacle avoidance behavior, the robot needs to fellow accurately limit-cycle vector fields [18], [24], [27], [15]. These vector fields are given by two differential equations:

- For the clockwise trajectory motion (cf. Figure 5(a)):

$$\begin{aligned} \dot{x}_s &= y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= -x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \quad (5)$$

- For the counter-clockwise trajectory motion (cf. Figure 5(b)):

$$\begin{aligned} \dot{x}_s &= -y_s + x_s(R_c^2 - x_s^2 - y_s^2) \\ \dot{y}_s &= x_s + y_s(R_c^2 - x_s^2 - y_s^2) \end{aligned} \quad (6)$$

where (x_s, y_s) corresponds to the position of the robot according to the center of the convergence circle (characterized by an R_c radius). Figure 5 shows that the circle of "$R_c = 1$" is a periodic orbit. This periodic orbit is called a limit-cycle. Figure 5(a) and 5(b) show the shape of equations (5) and (6) respectively. They show the direction of trajectories (clockwise or counter-clockwise) according to (x_s, y_s) axis. The trajectories from all points (x_s, y_s) including inside the circle, move towards the circle.

The proposed control law which permits to follow these trajectories is an orientation control, the robot is controlled according to the center of its axle, i.e., while taking (l_1, l_2) = (0, 0) (cf. Figure 4). The desired robot orientation θ_d is given by the differential equation of the limit-cycle (5) or (6) as:

$$\theta_d = \arctan(\frac{\dot{y}_s}{\dot{x}_s}) \quad (7)$$

and the error by

$$\theta_e = \theta_d - \theta \quad (8)$$

We control the robot to move to the desired orientation by using the following control law:

$$w = \dot{\theta}_d + K_p \theta_e \quad (9)$$

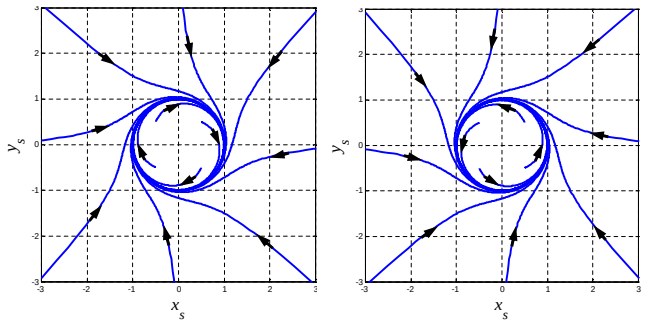
with K_p a constant > 0, $\dot{\theta}_e$ is given then by:

$$\dot{\theta}_e = -K_p \theta_e \quad (10)$$

Let's consider the following Lyapunov function

$$V_2 = \tfrac{1}{2} \theta_e^2 \quad (11)$$

$\dot{V}_2$ is equal then to $\theta_e \dot{\theta}_e = -K_p \theta_e^2$ which is always strictly negative (so, asymptotically stable). It is to note that the nominal speed of the robot v when this controller is active is a constant.



(a) Clockwise (b) Counter-Clockwise

Fig. 5. Shape possibilities for the used limit-cycles

IV. Orbital obstacle avoidance algorithm

In what follows, the overall methodology to achieve the proposed obstacle avoidance algorithm will be given. The algorithm is developed according to stimuli-response principle. To implement this kind of behavior it is important to:

- detect the obstacle to avoid (cf. Section II),
- give the direction of the avoidance (clockwise or counter-clockwise),
- define an escape criterion which defines if the obstacle is completely avoided or not yet.

All these different steps must be followed and applied while guaranteeing that: the robot trajectory is safe, smooth and avoids undesirable situations as deadlocks or local minima ; and that the stability of the applied control law is guaranteed. The necessary steps to carry out the obstacle avoidance algorithm (2) are given below:

1) For each sample time, obtain the distance D_{ROi} and perpendicular distance D_{PROi} for each potentially disturbing obstacle "i" (i.e., $D_{PROi} \leq R_{Ii}$) (cf. Figure 1),
2) Among the set of disturbing obstacles (which can constrain the robot to reach the target), choose the closer to the robot (the smallest D_{ROi}). This specific obstacle has the following features: radius R_{Oi} and (x_{obst}, y_{obst}) position,
3) After the determination of the closest constrained obstacle, we need to obtain four specific areas (cf. Figure 6) which give the robot behavior: clockwise or counter-clockwise obstacle avoidance ; repulsive or attractive phase (cf. Algorithm 2). To distinguish between these 4 areas we need to:
   - define a specific reference frame which has the following features (cf. Figure 6):
     - the X_O axis connects the center of the obstacle (x_{obst}, y_{obst}) to the center of the target. This axis is oriented towards the target,
     - the Y_O axis is perpendicular to the X_O axis and it is oriented while following trigonometric convention.
   - apply the reference frame change of the position robot coordinate $(x, y)_A$ (given in absolute reference frame) towards the reference frame linked to the obstacle $(x, y)_O$. The transformation is achieved while using the following homogeneous transformation:

$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_O = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & x_{obst} \\ \sin\alpha & \cos\alpha & 0 & y_{obst} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_A \quad (12)$$

Once all necessary perceptions are obtained, one can apply the proposed orbital obstacle avoidance strategy given by Algorithm 2. To obtain the set-points, it is necessary to obtain the radius "R_c" and the direction "clockwise or counter-clockwise" of the limit-cycle to follow. The position (x_O, y_O) gives the configuration (x, y) of the robot according to obstacle reference frame. The definition of this specific reference frame gives an accurate means to the robot to know what it must to do. In fact, the sign of x_O gives the kind of behavior which must be taken by the robot (attraction or repulsion). In repulsive phase, the limit-cycle takes different radii to guarantee the trajectory smoothness. The sign of y_O gives the right direction to avoid the obstacle. In fact, if $y_O \geq 0$ then apply clockwise limit-cycle direction else apply counter-clockwise direction. This choice permits to optimize the length of robot trajectory to avoid obstacles. Nevertheless, this direction is forced to the direction taken just before if the obstacle avoidance controller was already active at $(t - \delta T)$ instant (cf. Section V-B).

Input: All the features of the closest obstacle
Output: Features of the limit-cycle trajectory to follow

//I) Obtaining the radius "R_c" of the limit-cycle
if $x_O \leq 0$ **then**
  $R_c = R_{Ii} - \xi$ (**Attractive phase**)
  {*with ξ a small constant value as $\xi \ll$* Margin *(cf. Section II) which guarantees that the robot do not navigate very closely to the R_{Ii} radius (which causes the oscillations of the robot (cf. Figure 9))*}
else
  {*Escape criterion: go out of the obstacle circle of influence with smooth way*}
  R_c = R_c + ξ (**Repulsive phase**)
end
//II) Obtaining the limit-cycle direction
if *obstacle avoidance controller was active at $(t - \delta T)$ instant* **then**
  Apply the same direction already used, equation (5) or (6) is thus applied.
  {*This will permit to avoid several conflicting situations (cf. Rule 2 below)*}
else
  {*The limit-cycle set-point is given by:*}
  $\dot{x}_s = sign(y_O)y_s + x(R_c^2 - x_s^2 - y_s^2)$
  $\dot{y}_s = -sign(y_O)x_s + y(R_c^2 - x_s^2 - y_s^2)$
end

Algorithm 2: Obstacle avoidance algorithm

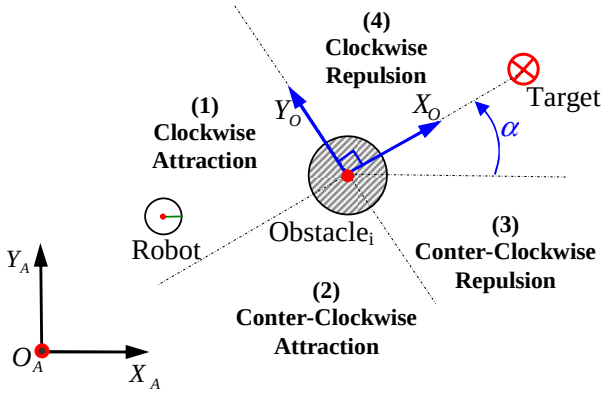


Fig. 6. The 4 specific areas surrounding the obstacle to avoid

V. Conflicting situations management

The good performance of proposed algorithm 2 need to manage some conflicting situations which are due to local minima or dead ends. The rules used to avoid these situations are given below.

A. Rule 1 - What obstacle to avoid?

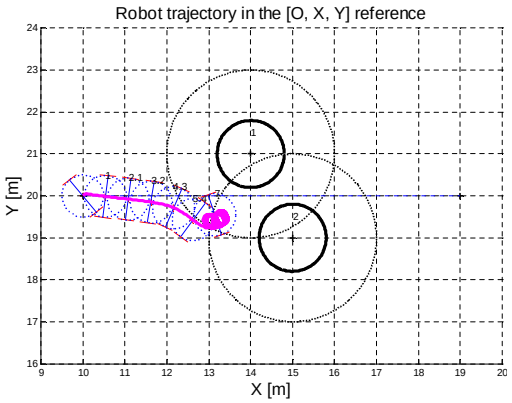
if *Two or more constrained obstacles have the same value of the distance* D_{ROi} *(cf. Figure 1)* **then**
  the robot will choose to avoid the one with the smallest D_{PROi}
end
if *It is already the same* D_{PROi} **then**
  the robot will choose the smallest obstacle D_{TOi} (cf. Figure 1)
end
if *It is already the same* D_{TOi} **then**
  choose arbitrary one of these obstacles
end

Algorithm 3: Rule 1

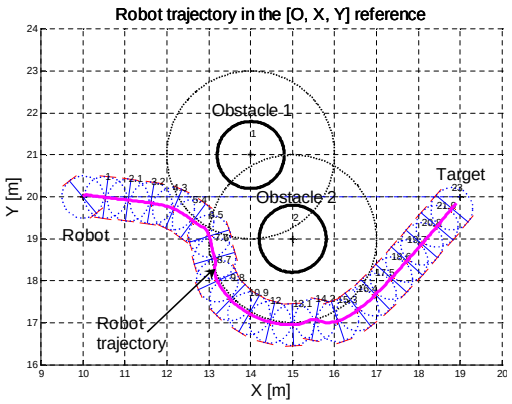
B. Rule 2 - How to avoid local minima and dead ends?

As given in Algorithm 2 (line 9 and 10) the direction of the limit-cycle can be compelled to avoid conflicting situations. This case is given for example when the robot must avoid two or more obstacles with an overlapped region. Figure 7(a) shows what happens to the robot when it do not follows this rule. In Figure 7(b) the robot continues to avoid the obstacle 2 in counter-clockwise according to the rule 2 instead of avoiding it in clockwise direction. Therefore, with this short memory information on the antecedent direction of the avoided obstacle, the robot can perform efficiently its navigation while avoiding this conflicting situation. In [24] authors use, in the same above situation, the definition of a virtual obstacle which contains all the overlapped obstacles, but this method need more time to achieve the obstacles skirting. This is due to the more important distance covered by the robot. In fact, we can easily suppose that when there are two or more overlapped obstacles that the new equivalent virtual obstacle will have a bigger radius than each individual obstacle and this radius will increase according to the furthest obstacles. To illustrate this case, let's take the specific example where a lot of overlapped obstacles are in a straight line. The equivalent virtual obstacle will be given by a very big circle which is not at all justifiable in that obstacles configuration. Moreover, the applied method given in [24] is, in our opinion, less reactive in the sense that it needs more information on the positions of all overlapped obstacles (even if the obstacle doesn't immediately disturb the navigation of the robot), whereas ours permits switching from one obstacle to another according to only reactive rules (cf. Section III-A).

Otherwise, figure 8 gives the robot trajectory when the obstacles are disposed as *U-shape* [28]. This obstacle configuration leads generally to dead end but it is not the case with algorithm 2.



(a) Without imposing the direction, the robot falls in a local minima



(b) While imposing the direction

Fig. 7. Influence of the rule 2

C. Rule 3 - How to avoid trajectory oscillations?

Figure 9 shows the efficiency of the proposed algorithm 2 to avoid the trajectory oscillations when the robot skirts the obstacle. Instruction codes 1 to 7 of Algorithm 2 permits to the robot to do not oscillate between the position where $D_{ROi} \leq R_{Ii}$ (activation of "obstacle avoidance" controller) and $D_{ROi} \geq R_{Ii}$ (activation of "attraction to the target" controller).

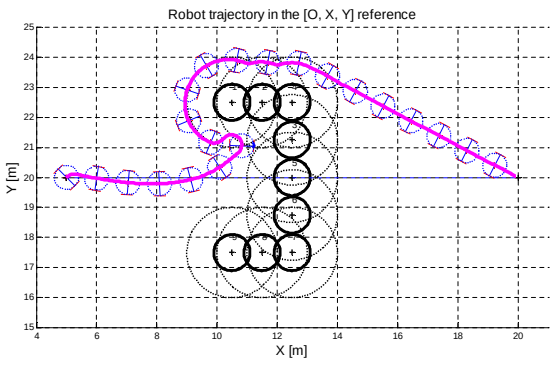
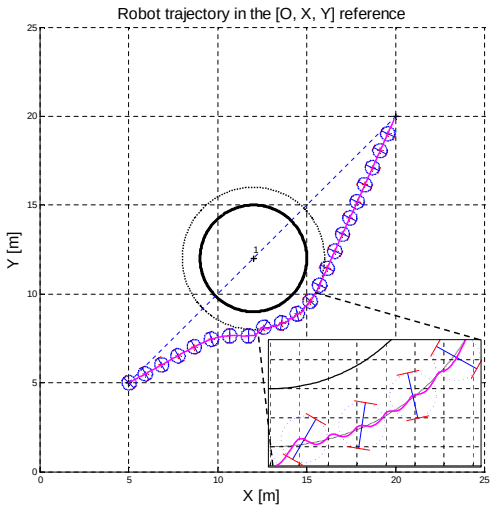
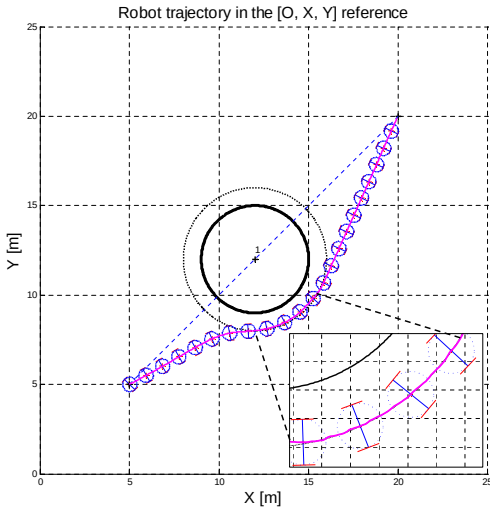


Fig. 8. Experimentation with U-shape obstacles



(a) With oscillations



(b) Without oscillations

Fig. 9. Avoidance of trajectory oscillations when Algorithm 2 is used

VI. Simulations Results

Figure 10 shows the progress value of Lyapunov functions attributed to each controller $V_i|_{i=1..2}$ (cf. Figure 2) when the navigation is performed (cf. Figure 3(a)). These functions decrease asymptotically to the equilibrium point. The demonstration of the stability of the overall proposed structure of control is given in [15].

Otherwise, to demonstrate the efficiency of the proposed obstacle avoidance algorithm, a statistical survey was made while doing a large number of simulations in different cluttered and unstructured environments (cf. Figure 11(b)). We did 1000 simulations with every time 25 obstacles with different positions in the environment. All simulations permits to the robot to reach the target in finite time. These simulations prove also the gain in time given when the orbital method is applied (cf. Figure 11(b)) instead of the one which activates the obstacle avoidance controller only when the robot is inside of the circle of influence (cf. Figure 11(a)). For

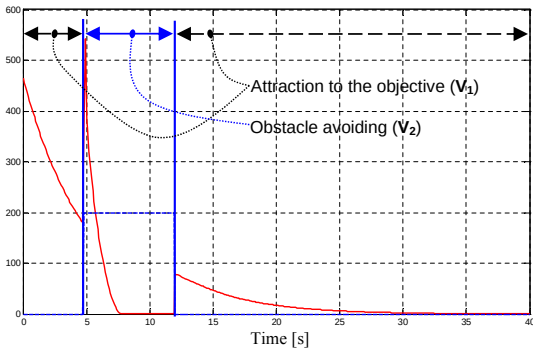
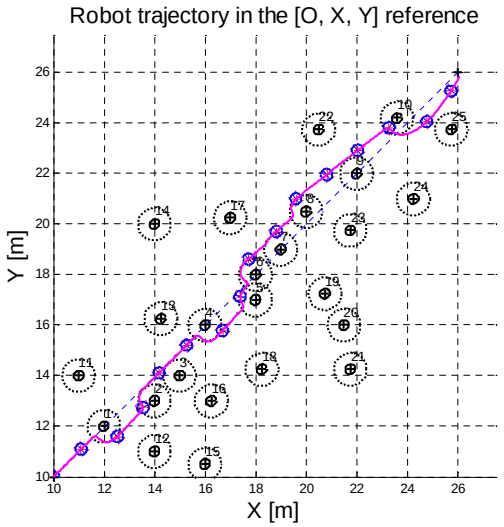
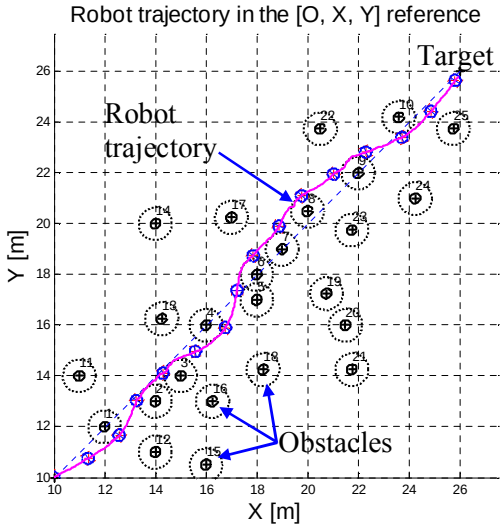


Fig. 10. Evolution of Lyapunov functions for the two used controllers during the robot navigation.

these two simulations (cf. Figure 11(a) and 11(b)) the gain in time is of 8% and the mean time of the 1000 simulations



(a) Without orbital algorithm



(b) With orbital algorithm

Fig. 11. Smooth trajectory obtained with the proposed orbital algorithm

gives an improvement of 6%. The trajectories given by the proposed algorithm are smoother (cf. Figure 11(b)) than those without (cf. Figure 11(a)).

VII. Conclusion and further work

In this paper, an obstacle avoidance algorithm based on orbital limit-cycle trajectories is proposed. This algorithm was embedded in an on-line behavioral control architecture and permits for a mobile robot to navigate in cluttered environments with safe and reliable way. In addition to the use of limit-cycles, the algorithm uses specific reactive rules which allows to the robot to avoid deadlocks, local minima and oscillations. These simple rules are efficient and permits to the proposed algorithm to do not becomes more and more complex. In other terms, the proposed control structure is open and flexible in the sense that it can manage a lot of other conflicts situations while only adding simple reactive rules. Otherwise, the stability proof of the overall control architecture is given. Statistical survey in different environments proves the efficiency and the flexibility of the control. The proposed algorithm allows also to reduce the time needed to reach the target. In fact, according to this algorithm, robot anticipates the collisions with obstacles according to smooth local trajectory modifications. Future work will first test the proposed control architecture on the CyCab vehicle [20]. The second step is to adapt the proposed structure of control to more complex tasks like the navigation in highly dynamical environment.

References

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, pp. pp.90–99, Spring 1986.

[2] R. C. Arkin, "Motor schema-based mobile robot navigation," *International Journal of Robotics Research*, vol. 8, no. 4, pp. pp.92–112, 1989.

[3] A. Elnagar and A. Hussein, "Motion planning using maxwell's equations," in *IEEE/RSJ International Conference on Intelligent Robots and System, Lausanne, Switzerland*, 2002.

[4] A. Saffiotti, E. Ruspini, and K. Konolige, "Robust execution of robot plans using fuzzy logic," in *in Fuzzy Logic in Artificial Intelligence : IJCAI'93 Workshop*, Springer-Velag, Ed., Chambéry-France, 1993, pp. 24–37.

[5] O. Motlagh, T. S. Hong, and N. Ismail, "Development of a new minimum avoidance system for a behavior-based mobile robot," *Fuzzy Sets and Systems*, 2008.

[6] C. Ordonez, E. G. C. Jr., M. F. Selekwa, and D. D. Dunlap, "The virtual wall approach to limit cycle avoidance for unmanned ground vehicles," *Robotics and Autonomous Systems*, vol. 56, no. 8, pp. pp.645–657, 2008.

[7] L. P. Zapata R., Cacitti A., "Dvz-based collision avoidance control of non-holonomic mobile manipulators," *JESA, European Journal of Automated Systems*, vol. 38(5), pp. 559–588, 2004.

[8] J. Minguez, F. Lamiraux, and J.-P. Laumond, *Handbook of Robotics*, 2008, ch. Motion Planning and Obstacle Avoidance, pp. pp.827–852.

[9] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[10] E. Rimon and D. E. Koditschek, "Exact robot navigation using artficial potential flelds," *IEEE Transactions on Robotics and Automation*, vol. 8(5), pp. 501–518, Oct. 1992.

[11] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21(5), pp. 864–874, Oct. 2005.

[12] D. C. Conner, H. Choset, and A. Rizzi, "Integrated planning and control for convex-bodied nonholonomic systems using local feedback," in *Proceedings of Robotics: Science and Systems II*. Philadelphia, PA: MIT Press, August 2006, pp. 57–64.

[13] M. Egerstedt and X. Hu, "A hybrid control approach to action coordination for mobile robots," *Automatica*, vol. 38(1), pp. 125–130, 2002.

[14] J. Toibero, R. Carelli, and B. Kuchen, "Switching control of mobile robots for autonomous navigation in unknown environments," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1974–1979.

[15] L. Adouane, "An adaptive multi-controller architecture for mobile robot navigation," in *10th IAS, Intelligent Autonomous Systems*, Baden-Baden, Germany, July 23-25 2008, pp. 342–347.

[16] L. Adouane and N. Le Fort-Piat, "Behavioral and distributed control architecture of control for minimalist mobile robots," *Journal Européen des Systèmes Automatisés*, vol. 40, no. 2, pp. pp.177–196, 2006.

[17] A. Stuart and A. Humphries, *Dynamical systems and numerical analysis*. Cambridge University Press, 1996.

[18] H. K. Khalil, *Frequency domain analysis of feedback systems*, P. Hall, Ed. Nonlinear Systems: Chapter7, 3 edition, 2002.

[19] M. Yerry and M. Shephard, "A modified quadtree approach to finite element mesh generation," *Computer, Graphics and Applications*, 1983.

[20] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier, "The cycab: a car-like robot navigating autonomously and safely among pedestrians," *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 51–68, 2005.

[21] W. H. Huang, B. R. Fajen, J. R. Fink, and W. H. Warren, "Visual navigation and obstacle avoidance using a steering potential function," *Robotics and Autonomous Systems*, vol. 54, no. 4, pp. 288–299, April 2006.

[22] H. Zhang, S. Liu, and S. X. Yang, "A hybrid robot navigation approach based on partial planning and emotion-based behavior coordination," in *International Conference Intelligent Robots and Systems*, Beijing, China, October 2006, pp. 1183–1188.

[23] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. pp.14–23, March 1986.

[24] D.-H. Kim and J.-H. Kim, "A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer," *Robotics and Autonomous Systems*, vol. 42(1), pp. 17–30, 2003.

[25] L. Adouane, "Hybrid and safe control architecture for mobile robot navigation," in *9th Conference on Autonomous Robot Systems and Competitions*, Portugal, May 2009.

[26] J.-P. Laumond, *La robotique mobile*. Hermès, 2001.

[27] M. S. Jie, J. H. Baek, Y. S. Hong, and K. W. Lee, "Real time obstacle avoidance for mobile robot using limit-cycle and vector field method," *Knowledge-Based Intelligent Information and Engineering Systems*, pp. 866–873, October 2006.

[28] M. Wang and J. N. Liua, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. pp.625–643, 2008.

This page is left blank intentionally

On the learning of inter-field connections in a dynamic field architecture for human-robot collaboration

Emanuel Sousa, Estela Bicho and Wolfram Erlhagen

***Abstract*— There is a growing number of applications for human-robot interaction that require the robot as a social partner rather than a tool controlled by the human. A promising way to design sociable robots is to draw inspiration from studies in cognitive science that investigate the neuro-cognitive principles underlying action selection in a social context. We have recently developed and validated a control architecture for collaborative joint action that reflects these principles. The multi-layered architecture is based on the framework of dynamic neural fields that allows to implement necessary cognitive skills like goal inference, decision making and prediction.**

Here we report about our ongoing research on the learning of connections between the various layers which were hand-coded in the previous robotics applications. We adopt a Hebbian perspective and apply an associate learning mechanism to establish the inter-field connections. We also propose and test an algorithm that generates multiple training inputs with the goal to optimize the learning process. The methodology was implemented in *ARoS*, an anthropomorphic robot built at the University of Minho, and tested in a joint construction task. Our primary experimental results show that the proposed learning process is a valid approach towards designing autonomous robots able to develop and represent new task knowledge with only limited intervention of the human designer.

I. INTRODUCTION

As robots have already started moving out of laboratory and manufacturing environments to share work domain with humans, the design of robots able to interact with humans in a natural and efficient way becomes increasingly important [1]. A promising way to design sociable robots is to draw inspiration from studies in cognitive science that investigate the neuro-cognitive principles underlying social cognition in humans and other primates. Successful collaboration in joint action tasks requires the capacity to infer the action goals of others and to choose an adequate complementary action based on this prediction. In previous work, our group has developed and validated a robot control architecture for action understanding and goal-directed imitation that is based on the idea of motor simulation in the human mirror circuit [2]. More recently we have extended this neuro-inspired architecture to include also the process of action selection in cooperative tasks [3]. Since the inferred action goal of the partner normally does not determine alone the most adequate complementary behavior, other information sources (e.g., shared task knowledge, contextual cues) have to be integrated in the decision process. The architecture consists of several interconnected layers in which each layer is formalized by a dynamic neural field [4], [5], [6]. Activity patterns in the populations of each layer represent high level task knowledge like contextual cues, action means and potential action goals [2], [7]. Activations at a perceptual level (e.g. observing a grasp movement) framed in a specific context (also coded in terms of neural activation) propagate through synaptic connections and produce activation patterns in subsequent layers, that predict the ultimate action goal of the observed motor act (e.g., grasping for placing). This dynamic process can thus be seen as implementing a goal inference capacity. The architecture has been successfully validated in a construction task in which a human and a robot have to jointly assemble a "toy-vehicle", from components distributed on a table [3], [8], [9]. In the experiments, the robot acted not just like a mere servant that simply followed a set of predefined rules but more like a social partner, performing itself parts of the task and anticipating the needs of the human user.

Here, we address the problem of learning the inter-field connections of the control architecture that were in our previous robotics experiments hand-coded. The ultimate goal of this line of research is to increase the capacity of the robot to adapt to changes in joint action tasks in a flexible and efficient manner. This in turn requires that the robot is able to develop representations of task-specific information that are not completely pre-defined by the human designer. In order to advance towards an autonomous learning capacity we adopt a Hebbian perspective [10] frequently applied for unsupervised learning in artificial neural networks. The reason for this choice is twofold. The distributed control architecture based on dynamic neural fields can be seen as a multi-layered neural network for which the connections have to be established during learning and practice. We use a mathematical formulation of Hebb's rule to learn the weights of the synaptic links between dynamic neural fields. Second, correlation based learning is a simple and neuro-plausible learning process. It thus integrates smoothly into our general approach to human-robot cooperation that is inspired by neuro-cognitive mechanisms.

We applied the learning method to the joint construction task for which the dynamic field architecture had

This work was supported by a grant and material support provided under the scope of the projects: i)*LEMI* "Learning to read the motor intention of others: towards socially intelligent robots", (POCI/V.5/A0119/2005, 08/2007-12/2008) financed by FCT; ii) *JAST* "Joint action Science and Technology", project financed by the European Commission, (ref. IST-2-003747-IP); iii)"Anthropomorphic robotic systems: control based on the processing principles of the human and other primates' motor system and potential applications in service robotics and biomedical engineering", financed by FCT and University of Minho, (ref. CONC-REEQ/17/2001).

Emanuel Sousa is with the Department of Industrial Electronics, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal esousa@dei.uminho.py

Estela Bicho is with the Department of Industrial Electronics, University of Minho, Campus de Azurém, 4800-058 Guimarães, Portugal estela.bicho@dei.uminho.py

Wolfram Erlhagen is with the Dept of Mathematics for Sciences and Technology, University of Minho, Guimarães, Portugal wolfram.erlhagen@mct.uminho.pt

already been tested. The results show that the Hebbian learning of the synaptic connections lead to a similar team performance compared to the one achieved when the connections were hand-coded. In addition, we tested variation of specific learning parameters that may affect how social the robot behaves. The selection of a complementary action may reflect the anticipated needs of the user or, alternatively, the robot focuses on first finishing its own assembly steps first.

In order to guarantee an efficient learning process the set of input/output vectors used during training must cover as much context situations as possible. In complex learning situations this normally results in a quite large number of vector pairs. However, for a known task one can exploit that not all information in the scene is equally important. Relevant cues can be distinguished from redundant or irrelevant information. It is therefore possible to design an algorithm that generates input/output pairs by adding the redundant information to the relevant constraints [11]. From a small set of relevant information, a larger number of training pairs is achieved and an efficient training can be accomplished.

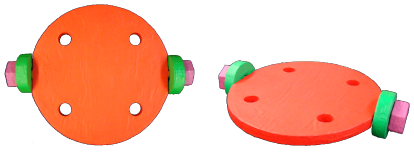
This paper is organized as follows: Section II gives a global overview of the cognitive architecture and the construction task. Section III describes the implementation of the training method and the used rules and algorithms. In section IV the results of the experiments are presented and analyzed. A discussion of the results and future work is made in section V.

II. DNF-based Cognitive Architecture for Joint Action

The distributed control architecture reflects known neuro-cognitive mechanisms underlying the coordination of action and decisions in human joint action. We use the example of the assembly task to explain the functional role of the different layers and the connectivity between the layers. It is important to note, that the architecture can be adapted to other joint action tasks.

A. The joint construction task

The purpose of the construction task is to jointly build a toy (see fig. 1) made of simple parts, namely a base with two plugs, two wheels and two bolts. In the beginning of the task, the components are distributed in the separated working areas of the two teammates. The human and the robot have to attach components at their respective construction side. In addition, the initial distribution of objects in the two working areas makes a direct interaction necessary since handing over processes have to be coordinated among the teammates (see fig. 2 for an example).



(a) (b)

Fig. 1. Final configuration of the object to be constructed by human-robot team.

Fig. 2. Example of task setup, with a human and a robot interacting.

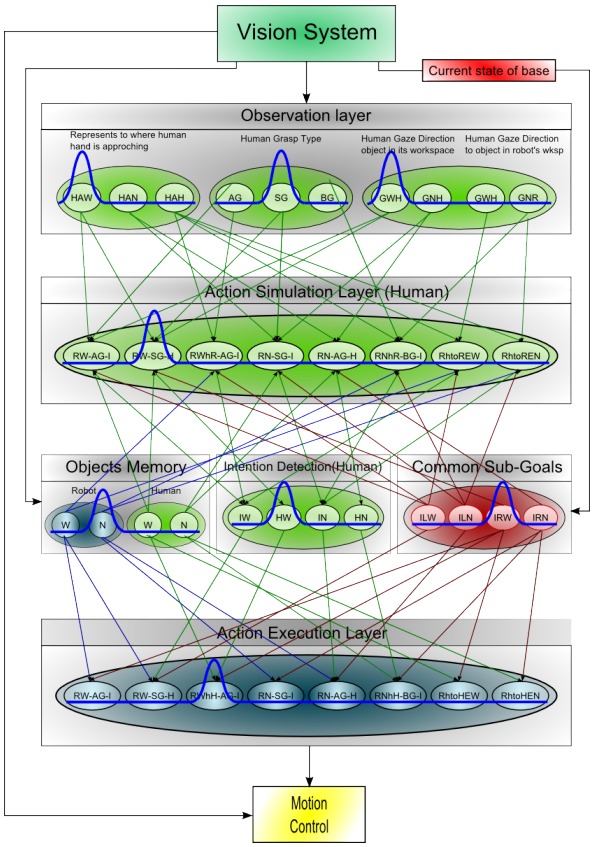


Fig. 3. Control architecture for joint action

B. Description of the architecture

Fig. 3 shows a sketch of the control architecture. It is divided in the following six layers (see [7], [3] for a discussion of the implemented neuro-cognitive mechanisms):

1) Observation Layer: The *Observation layer* codes the perceptual inputs that arrive from the vision system. In the current joint construction task it is divided in three sections: The "Hand approaching" motion that indicates toward where the human partner's hand is approaching; The "Grasp type" that indicates how the human is grasping the object and the "Gaze direction" that indicates towards which object the human is looking.

2) Objects Memory Layer: It encodes which objects are present in the working areas of the human and the robot.

3) Common Sub-goals: In a complex task there are several stages or sub-goals that must be accomplished. This layer codes which ones have already been achieved and which ones still must be completed.

4) Action Simulation Layer - ASL: The activations in the *Observation Layer*, the *Object Memory Layer* and the

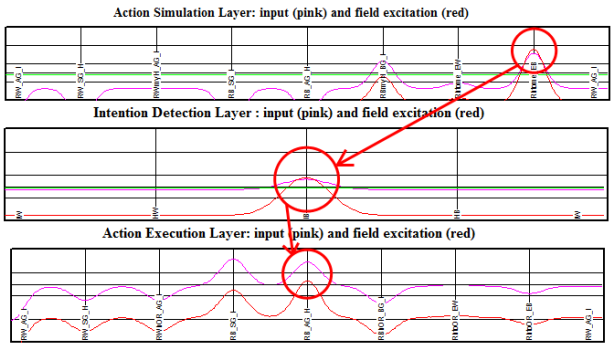


Fig. 4. Example of activation patterns in the layered dynamic field architecture for a handing over sequence. The human user reaches his empty hand towards the robot. As represented by the peak in the *Action Simulation Layer*, the robot interprets this hand gesture as a request for a component. In layer *IDL* the inferred intention of the human user to attach a bolt is represented. This activation pattern evolves in response to the input from *ASL* and the inputs from layers encoding the location of components and the common sub-goals (not shown). The activation peak in *AEL* represents the decision of the robot to grasp a bolt in its workspace with an above grip for handing it over to the human.

Common Sub-Goals Layer propagate through the synapses and produce the input to the dynamic field that formalizes this layer. The result is the simulation of the goal directed action that the human user is performing.

5) Intention Detection Layer - IDL: Intention detection layer codes the predicted intentions. It receives direct input by the Simulation Layer.

6) Action Execution/Selection Layer - AEL: Based on the inputs of the *Intention Detection Layer*, *Common Sub-goals Layer* and *Objects Memory Layer*. This field produces as output a complementary action to the intention manifested by the partner, toward the completion of the task.

It was mentioned already that the layers of the architecture are interconnected by numeric synapses. In typical neural networks, all neurons of the input layer are connected to every neuron of the output layer forming a dense net of connections between the layers. For the present implementation we have simplified the learning of the connection pattern by assuming that connections are learned only between specific field neurons. As illustrated in fig. 3, in each layer localized activation peaks encode task-relevant information. In the *Intention Layer*, for instance, peaks may occur at four different field locations encoding the possible intentions of the partner. Similarly, in the *Action Simulation Layer* eight different sequences are stored. It is thus sufficient to learn the connection and their weights between a limited number of neurons which greatly facilitates the learning process. We model the localized input to a connected field by using a Gaussian function centered on the neuron devoted to learning. The amplitude of the Gaussian is defined by the activity which propagates through the "'synaptic"' connection.

Fig. 4 shows an example of an activation of the layers. One can see that the *Action Simulation layer - ASL*, and the *Action Execution Layer - AEL* have both eight labels matching eight possible activations on each one. The *Intention Detection Layer - IDL* has four subpopulations. Next we show how the Hebbian principles are used for establishing the inter-field connections and their weights.

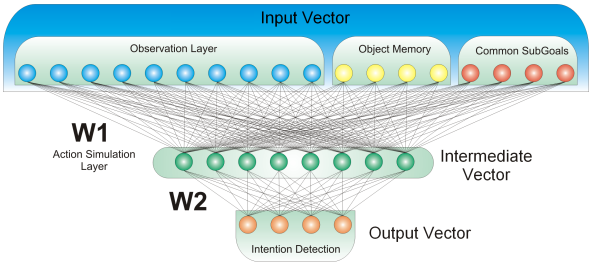


Fig. 5. Neural Networks W_1 and W_2. W_1 (18×8), has an input vector that results from the concatenation of the *Observation Layer* (V_{OL}), the *Objects Memory Layer* (V_{OML}) and the *Common Sub-goals Layer* (V_{CSG}). The output vector is constituted by the *Action Simulation Layer* (V_{ASL}). W_2 (8×4), connects the *Action Simulation Layer* with the *Intention detection Layer*(V_{IDL})

III. Setting inter-field connections via Hebbian Learning

The architecture was implemented on a regular desktop computer that controls the physical actuators of the robot. The programming language used was *C++*.

A. Networks

For implementing the learning scheme for establishing the inter-field connections three neural networks were defined, each formalized by a connection matrix (Fig. 5 and 6). W_1 links the input vector V_{input} with a intermediate vector V_{ASL}. It receives task context information together with perceptual cues and outputs an internal simulation of the goal directed action. The mathematical formulation is given by:

$$\begin{aligned} V_{ASL} &= W_1 \times V_{input} \\ V_{input} &= [V_{OL}|V_{OML}|V_{CSG}] \end{aligned} \tag{1}$$

The second network W_2 links the intermediate vector to the output vector V_{IDL}, meaning that for each action simulation it determines the corresponding intention. The output is given by:

$$V_{IDL} = W_2 \times V_{ASL} \tag{2}$$

The third network, W_3 receives an input vector that joins information about the inferred intentions and task context. The output vector represents a choice of a complementary action.

$$\begin{aligned} V_{AEL} &= W_3 \times V_{middle} \\ V_{middle} &= [V_{IDL}|V_{OML}|V_{CSG}] \end{aligned} \tag{3}$$

B. Hebbian Learning

There are several variations of the Hebbian rule that could be, in principle, applied to train the synaptic links represented in the connection matrices. For the present implementations we opted for using the simplest version of a correlation-based rule:

$$W_{ij} = W_{ij}^{previous} + \Delta W_{ij} \tag{4}$$

where the weight variation is given by:

$$\Delta W_{ij} = \eta x_{ki} y_{kj} \tag{5}$$

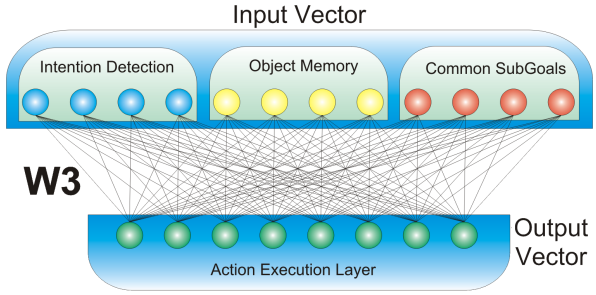


Fig. 6. Neural Network W_3 (12×8). The net receives as input the concatenation of the *Intention Detection Layer*, the *Common Subgoals Layer* and the *Objects Memory Layer*. The output defines the *Action Execution/Selection Layer* (V_{AEL})

where x_{ki} and y_{kj} are the values of the vectors x_k and y_k of indexes i and j respectively. η is a positive constant normally designated as *learning rate*. Written in the matrix form the result is:

$$W = W^{previous} + \eta Y_k X_k^\top \tag{6}$$

This simple rule has known drawbacks like for instance the lack of a decaying term that would prevent from unlimited weight growths [12]. However, the specific application allows its use, primarily, since the total number of training pairs is known. It is then possible to choose an adequate value of η that avoids large values in the connection weights. Also, the shape of the inputs is known and there is no noise to cause irregular growths.

Other rules like the *Instar* [13] and *Ojas's Rule* [14] were also tested. However, their performances with respect to the learning speed were weaker.

C. Training

Equation 6 was then applied to the three matrices and the following expressions were obtained:

$$\begin{aligned} W_1 &= W_1^{prev} + \eta \times V_{ASL}^k \times \left[V_{OL}^k | V_{OML}^k | V_{CSG}^k\right]^\top \\ W_2 &= W_2^{prev} + \eta \times V_{IDL}^k \times {V_{ASL}^k}^\top \\ W_3 &= W_3^{prev} + \eta \times V_{AEL}^k \times \left[V_{IDL}^k | V_{OML}^k | V_{CSG}^k\right]^\top \end{aligned} \tag{7}$$

To train the network with as much possible input vectors an algorithm was developed for generating the multiple vectors based on information about the relevance of the active neurons for the output. For each output there is a set of possible input vectors. In each set, some input neurons are always active and others always inactive. The remaining are redundant, meaning that they can be both active or inactive. For each output the algorithm receives an informative vector indicating which neurons must be active or inactive and which are redundant. The resulting vectors that are then used for training, are similar in the relevant neurons and cover all combinations of non-relevant activations. Fig. 7 shows the algorithm. The task of programming the robot proved to be easier by using this method. The algorithm for generation of multiple combinations outputted 160 combinations for the network W_1, based only on eight given training vectors. Also, this method avoids the pre-establishment by hand of hundreds of connection weights.

IV. RESULTS

A set of experiments were conducted in order to test the resulting performance during the execution of the task. The construction process was repeated several times with different initial setups. The purpose was to check if the process of inference is accurate and capable of determining the correct partner intentions and, if the chosen complementary actions are adequate.

A. Goal inference and choice of complementary actions

For the first experience both the human and robot had a wheel and a bolt in their workspace. Fig. 8 shows the activations of the fields in *Action Simulation Layer*, *Intention Detection Layer* and *Action Execution layer*. In fig. 8(a) one can see the activation patterns resulting from the observation of the human grasping a wheel with an "Above Grip". In layer *ASL* the action sequence "RW-AG-I" (reach towards wheel, grasp it with an above grip) is represented which is linked to intention of the human user to insert a wheel represented by the activation peak in layer *IDL*. Based on the inferred goal, the robot selects the identical action sequence "RW-AG-I" as a complementary behavior as shown in layer *AEL*. Fig. 8(b) shows the pattern of coordinated team behavior for the bolts that are used to fix the wheels on the axle of the platform. The human grasps a bolt in his workspace with a "Side Grip", this activates the respective action sequence in *ASL* linked to the associated goal "Insert Bolt" in layer *IDL*. Since the robot has a bolt in its own workspace, it decides to copy again the observed action sequence for inserting a bolt. Fig. 9 illustrates a trail in which the initial distribution of components in the two working areas is different. The human has two wheels and a bolt in his workspace whereas the robot has only a bolt. First (fig. 9(a), the human grasps a wheel with an "Above Grip" An activation pattern appears in *ASL* representing the goal-directed action sequence "RW-AG-I" linked to the goal "Insert Wheel" in *IDL*. Since the robot has no wheels in its workspace the adequate complementary behavior of the robot is now a request for a wheel by reaching its hand towards the teammate (see the activation peak "RhtoHEW" in layer *AEL*). Next (fig. 9(b)) the human grasps another wheel in his workspace but now with a "Side Grip", which is the most comfortable and

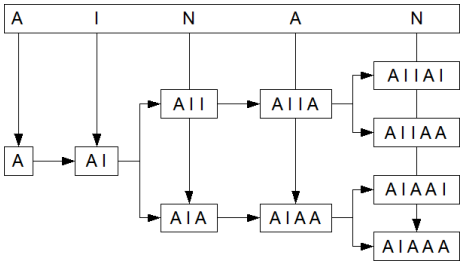
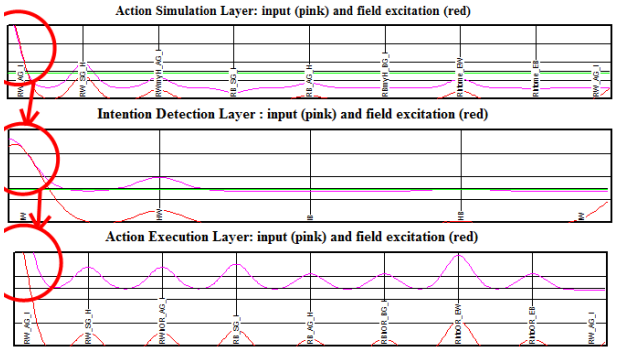
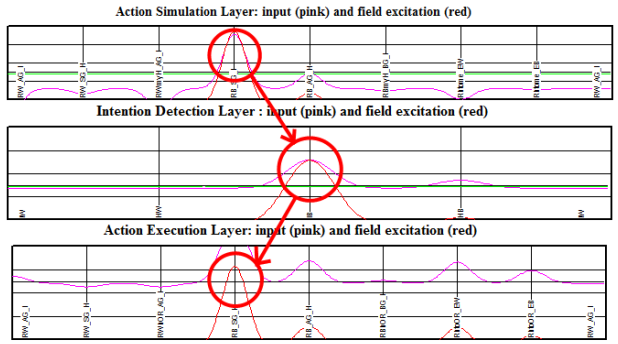


Fig. 7. Algorithm for generation of the input vectors: "A" means "Active", "I" means "Inactive" and "N" means "Non-relevant". The special vector is ran and copied for the new vectors. Every time a "Non-relevant" appears the generated vectors are duplicated and the resulting vectors receive an "A" and a "I" respectively.

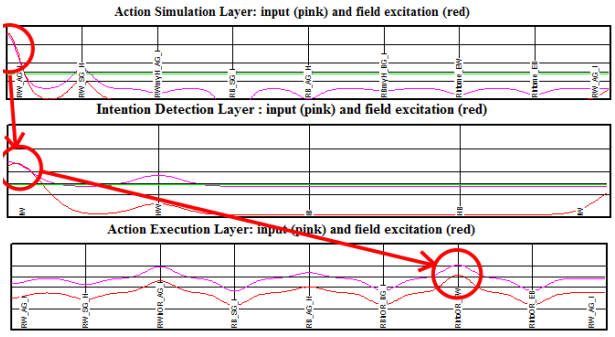


(a) The human grasps a wheel with an "Above Grip". In *ASL*, the sequence "RW-AG-I" is activated (Reach wheel in above grip to insert). The *IDL* produces the output "Insert Wheel". TThe dynamics in the field architecture produces the correct complementary action 'RW-AG-I" in *AEL* and the robot grasps a wheel in its workspace to attach it.

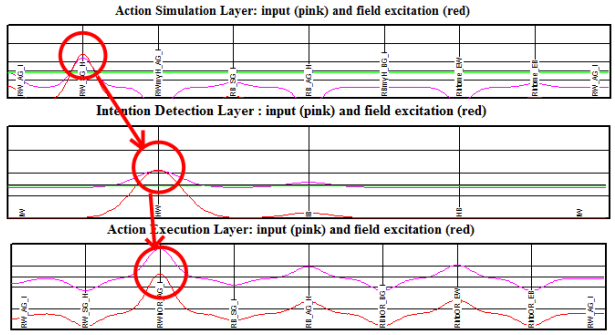


(b) The human grasps the bolt with a "Side Grip". The sequence "RB-SG-I" is activated in *ASL* (reach bolt in side grip to insert). The *IDL* produces the output "Insert Bolt". The system outputs the correct complementary action 'RB-SG-I" in *AEL* and the robot grasps his own bolt to insert.

Fig. 8. Layers activation: Both the human and the robot have wheel and a bolt. There is no need to share parts.



(a) The human grasps a wheel with an "Above Grip". In *ASL* the sequence "RW-AG-I" is activated (Reach wheel in above grip to insert). The *IDL* produces the output "Insert Wheel". The robot has no wheel so it outputs the action 'RhtoHEW" in *AEL* and the robot asks the human for a wheel.



(b) The human grasps another wheel in his workspace but now with a side grip. In *ASL* the sequence"RW-SG-H" becomes activated (Reach wheel in side grip to handover). The *IDL* produces the output "Handover Wheel". The complementary action chosen in *AEL* is "RWhH-AG-I" and the robot reaches towards the object in the hand of the partner with the intention to grasp and attach it.

Fig. 9. Layers activation: Only the human has wheels. The robot asks for a wheel and the human delivers it.

secure way to hand it over. The respective action sequence ("RW-SG-H") is represented in layer *ASL*. Through learned synaptic connections the representation in *ASL* activates the associated goal ("Handover Wheel") in *IDL*. The inferred intention of the human user in turn activates the complementary action sequence in layer *AEL* of the robot that triggers the grasping of the wheel with an above grip for attaching it to the platform ("RWhH-AG-I")

B. "Selfish" versus "social" behavior

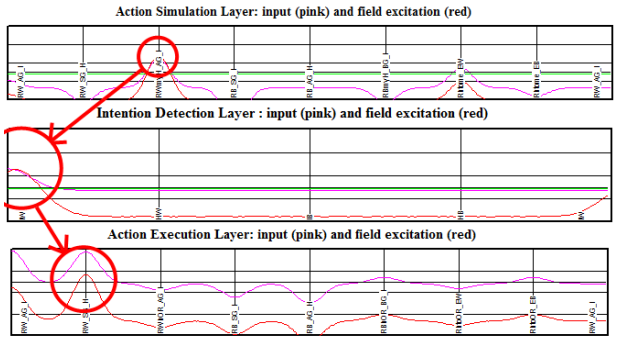
An important aspect that was verified during the tests was the possibility of changing ARoS's "personality". More concretely, we were able to control the way how the robot reacts to situations where different courses of action are possible. For instance, when only the robot has wheels, the human requests one by extending his hand. Two response actions can be produced by the control architecture: Delivering a wheel to the human as requested or, ignore the request and insert a wheel on its side. When training the nets with the same factor η for all the associations, *ARoS* behavior was "selfish" and the activation in the *AEL* was "RW-AG-I" (fig. 10(b)). However, each of the eight possible outputs of the network is trained separately, so it is possible to use different values of η. By increasing the value of η for the actions of "Handover" ("Handover Wheel" and "Handover Bolt") it was possible to change *ARoS* behavior and make it more "Generous" (figure 10(a)).

V. DISCUSSION

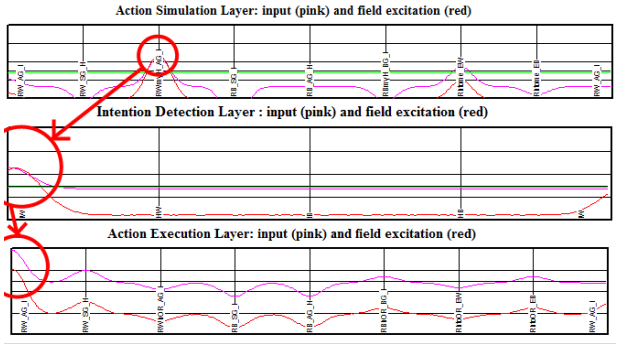
Learning plays and extremely important role in human interactions. It allows us to adapt to others behavior, interpret their actions and anticipate their intentions. Endowing robots with these abilities will allow a better human-machine interaction by making them more adaptable, capable of learning new tasks and producing changes in the knowledge of the older ones.

Thus far, the learned distributed system of dynamic fields for the joint construction task proved to produce reliable results in all tested scenarios. Most importantly, the capacities to infer goals and to select adequate complementary actions were established using the Hebbian learning rule. Moreover, it was shown that it is possible to change *ARoS* "personality" by making it act more "generously" or more "selfishly" by simply adjusting the learning factor that define the strength of the connection weights.

An important question that might come up is why to train the nets with all the possible vectors. The reason can be seen by analyzing Hebb's rule. It says that when two neurons are activated together the connection between them strengthens. Each vector has several neurons. For a given output vector there may be several possible input vectors. Within those, some neurons are always active and



(a)



(b)

Fig. 10. Layers activation: "Selfish" vs "Generous" behavior. In the initial task setup, the human has no wheels, so he asks *ARoS* for a wheel. Depending on the values of the η constants the system's response maybe "generous" - handover a wheel (a) or "selfish" - insert his own wheel (b)

others may or may not be active. By giving all the possible pairs, it is statistically guaranteed that connections are strengthened according to the exact frequency with which they become activated.

The presented learning scheme requires that the programmer has complete knowledge of the task in order to be able to indicate correctly which input actions are relevant for each output. That is necessary, in order to establish the correct training combinations. Such fact constitutes a limitation regarding the complexity of tasks that can be trained into the architecture. On the other hand, all the training is made during the programming stage and the connection weights cannot be altered posteriorly, meaning that it is not possible to update the robot knowledge without reprogramming it.

To overcome these limitations is a major concern in many of the current robotics applications. A possible solution is to endow robots with the ability of learning on-line by observing and imitating an experienced teacher (e.g., [15], [2]). Social learning is considered in general a powerful means to restrict the normally huge search space in which a solution for a particular problem has to be found. We are planning to further explore the idea of imitation learning with the robot *ARoS* in cooperative human-robot tasks, exploiting the new insights about Hebbian learning in dynamic field architectures achieved in the present study. However, we expect that some adjustments to the learning rule have to be made. Continuous on-line learning and adaptation require that competitive and/or decrease factors have to be integrated into the learning process to guarantee a flexible adjustment of weights and to prevent from unlimited weight growth. Therefore variations of the Hebbian rule like the ones proposed in [13] and [16] will be considered.

VI. ACKNOWLEDGMENTS

The authors gratefully acknowledge the contribution of Eliana Silva, Flora Ferreira, João Ferreira, Joaquim Silva, Luís Louro, Manuel Pinheiro, Nzóji Hipólito, Rui Silva and Toni Machado.

REFERENCES

[1] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 143–166, 2003.

[2] W. Erlhagen, A. Mukovskiy, E. Bicho, G. Panin, C. Kiss, A. Knoll, H. T. van Schie, and H. Bekkering, "Goal-directed imitation for robots: A bio-inspired approach to action understanding and skill learning," *Robotics and Autonomous Systems*, vol. 54, no. 5, pp. 353–360, 2006.

[3] E. Bicho, L. Louro, N. Hipólito, and W. Erlhagen, "A dynamic neural field architecture for flexible and fluent human-robot interaction," in *Proceedings of the 2008 International Conference on Cognitive Systems*. Karlsruhe, Germany: University of Karlsruhe, April 2008.

[4] S. Amari, "Dynamic of pattern formation in lateral inhibition type neural fields," *Biological Cybernetics*, vol. 87, pp. 27–77, 1977.

[5] K. Kishimoto and S. Amari, "Existence and stability of local excitations in homogeneous neural fields," *Journal of Mathematical Biology*, vol. 7, pp. 303–318, 1979.

[6] G. Schöner, M. Dose, and C. Engels, "Dynamics of behavior: Theory and applications for autonomous robot architectures," *Robotics and Autonomous Systems*, vol. 16, no. 2-4, pp. 213–245, 1995.

[7] W. Erlhagen and E. Bicho, "The dynamic neural field approach to cognitive robotics," *Journal of Neural Engineering*, vol. 3, p. 3654, 2006.

[8] E. Bicho, L. Louro, N. Hipólito, and W. Erlhagen, "A dynamic field approach to goal inference and error monitoring for human-robot interaction," *New Frontiers in Human-Robot Interaction*, submitted.

[9] R. Silva, "Design e construo de um robot antropomrfico," Master's thesis, University of Minho, 2008.

[10] D. O. Hebb, *The organization of behavior*. Cambridge, MA, USA: Lawrence Erlbaum Associates Inc, 1949.

[11] E. Sousa, "Learning in a dynamic field based architecture for human-robot interaction: A hebbian perspective," Master's thesis, University of Minho, September 2008.

[12] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston: PWS Publishing Company, 1996.

[13] S. Grossberg, "Classical and instrumental learning by neural networks," *Progress in Theoretical Biology*, vol. 3, 1974.

[14] E. Oja, "A simplified neuron model as a principal component analyzer," *Journal of mathematical biology*, vol. 15, pp. 267–273, 1982.

[15] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Sciences*, vol. 6, no. 11, pp. 481–487, 2002.

[16] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, pp. 459–473, April 1989.

An Artificial Low Cost Robot Arm System Playing Tic-Tac-Toe

Mehmet Serdar Guzel[1] Yasin Hinislioglu[2]

[1] School of Mechanical and Systems Engineering Newcastle University
[2] Knowledge and Information Technologies, Ahmet Yesevi University

[1] e-mail : m.s.guzel@newcastle.ac.uk [2] e-mail : yaskil@gmail.com

Abstract

This paper presents an intelligent robotic software system. It is proposed to a Robot arm system, equipped with a simple web camera, and has an appropriate AI (Artificial Intelligent) engine to play Tic-Tac-Toe game against a human opponent dynamically. Three key challenges were investigated during this project. Firstly A simple but efficient image processing techniques is used for robot vision system. Secondly A min-max tree algorithm is implemented as the primary control of the proposed system to determine the next best move. At the last stage a special kinematic solution is developed for kinematic characteristic of system. Consequently, a full automated system has been created which plays a simple board game with a human opponent. Robot arm works completely automated without human contribution during the game session

Keywords: Tic Tac Toe, Lynx-6, Kinematics, Image Processing, Software, Robot Arm., Min Max, DOF

1. Introduction

Board games are the games played on the board with certain pieces, which are moved across the board. It is a common knowledge that simple board games are considered to be the perfect entertainment for families because these games are known to provide fun to people of all ages. Some well-known board games (for example, chess, checkers, Tic-Tac-Toe) possess intense strategic value. Many computers based board game systems have been developed in the last 30 years. In this study a flexible and low cost Robotic system is introduced for playing tic-tac-toe against human opponent. Some systems and techniques have been improved so far for to play Tic-Tac-Toe within manipulators. Vuittonet and Gray worked on a Lego robot, their study describes the Java-based development of a set of robots that coordinate to play the game of Tic-Tac-Toe [1]. On the other hand Soares and Goncalves improved a system playing Tic-Tac-Toe game against a robot manipulator using a vision system [2].

Moreover many AI and Neural Network approach have been studied so far for playing Tic-Tac-Toe. Sungur and Ugur worked on optimizing neural networks for playing Tic-Tac-Toe. They worked on Hopfield network, Boltzmann machine and Gaussian machine and the performances of the models were compared through simulation [3]. Siegel worked with Artificial Neural Network (ANN) to play Tic Tac Toe. The learning method was reinforcement learning [4]. Chellapilla and Fogel worked to describe efforts of hybridize neural and evolutionary computation to learn appropriate strategies in zero and nonzero-sum games, including the Tic-Tac-Toe and Checkers [5]. This paper presents a new approach for Tic Tac Toe playing systems. Kinematics Analysis of a 5 DOF (Degree of Freedom) Lynx robot arm is amended with both Artificial Intelligence algorithms and Image Processing techniques during this proposed system. Proposed software package supports robot vision via web cameras and uses Artificial intelligence techniques for Tic-Tac-Toe game. This low cost 5-DOF robotic arm is controlled completely by a computer, without human input. The arm has supported by vision system (a web cam) connected to pc over USB port. System processes images via web cam and identifies game objects and their physical locations with developed algorithms. An unbeatable artificial intelligence method is used for the AI Engine of the system. Flexibility of the system is directly related to

design of software which is able to support different kinds of robots (6 DOF, 5 DOF, Hexapod, Biped etc) with plug-ins. Also different kinds of connection methods are supported, such as Bluetooth, com, USB etc. Lynx-6 robot arm which is shown in figure 1 has 5 DOF with a grip movement. It is similar to human arm from the number of joints point of view. These joints produce shoulder rotation, shoulder back and forth, elbow, and wrist up and down, wrist rotation and gripper motion [6].

Figure 1: Five Dof Lynx-6 Robot Arm

2. System Organization

Tic-Tac-Toe is a two player board game which is sometimes referred to as a noughts and crosses, the game is played on a board consisting of 9 cells arranged as a 3×3 square, i.e. three rows and three columns. **O** and **X**, who take turns marking the spaces in this 3×3 grid, usually **X** going first. The player who succeeds in placing three respective marks in a horizontal, vertical or diagonal row wins the game. A typical game board is shown in **figure 2**.

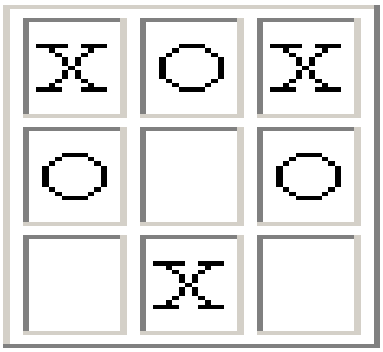


Figure 2: Tic-Tac-Toe Game Board

Proposed system is shown in figure 3. ,in which there is a Lynx-6 Robot arm equipped with an electro magneto, a simple web camera and a game board having metallic pieces. In each robot turn, the arm determines the user movement and its physical coordinates via developed image processing algorithm. After detection step system generates the best movement via AI engine based on this input and relocates itself by the help of inverse kinematic equations. Every movement of robotic arm is calculated dynamically in run time phase. Robotic arm holds the game objects by the help of the electro magnet. This programmable electro magnet is also designed for this project.

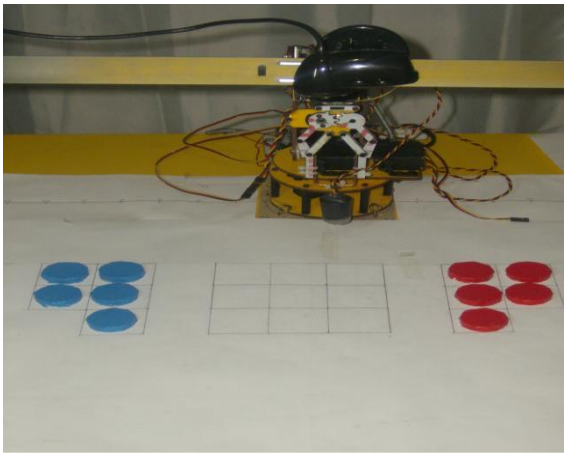


Figure 3: Main View of the System

3. Software

The tool is implemented with C# programming language in Visual Studio .Net platform. The Developed software tool supports two main operation modes, simulation and real time control. Program accepts user commands in every mode. In the real time application mode, System accepts user commands and sends output responses via control card to servo motors. The proposed software can control robot arm and camera with minimum parameter changing in any environment in spite of illumination or other environmental problems. The second mode is simulation mode in which 2D model of the arm and game board are simulated. Even if any user does not have a real physical

robot, complete system can be simulated via this mode .Moreover , in this mode motional characteristics of 2,3,4 or 5 DOF manipulators can be tested apart from the game simulation . The main screen of the software is shown in figure 4. The main section, after user interface is plug-ins which makes application flexible. Flexible structure enables student or researcher to develop their own screens or implement their own algorithms for further applications. The flowchart of the proposed software is shown in figure 5. Initially, proposed plug-in system searches all possible installed plug-in and runs any detected plug-in simultaneously. Each element of the application is implemented inside plug-in. Arm simulator, forward kinematics (FK), inverse kinematics (IK), serial connection Eye matrix library implemented as plug-in and can be changed without compiling main software. It only requires to recompile desired plug-in. Each plug-in is allowed to access other plug-in data. Data results of IK module can be accessed by FK module and vice versa. Data results of both modules can also be accessed from Arm Simulator plug-in. Also each plug-in connects robotic hardware using Serial Connection plug-in. Serial Connection could easily be changed with Wireless or other type of connections. This flexibility makes this application extremely powerful tool for robotics students. Advanced level students can implement their own features without writing whole system again. The software architecture consists of several subsections and several engines. There are four main engine is improved within this software tool these are Kinematic, Simulator, Eye and AI engines.

Kinematics Engine: It is responsible with the solutions of forward and inverse kinematics equations and the calculations related to joints and Cartesian coordinates.

Simulator Engine: It is responsible with the interpretation of the robot model, construction of internal data structures to represent the robot and manipulating the joint angles, as specified by simulation commands and the robot's kinematics

AI Engine: This engine provides a min-max tree algorithm for the primary control of the robot to determine the next best move.

Eye Engine: This engine is directly related to control webcam and to provide image processing method for the grabbed frames through the camera.

4. Kinematics

Kinematics engine is responsible with solution of motional characteristics of Lynx-6 Robot Arm. Kinematics are mainly divided into two groups: *forward kinematics* and *inverse kinematics*. In forward kinematics, the length of each link and the angle of each joint are given and the position of any point in the work space of the robot is calculated. In inverse kinematics, the length of each link and the position of the point in work space are given and the angle of each joint is calculated.

A: Forward Kinematics

To calculate forward kinematics equations for 5 dof Lynx the transformation matrices corresponding to robot arm joints were manipulated and the final forward kinematics solution was shown below **[7]**.

$$T_f = A_1 * A_2 * A_3 * A_4 * A_5 , \quad \text{for 5 Dof} \qquad (1)$$

B: Inverse Kinematics

Inverse Kinematics analysis determines the joint angles for desired position and orientation in Cartesian space. The robot arm manipulator which is used in this study has five degrees of freedom (DOF). To determine the joint angles, Final matrix equation is multiplied by A_n^{-1} (n=1,2,3,4,5,6) on both sides sequentially and the generated linear equations were solved **[7]**. The inverse kinematic solutions for five DOF Lynx arm are shown below,

$$Tf = \begin{bmatrix} nx & ox & ax & px \\ ny & oy & ay & py \\ nz & oz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

θ_{234} (WARTG=Wrist Angle Relative to G round) (2)

$\theta_1 = \arctan(py/px)$ or $\theta_1 = \theta_1 + 180°$ (3)

$\theta_2 = \arctan(S_2 / C_2)$ (4)

$\theta_3 = \arctan(S_3 / C_3)$ (5)

$\theta_4 = \theta_{234} - \theta_3 - \theta_4$ (6)

$\theta_5 = \arctan(S_5 / C_5)$ (7)

Where $S_n = \sin(\theta_n)$ and $C_n = \cos(\theta_n)$.

For more detailed solution of kinematics equations, please look at the given reference.

5. Min Max Algorithm

The Min-Max algorithm is applied in zero sum games, such as tic-tac-toe, checkers, chess, go, and so on. All these games have at least one thing in common, they describes a situation in which a participant's gain or loss is exactly balanced by the losses or gains of the other participant(s). Also they can be described by a set of rules and premisses. With them, it is possible to know from a given point in the game, what are the next available moves. So they also share other characteristic, they are 'full information games'. Each player knows everything about the possible moves of the adversary. Before explaining the algorithm, a brief introduction to search trees is required. Search trees are a way to represent searches. In Figure 6 a representation of a search tree is shown. The squares are known as nodes There are two players involved, MAX and MIN. A search tree is generated, depth-first, starting with the current game position upto the end game position. Then, the final game position is evaluated from MAX's point of view, as shown in Figure 6. Afterwards, the inner node values of the tree are filled bottom-up with the evaluated values. The nodes that belong to the MAX player receive the maximun value of it's children. The nodes for the MIN player will select the minimun value of it's children The MAX player will try to select the move with highest value in the end. But the MIN player also has something to say about it and he will try to select the moves that are better to him, thus minimizing MAX's outcome [8,9].

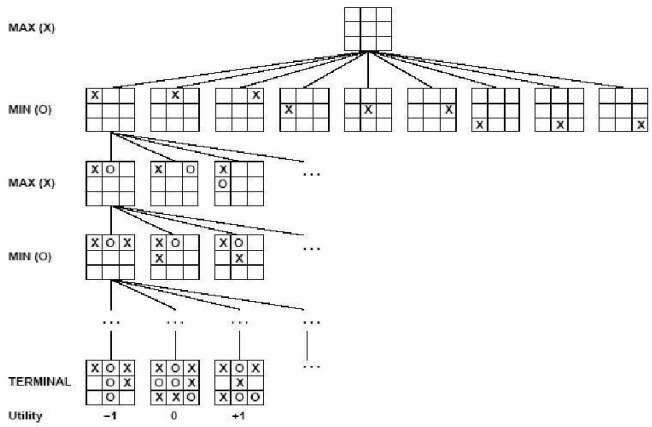


Figure 6: Min-Max Search Tree

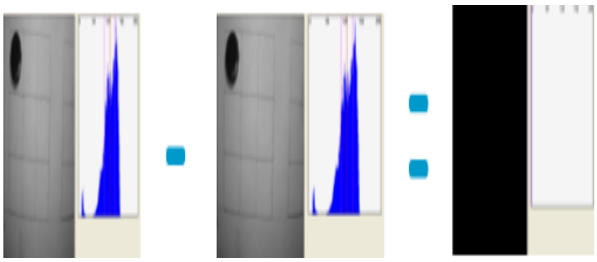
In this study, Min Max algorithm is used for decision of the developed system. Basically the steps of the algorithm are shown below from a to d.

a) Expand game tree as far as possible (look ahead)
b)Evaluate all states of the search tree
c)Use those states to determine the best move
d)Choose the move at the root that will lead to the best move.
It is possible to expand search tree to all possible moves in modern hardware. This enables AI to search all possible moves and enables to decide best available position at runtime. This ability makes AI unbeatable and makes Tic-Tac-Toe problem solved in point of AI methodology. This algorithms is implemented for the proposed system, It is the fact that, this methodology provides optimum solution for this game.

6. Developed Algorithm for Vision System

Vision operation is provided by a low cost web camera during this study. The camera takes record of the game board continuously and a simple but efficient image processing methods are used for detection. There are many techniques have been improved recently for detecting game pieces on the game boards. Most of them are very complex to design and hard to implement. Hence a simple algorithm is improved for the vision system. Mainly developed algorithm relies on image subtraction, one of most well

known arithmetic operations used in image processing operations [10] example is also shown in Figure 7.



$$Q(i,j) = P_1(i,j) - P_2(i,j)$$

Figure 7: Image Subtraction

Mainly all the possible movement of the game is recorded by robot system before the game started , Whenever any movement is occurred by the human opponent it is recorded by the camera at jpeg format and active image is subtracted from all images to detect the exact movement. The steps of the algorithm and explanations are shown below from e to l.

e) The images of the 18 positions on the game board are stored on the image database at JPEG format.
f) Active state of the game board is taken by the camera at JPEG format.
g) Active image is subtracted by all images stored in the database respectively using the technique shown figure 7.
h) A threshold value is used
i) If the difference of the any pixel value during subtraction operation bigger than determined threshold value related variable will increase.
j) The steps g, h, i are applied for all images stored on the database.
k) At the end the smallest variable is selected with selection sort algorithm and the image with related this variable shows the active movement.
l) Detection operation is succeeded and active movement is used by the decision center.

This calibration step mentioned at e is made only one time at deployment phase and robot can play many times against rival until the position of the whole system is changed. In this study developed algorithm is used on both gray and colored images. After testing of both methodologies accuracy, it can be stated that colored images gives more reliable and robust results.

7. Test Results and Discussion

The system was tested with two different way first of all AI (Artificial Intelligence) engine is tested by with five different experimental, they are all different age groups and have different professional skills. The test result are shown table 1. The test results show expected values. AI mechanism is designed as unbeatable so, the best success against the system can be a draw. After that some test were done for vision system. Firstly optimum threshold value is searched and exact threshold values were reached for different light levels. The graphic of the test results are shown figure 8. The threshold values were tested under differently enlightened environments and exact threshold values were determined. It is concluded that 200 is a good value for threshold in all conditions so that 200 is used as threshold value of the system.

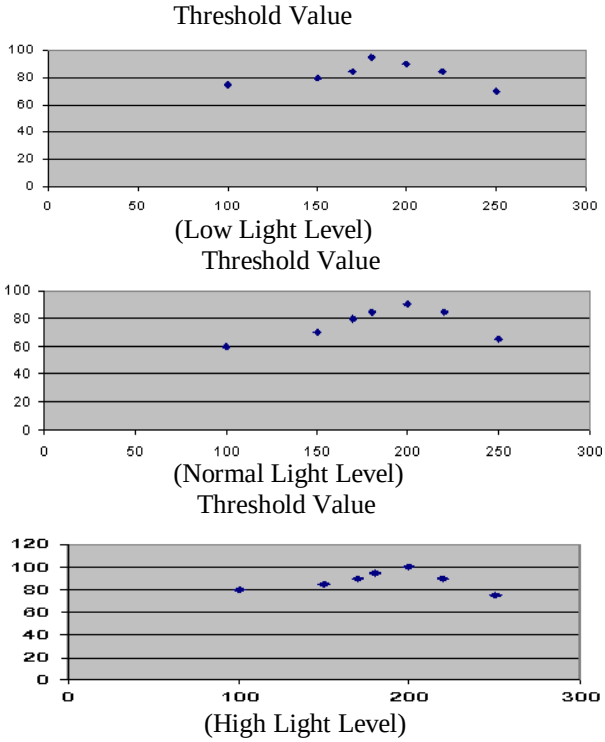


Figure 8: Threshold values for different light levels

Game number (15) , X (AI) , O(Human)			
X won	O won	Draw	Percentage of wining
9	0	6	%60
13	0	2	%86
7	0	8	%47
12	0	3	%80
10	0	5	%66
8	0	7	%53

Table 1: Test Result for AI Mechanism

When the calibration step is determined by any environment, changing on the light level does not affect vision system with this threshold value. However when an error is recognized by the system, automatic threshold arranging system are triggered and threshold value are changed automatically. One problem is related with the digital servo motors of the system. Three HS-5745MG [11] digital servo motors are used by the system. They are very powerful so that there is little deviation is reported which does not affect the operation of whole system. However they could be rested every 15 minutes to prevent deviation. It is obtained that after 15 minutes running the deviation rate increases steadily.

Conclusions

In this study it is proposed to create an artificial software tool for low cost manipulator. An artificial low cost robotic system is also designed to play Tic-Tac-Toe against humans within this tool. The tool has a user-friendly interface and developed with C# in .Net 2005 platform. The hardware part of the system consists of a low cost Lynx-6 manipulator and a basic web camera for vision operations. Min Max algorithm is used for AI engine of the system and a basic and fast vision algorithm is also developed to detect positions of the pieces on the game board during this study. It also works on motional characteristics of flexible. Whole software relies on plug-in architecture which makes application flexible. The proposed system enables researcher to develop their applications with little effort. The video of the system also can be easily reached in (http://uk.youtube.com/watch?v=HFuBMOLuAn0).

Manipulator solves both inverse and forward kinematics. The developed software tool is designed as an open source tool; Researchers and students can add new module to the system and create their applications. There are many robot based systems are developed for Tic-Tac-Toe games. Most of them use sensors for detection operations instead of camera and do not have flexible structure. It is stated that, this flexible plugging based software tool will be a new approach for researcher and student. This free Software package will be on the official web page of the project in a short period of time shown below. (http://yaskil.blogspot.com).Manipulator solves both inverse and forward kinematics. The developed software tool is designed to be Software is also open source tool; researchers and students can add new module to the system and create their applications. There are many robot based systems are developed for Tic-Tac-Toe games. Most of them use sensors for detection operations instead of camera and do not have flexible structure. It is stated that, this flexible plugging based software tool will be a new approach for researcher and student. This free Software package will be on the official web page of the project in a short period of time (http://yaskil.blogspot.com).

Consequently this study shows that it is possible to create fully automated intelligent robotics systems by using low-cost robot, a computer and a simple webcam.

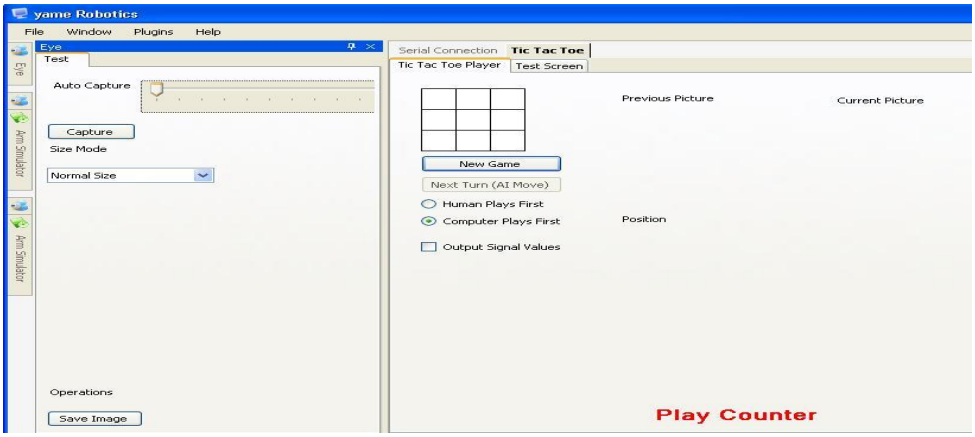


Figure 4: Main Screen of the Software

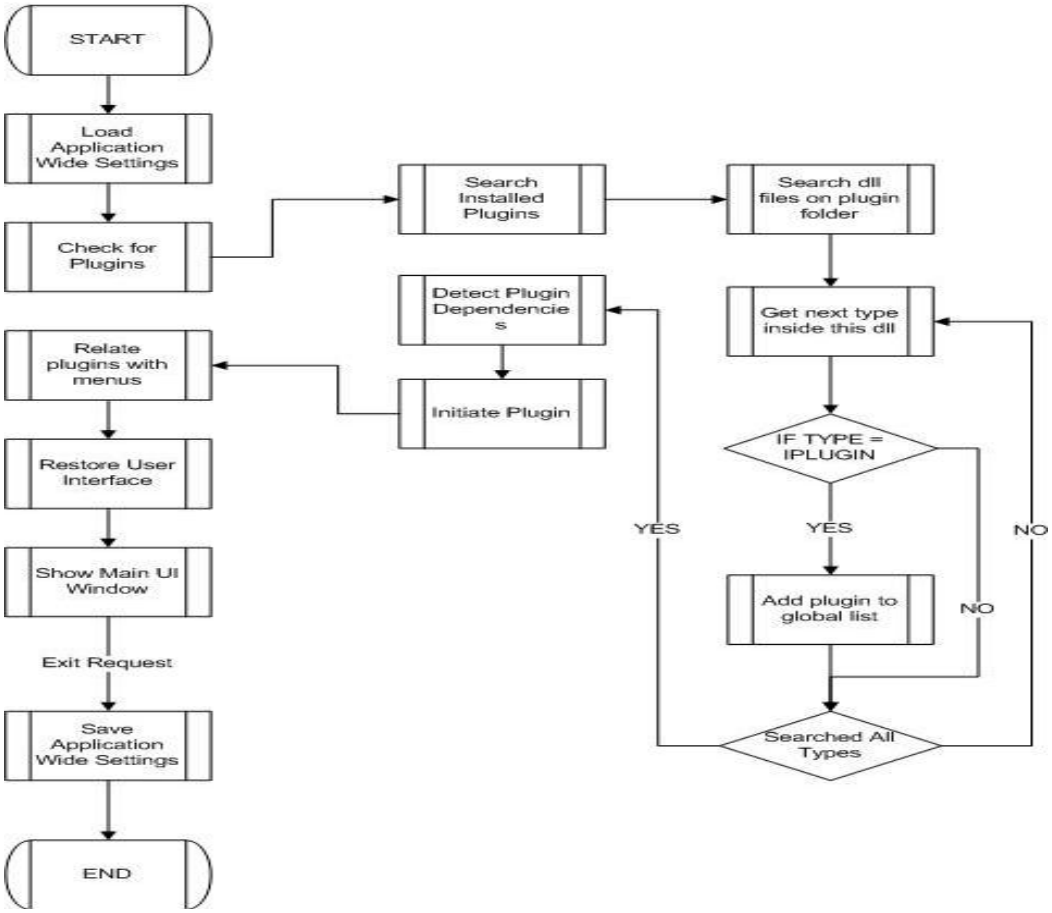


Figure 5: The flowchart of Software

References

[1] Ruben Vuittonet , Jeff Gray , Tic-Tac-LEGO: , An Investigation into Coordinated Robotic Control, Proceedings of the 44th annual southeast regional conference, pp 796-781, 2006.

[7] Baki Koyuncu , Mehmet Güzel, Software Development for the Kinematic Analysis of a Lynx 6 Robot Arm ,International Journal of Applied Science, Engineering and Technology , Volume 4, Number 4 pp. 228-233 ,2007

[8] Paulo Pinto, Introducing the Min-Max Algorithm , 28 July 2002.

[9] Russell, S. and Norving, P. Prentice Hall , Artificial Intelligence A Modern Approach , pp 67-76 , 2001

[10] Rafael C. Gonzales, and Richard E. Woods, Prentice Hall , Digital Image Processing , pp 108-112 , 2002.

[11] SevoCity ,2008 "http://www.servocity.com/html/hs-5745mg_digital_1_4_scale.html"

Code migration from a realistic simulator to a real wheeled mobile robot

José Gonçalves, José Lima, Paulo Malheiros and Paulo Costa

VIDEO

***Abstract*— This paper describes the code migration from a realistic simulator to a real wheeled mobile robot. The robot software consists in the localization and navigation of an omnidirectional robot in a structured environment. The localization estimate is achieved by fusing odometry and infra-red distance sensors data, applying an extended Kalman filter.**

I. Introduction

This paper describes the code migration from a realistic simulator to a real wheeled mobile robot. Code migration from realistic simulators to real world systems is the key for reducing the development time of robot control, localization and navigation software [1]. For this purpose it was developed a realistic simulator (available for download at [2]). Due to the inherent complexity of building realistic models for the robot, its sensors and actuators and their interaction with the world, it is not an easy task to develop such simulators.

The developed robot software consists in the localization and navigation of an omnidirectional robot in a structured environment. The robot is equipped with brushless motors and infra-red distance sensors. The localization estimate is done by fusing odometry and the distance sensors data, applying an extended Kalman filter. In the first place it is presented how to develop robot code and how to migrate it to the real robot, then it are presented the distance sensor modeling, the absolute position estimation and the localization algorithm based on a kalman filter. Finally some conclusions are presented.

II. Code migration to a real robot

A. Code generated with the simulator

Initially, the localization and navigation software is generated with the simulator (Figure 1). The simulator provides to a Remote application the distance sensors data with noise as modeled in Section III [3], the encoders data and the real robot position. The Remote application executes the localization and navigation algorithms and returns the speed references for each wheel to the simulator. The applied communication protocol to exchange data between the Remote application and the simulator is UDP, as shown in Figure 2. As an example, a robot trajectory produced in the simulator is shown in Figure 3. The pose estimate error and its variance are also shown in Figure 4.

José Gonçalves and José Lima are with the Polytechnic Institute of Bragança, Department of Electrical Engineering, Bragança, Portugal {goncalves,jllima}@ipb.pt

Paulo Malheiros and Paulo Costa are with the Faculty of Engineering of University of Porto, DEEC, Porto, Portugal {paulo.malheiros,paco}@fe.up.pt

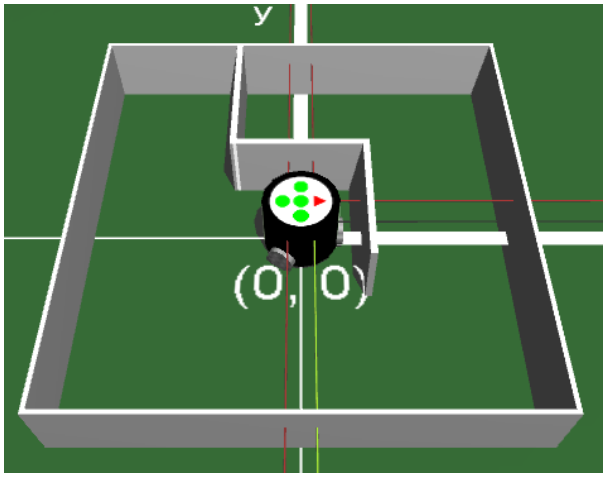


Fig. 1. Robot simulator snapshot.

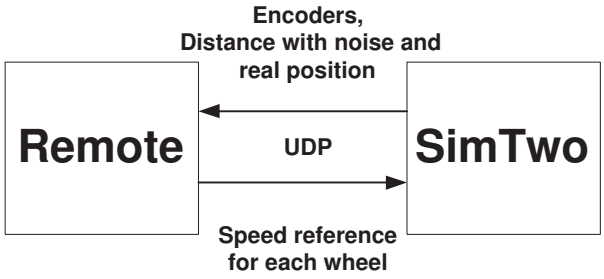


Fig. 2. Simulator communicating with the Remote application

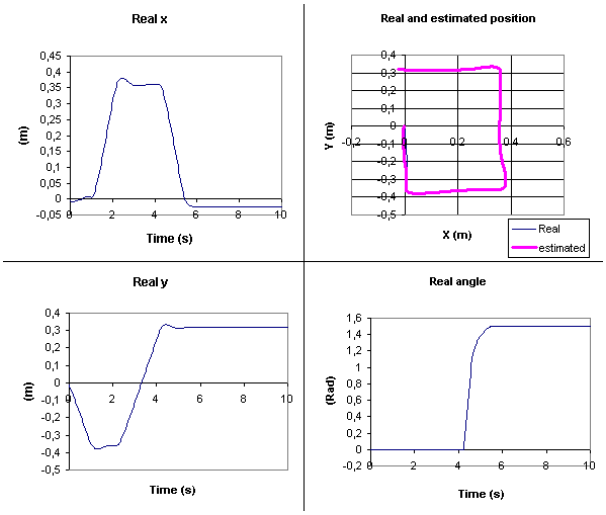


Fig. 3. Simulated robot trajectory

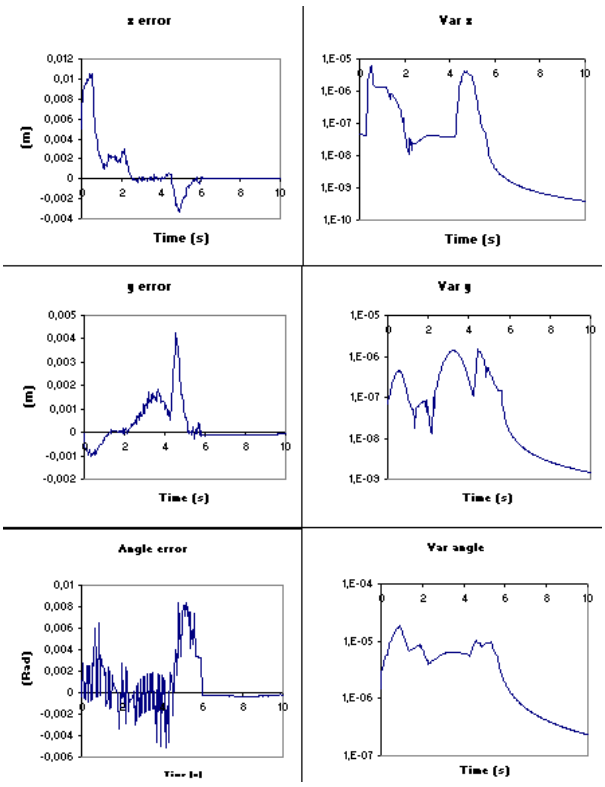


Fig. 4. Simulated robot estimate error and variance

B. Developed code applied to the real robot

The architecture used to migrate the robot code to the real robot is presented in Figure 5. The Remote Application is shared with the simulation so that the generated code can be applied to the real robot without any changes. The robot real position is provided to the Gate application at a 25 Hz rate, by a global vision system described in [4] (with the difference that was used only the absolute measurements without applying the Kalman filter). The control loop is initiated by the robot when it sends to the Gate application, via RS232, the encoders and the sensor data at a 25 Hz rate. Then, the Gate application provides to the Remote application the distance sensors data, the encoders data and the real robot position via UDP. The Remote application executes the localization and navigation algorithms and returns to the Gate application the speed references of each wheel. Finally, the speed references are sent to the real robot via RS232 protocol. The trajectory executed with the real robot, shown in Figure 6, is similar to the simulated making the simulation very useful, because it allows to reduce considerably the development time of the robot software. The real robot trajectory estimate error and variance are shown in Figure 7.

III. Distance Sensors Modeling

The Sharp family of infra-red range finders is very popular for robotics distance measurement applications. Some drawback of these sensors are their non-linear response and the mandatory minimum distance measurement requisites. The presented study is about the Sharp infra-red distance sensor GP2D120. In order to model the distance sensor it was necessary to collect a considerable amount of data, for this task it was used the industrial robot ABB IRB 1400 to

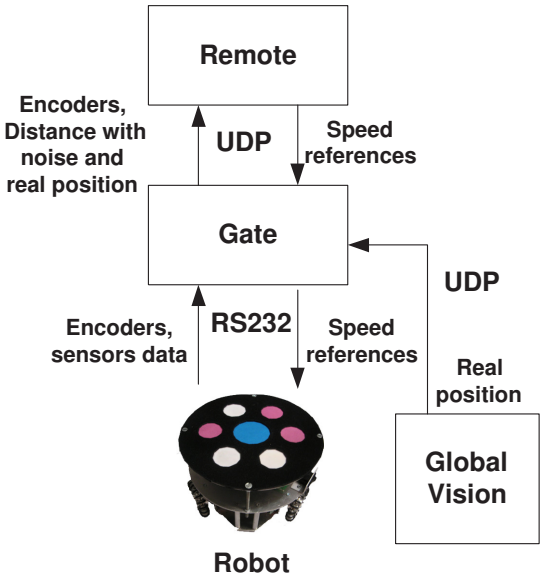


Fig. 5. Real world system architecture

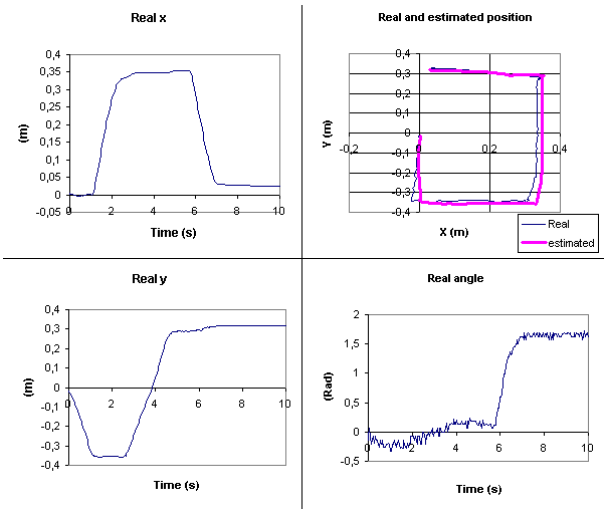


Fig. 6. Real robot trajectory

place an obstacle for different distances as shown in Figure 8. Industrial robots allow executing repetitive operations normally performed by human operators, without getting bored and without loosing precision [5]. The introduction of an industrial robot to place the obstacle in different known positions allows to increase the speed, repeatability and reduces errors in the process of distance sensor data collecting.

The sensor data is acquired using the internal analog to digital converter (ADC) of the Atmel AVR ATMega8 with 8 bit precision. At each mobile robot sample time the ADC registers 10 samples for each sensor, which are added and sent to a personal computer. In order to evaluate the sensor noise it are registered 256 mobile robot sample times data for each distance. As the used analog to digital converter was the provided internally with the micro-controller ATMega8, and since there is available an internal reference voltage of 2.56 V (V_{ref}), it is possible to have a precision increase using this reference, when compared to the alternative of using an external reference voltage of 5 V. To use this approach, a voltage divisor must

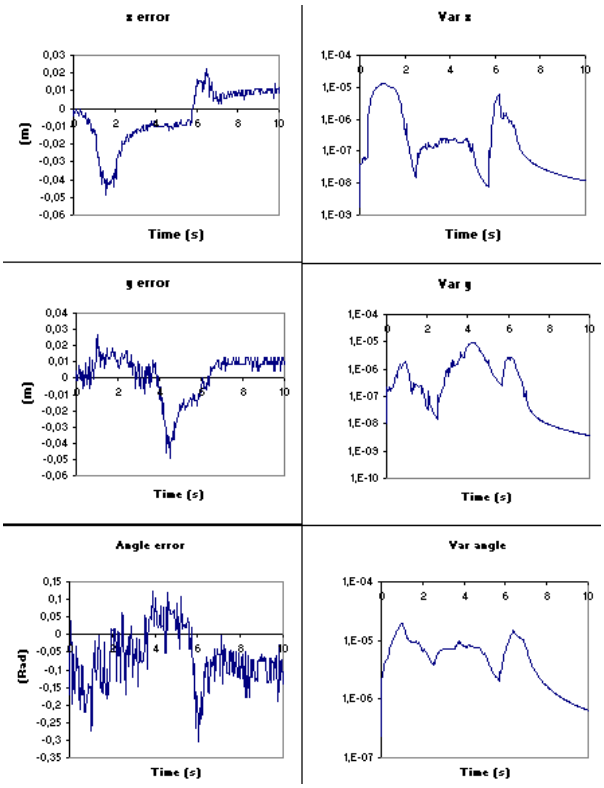


Fig. 7. Real robot estimate error and variance

be applied in order to lower the sensor voltages to values below the converter reference (2.56 V), since its maximum is nearly 3.2 V. This is important if the user wants to use sensor values from 7 to 10 cm. If the user makes the choice of using a minimal distance of 10 cm then it is not necessary to apply a voltage divisor because the value for 10 cm corresponds to nearly 2.33 V, which is below the internal converter reference, and the voltage decreases with the distance. For this application it was considered that it was important to use the sensor range from 7 to 100 cm, thus a voltage divisor was applied. The obtained voltage characteristic of the infra-red distance sensor is presented in Figure 9. It was calculated resorting to equation 1, where v is the voltage, si is the *ith* sample, n is the number of acquired samples and V_{ref} is the micro-controller internal reference voltage.

$$v = V_{ref}(\frac{\sum si}{n.255}) \quad (1)$$

The relation between the inverse voltage and the distance can be approximated to a line as shown in Figure 10, where the real and the approximated curve are presented. The used values to achieve the presented curve were from 7 to 100 cm, taking in account the chosen sensor minimal distance.

In order to obtain the distance in the real robot, having in mind the shown approximation, equation 2 can be applied, where d is the distance expressed in m and v is the sensor voltage, k_1 equals 0.1881 and k_2 equals 4.6779.

$$d = \frac{\frac{1}{v} - k_1}{k_2} \quad (2)$$

In order to obtain the distance variance it was made

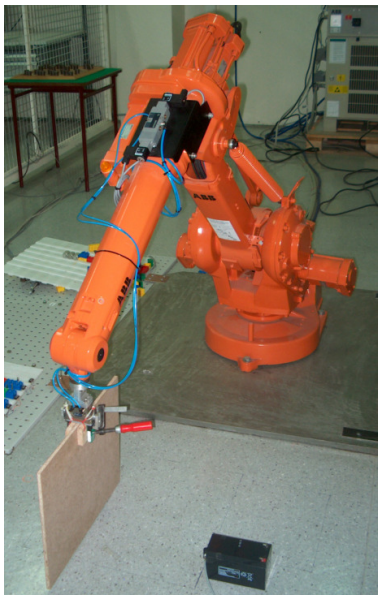


Fig. 8. IRB 1400 placing the obstacle.

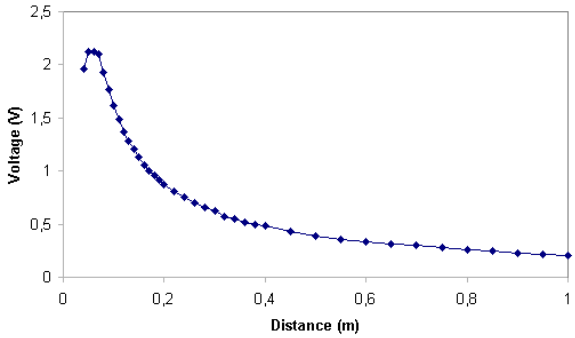


Fig. 9. IR distance sensor characteristic.

the approximation shown in equation 3, with a different derivative for each distance, where m is the voltage derivative presented in equation 4 and in Figure 12. The voltage equation was obtained from the approximation presented in equation 2.

$$v = m.d + b \quad (3)$$

$$m = \frac{-k_2}{(d.k_2 + k_1)^2} \quad (4)$$

This approximation was made in order to obtain the distance variance related with the known voltage variance.The simulated infra-red distance sensors provide the distance with the noise. Its variance is shown in Figure 13, which was obtained resorting to equation 5 applying the approximation of the voltage variance presented in Figure 11.

$$var(d) = \frac{Var(v)}{m^2} \quad (5)$$

An example of two simulated sensors for two different distances is shown in Figure 14. In the example both measures are presented with and without noise.

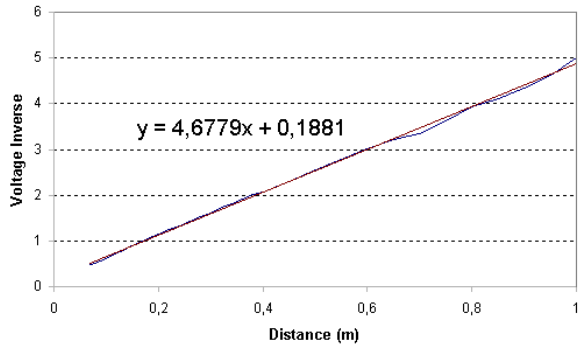


Fig. 10. Voltage Inverse VS Distance.

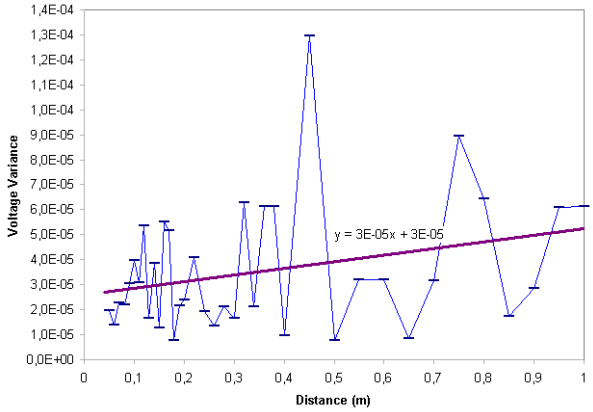


Fig. 11. Voltage variance.

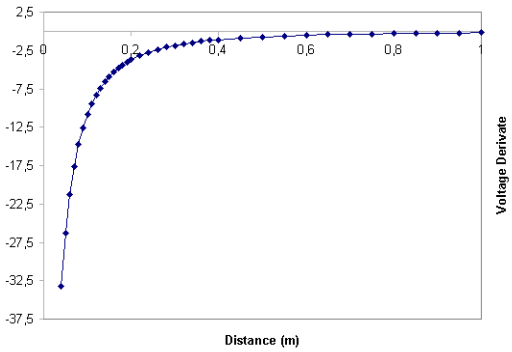


Fig. 12. Voltage derivative.

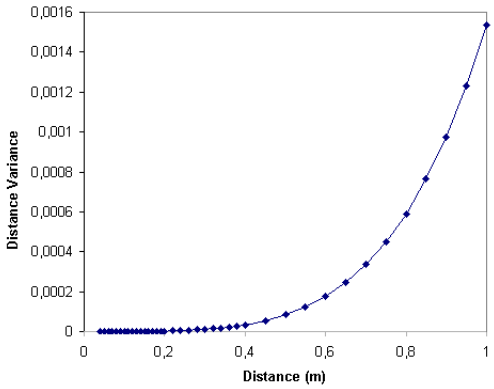


Fig. 13. Simulated sensors variance.

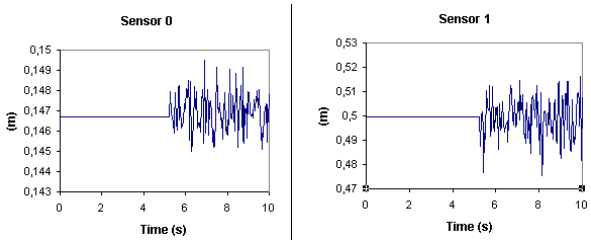


Fig. 14. Simulated sensors with and without noise

IV. Absolute position estimation

In order to extract the absolute position estimation it is necessary to estimate the x, y and θ. The robot disposes of three pairs of Sharp infra-red distance sensors, each par is capable of estimative each absolute position parameters. For this purpose the robot is given, initially, a correct position estimative and a map of its environment.

A. Simulator world construction

The robot, the on-board sensors and the environment are described using the standard eXtensible Markup Language (XML). The robot and its environment are shown in Figure 1. The developed environment has only 90 degrees angles being a possible representation of the real world where our buildings are mostly 90 degrees and usually very artificial [6]. The environment is built using blocks and the robot is a three wheel omnidirectional robot equipped with three pairs of infra-red distance sensors. The intersection of a sensor beam with a block returns a distance with noise.

B. Robot map

The robot knows its environment by knowing a Map, but instead of blocks the Map is composed by lines with the following parameters if it is an horizontal line:

- y position
- minimum x
- maximum x
- VFB2T - visible from bottom to top (boolean)

and by the following attributes if a vertical line:

- x position
- minimum y
- maximum y
- VFL2R - visible from left to right (boolean)

The environment is sensed with distance sensors characterized by the following parameters:

- Angle
- x position
- y position

The sensor parameters are in a local robot referential.

C. Information extracted from the Map

Since the robot is provided with a new position estimate at each sampling time (each 40 ms), it is possible to predict, for each sensor, a distance and the line that the sensors beam is expected to hit. In order to obtain, at each sample, the sensor parameters in the world referential it is necessary, in the first place, to offset the sensor position with the robot position estimate and then rotate the obtained position with the estimated robot angle. Finally the new angle sensor is the angle sensor summed with

the estimated robot angle. To obtain the distance to a line and the expected line that the beam sensor is hitting it is necessary to use equation 6.

$$(x_l, y_l) = (x_s, y_s) + d(u, v) \tag{6}$$

where:

- x_l - x position in the line where the beam is hitting
- y_l - y position in the line where the beam is hitting
- x_s - x position of the sensor in the world referential
- y_s - y position of the sensor in the world referential
- d - measured distance
- ϕ - Angle sensor parameter summed with the estimated robot angle
- u - $cos(\phi)$
- v - $sin(\phi)$

The shown parameters are illustrated in Figure 15.

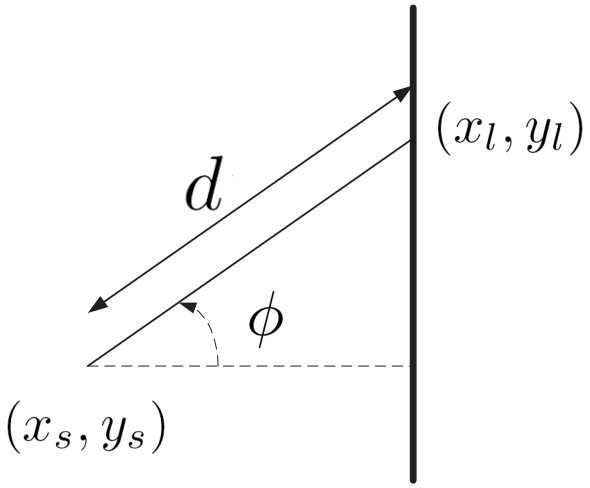


Fig. 15. Sensor hitting a wall

As an example, for a vertical line, as the distance in the x axis is a known constant it is possible to assume that:

$$d = \frac{x_l - x_s}{u} \tag{7}$$

Knowing d, from equation 7, it is possible to obtain y_l by the following equation:

$$y_l = y_s + dv \tag{8}$$

For an horizontal line the calculus process is similar to the example described for an vertical line. Repeating this process for all the robot Map lines (both horizontal and vertical), it is possible to know the distance and the expected wall that the sensor is expected to hit, by choosing the lowest positive distance.

D. Robot position estimation

The used approach to estimate the robot absolute position is valid only if the pair of sensors is expected to hit the same wall. If one of the sensors is expected to hit a different wall then it is considered that the estimative has an variance higher enough so that the filter neglects its contribution. This assumption is valid both for the angle and for x and y estimation. As an example it will be shown the calculation of the robot position estimation of a pair of sensors hitting a vertical wall as shown in Figure 16.

The pair of sensors has the following parameters in the local robot referential:

- x position - 0 for both sensors
- y position - L_s for Sensor 1 and $-L_s$ for Sensor 2
- $angle$ - 0 for both sensors

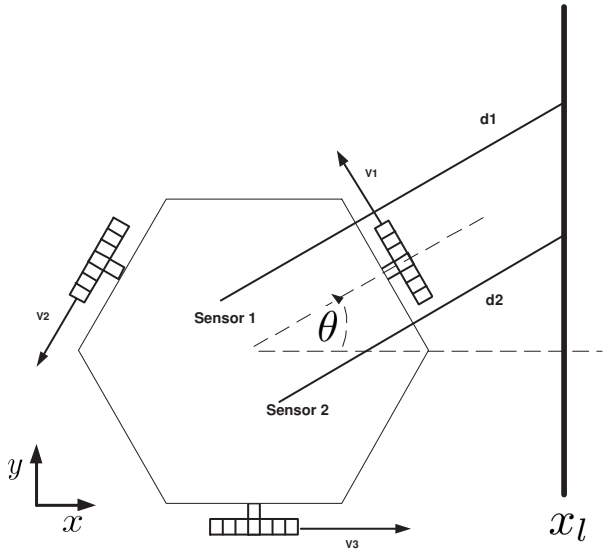


Fig. 16. Pair of sensors hitting a vertical wall

1) θ calculation: As both sensors are seeing the same wall it is possible to estimate the robot angle resorting to equation 9.

$$\theta = \arctan(\frac{d1 - d2}{2L_s}) \tag{9}$$

The horizontal walls also provides angle information. All the available information provided by the three pairs of sensors will be fused in order to obtain an optimal estimation.

If the expected distance of one of the sensors differs of more than 1.2 cm from the real measured distance, then it will be considered that there is no trust in the angle estimation, and its variance will be very high.

2) x and y calculation: As the two sensors are seeing the same wall it is possible to estimate the robot x position resorting to equation 10.

$$x = x_l - \frac{d1 + d2}{2}\cos(\theta) \tag{10}$$

A vertical wall do not provides information for y position so it is considered for this parameter that the variance is high enough that its contribution is negligible when fusing information from the different pairs of sensors.

If the expected distance of one of the sensors differs of more than 5 cm from the real measured distance, then it will be considered that there is no trust in the x or y estimation, and its variance will be very high.

3) Fusion: As there are available three pairs of sensors, there are three estimates for each parameter that must be fused in order to obtain an optimal estimate. To fuse the several estimates it is necessary to apply equation 11.

$$p = \frac{\frac{ps1}{var(ps1)} + \frac{ps2}{var(ps2)} + \frac{ps3}{var(ps3)}}{var(ps1)^{-1} + var(ps2)^{-1} + var(ps3)^{-1}} \quad (11)$$

where $var(psi)$ is the variance for a parameter given by the par of sensors i and p is the resulting optimal parameter estimation. All the robot position parameters depend on the distances $d1$ and $d2$. It is possible to approximate the parameters estimation variance as shown in equation 12 [7].

$$var(p) = \frac{\partial p}{\partial d_1}^2 var(d1) + \frac{\partial p}{\partial d_2}^2 var(d2) \quad (12)$$

V. Odometry and Distance Sensors Data Fusion

Odometry and infra-red distance sensors data fusion was achieved applying an extended Kalman filter. This method was chosen because the robot motion equations are nonlinear and also because the measurements error probability distributions can be approximated to Gaussian distributions [8][9].

A. Extended Kalman filter algorithm

With the kinematic dynamic model given by equations system (13) and Figure 16.

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} = \begin{pmatrix} -sin(\theta) & cos(\theta) & L \\ -sin(\frac{\pi}{3} - \theta) & -cos(\frac{\pi}{3} - \theta) & L \\ sin(\frac{\pi}{3} + \theta) & -cos(\frac{\pi}{3} + \theta) & L \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ w \end{pmatrix} \quad (13)$$

and considering that control signals change only at sampling instants, the state equation is:

$$\frac{dX(t)}{dt} = f(X(t), u(t_k), t), t\epsilon[t_k, t_{k+1}] \quad (14)$$

Where $u(t) = [V_1 V_2 V_3]^T$, that is, the odometry measurements are used as kinematic model inputs. This state should be linearized over $t = t_k$, $X(t) = X(t_k)$ and $u(t) = u(t_k)$, resulting in:

$$A^*k = \begin{pmatrix} 0 & 0 & \frac{-\sin(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\cos(\theta)}{2(1+\cos(\frac{\pi}{3}))} \\ 0 & 0 & \frac{\cos(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\sin(\theta)}{2(1+\cos(\frac{\pi}{3}))} \\ 0 & 0 & 0 \end{pmatrix} \quad (15)$$

with state transition matrix:

$$\phi^*(k) = exp(A^*(k)(t_k - t_{k-1})) \quad (16)$$

Resulting in:

$$\phi^*k = \begin{pmatrix} 1 & 0 & (\frac{-\sin(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\cos(\theta)}{2(1+\cos(\frac{\pi}{3}))})T \\ 0 & 1 & (\frac{\cos(\theta)}{2\sin(\frac{\pi}{3})} + \frac{\sin(\theta)}{2(1+\cos(\frac{\pi}{3}))})T \\ 0 & 0 & 1 \end{pmatrix} \quad (17)$$

Where T is the sampling time $(t_k - t_{k-1})$.

Thus the observations are obtained directly, H^* is the identity matrix.

The extended Kalman filter algorithm steps are as follows [10] [11]:

1) State estimation at time $t = t_k, X(k^-)$, knowing the previous estimate at $t = t_{k-1}$, $X(k-1)$ and control $u(t_k)$, calculated by numerical integration.
2) Propagation of the state covariance

$$P(k^-) = \phi^*(k)P(k-1)\phi^*(k)^T + Q(k) \quad (18)$$

Where $Q(k)$ is the noise covariance (14) and also relates to the model accuracy.
As there is a measure, the follow also apply:
3) Kalman gain calculation

$$K(k) = P(k^-)H^*(k)^T(H^*(k)P(k^-)H^*(k)^T + R(k))^{-1} \quad (19)$$

Where $R(k)$ is the covariance matrix of the measurements.
4) State covariation update

$$P(k) = (I - K(k)H^*(k))P(k^-) \quad (20)$$

5) State update

$$X(k) = X(k^-) + K(k)(z(k) - h(X(k^-, 0))) \quad (21)$$

Where z(k) is the measurement vector and $h(X(k^-, 0))$ is $X(k^-)$.

VI. Conclusions

Odometry and distance sensors data fusion was achieved applying an extended Kalman filter. This method was chosen because the robot motion equations are nonlinear and also because the measurements error probability distributions can be approximated to Gaussian distributions.

Code migration from realistic simulators to real world systems is the key for speeding up the developing time in robot software production. The developed code in the simulator was migrated to a real robot, reducing considerably the development time.

References

[1] O. Michel, $Webots^{TM}$: Professional Mobile Robot Simulation, ISSN 1729-8806, International Journal of Advanced Robotic Systems,volume 1,number 1, 2004.
[2] "SimTwo" http://www.fe.up.pt/∼ $paco/wiki$, 2008.
[3] J. Gonçalves, J. Lima, H. Oliveira and P. Costa, *Sensor and actuator modeling of a realistic wheeled mobile robot simulator,* 13th IEEE International Conference on Emerging Technologies and Factory Automation, Hamburg, 2008.
[4] J. Gonçalves, J. Lima and P. Costa *Real time localization of an omnidirectional robot resorting to odometry and global vision data fusion: An EKF approach*, IEEE International Symposium on Industrial Electronics, Cambridge, 2008.
[5] M. Groover, M. Weiss, R. Nagel, N. Odrey *Industrial Robotics: Technology, Programming, and Applications* McGraw-Hill, 1995.
[6] Fire Fighting robot competition FAQ *http://www.trincoll.edu/events/robot/FAQ,* Why does the arena have only 90 degree angles?, Trinity College Hartford Conneticut, 2008.
[7] M. I. Ribeiro, *Gaussian Probability Density Functions: Properties and Error Characterization,* Technical Report, IST, 2004.
[8] S. Thrun, W. Burgard and D. Fox, *Probabilistic Robotics,* MIT Press, 2005.
[9] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, *Principles of Robot Motion : Theory, Algorithms, and Implementations,* MIT Press, 2005.
[10] G. Welch and G. Bishop, *An introduction to the Kalman filter,* Technical Report, University of North Carolina at Chapel Hill, 2001.
[11] R. Negenborn, *Robot Localization and Kalman Filters - On finding your position in a noisy world,* Master Thesis, Utrecht University, 2003.

The Dynamic Modeling of a Bird Robot

Micael S. Couceiro, Carlos M. Figueiredo, N. M. Fonseca Ferreira, J. A. Tenreiro Machado

***Abstract*— This paper presents the development of computational simulation and a robotic prototype based on the dynamic modeling of a robotic bird. The study analyze the bird flight with different strategies and algorithms of control. The results are positive for the construction of flying robots and a new generation of airplanes developed to the similarity of flying animals.**

A bird robot prototype was implemented based on materials used to construct model planes. It consists on a body, wings and tail with servo actuators independently controlled though a microcontroller; a radio transmission system and batteries were also used in order to avoid wired connections between the computer and the robot.

I. INTRODUCTION

If ground robots are now endowed with advanced autonomous abilities, allowing them to avoid obstacles, to build internal maps of their environment, to choose the best action to undertake at any time, their flying equivalents are far from exhibiting such abilities, and a direct application of techniques developed for ground robots is difficult. Therefore, design and control of bird-like robots have been attracting much attention of many researchers, and various models of bird-like robots have been proposed.

Everything about a bird is made for flight and for this reason, the kinematic and dynamic modeling of bird-like robots is more complex than that of serial robots. Therefore, in order to construct a robotic bird, we first need to implement a computer simulation considering every single physical and dynamical aspect [1]. When developing control algorithms for flying robots, a simulation tool proves to be important to reduce the development time, avoid damages and find errors in the control system. A software simulation can be easily manipulated and monitored offering data that would be hard to measure on a real robot.

This paper analyses the major developments in this area and the directions towards future work.

The paper is organized as follows. Section two, presents the state of the art. Section three develops the kinematics of the robotic bird. In the section four it is shown the simulation platform. Section five gives a overview of the electronic system. Section six presents some highlights about the implementation of the first robotic bird. Finally, section seven outlines the main conclusion and future works.

II. STATE OF THE ART

The flight of insects has been interesting subject during the last half century, but the attempts of recreating it are well more recent [2]. Regarding to the flight of birds, airplanes designers are interested in the morphing capacities of wings. This area received a great impulse in 1996, when the Defense Advanced Research Projects Agency of E.U. (*DARPA*) launched a program MAV of three years with the objective of creating an insect with less than 15 centimeters length to make military recognition.

Some works of fixed wings had been successfully demonstrated, especially the black widower, the *AeroVironment Inc.* [3]. Several MAVs had been equally demonstrated, but no group has been capable to obtain a flight with flapping wings that could effectively take off and fly. Recently, several groups have equally studied the concept of morphing wings [4]. Based in these ideas some examples of modern *ornithopters* are in development. The research group of the University of Toronto developed an airplane with flapping wings with an internal combustion engine capable to support a pilot (Fig. 1).

Fig. 1. Plane with flapping wings.

Fig. 2a shows the *Cybird* developed in 2006, that uses a transmitter to control the flight. The direction of the flight can be modified moving the tail to the left or the right and the height is controlled with the flapping speed of the wings. With a battery and an engine, *Cybird* can fly for 8 to 10 minutes and can then be recharged to be ready to another flight.

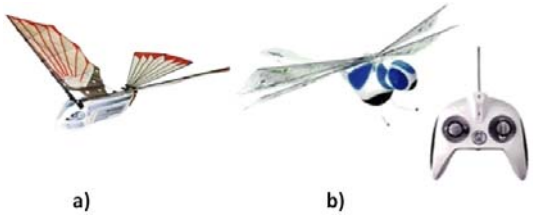


Fig. 2. a) Cybird , b) Dragonfly.

Just like a puzzle, *Cybird* is easy to assemble and disassemble, having the size of a pigeon with spreading wings of almost 76.2 cm. To land, the *Cybird* will have to reduce the flapping speed, gliding until it reachs the ground. Another similar platform called *Dragonfly* was developed in 2007, illustrated in Fig. 2b. Equally radio controlled, it resembles a dragonfly with spreading wings

This work was supported by GECAD

Institute of Engineering of Coimbra, Dept. of Electrotechnical Engineering, Quinta da Nora, 3030-199 Coimbra, Portugal micaelcouceiro@gmail.com; carlos.figueiredo@gmail.com; nunomig@isec.pt

Institute of Engineering of Porto, Dept. of Electrotechnical Engineering, Rua Dr Ant. Bern. de Almeida 4200-072 Porto, Portugal jtm@isep.ipp.pt

Edited by P.J.S. Gonçalves, P.J.D. Torres, O.M.D. Alves - ISBN 978-972-99148-8-6

of 40.6 cm with a light body and strong double wings. Another kind of wing structure consists on a set of small airfoil parts throughout the wing, as a regular fixed wing. However, these small parts are connected with a light and flexible material in order to make the wing fold and twist when it beats. This approach leads equally to a similar wing movement of those of real birds (Fig. 3). This ornithopter is one of the most recent constructions of the *Elecktro Vogel* series developed by [5].

Fig. 3. Horst Rbinger EV7 ornithopter.

III. KINEMATICS

A. The Geometry of the Robotic Bird

In order to visualize the behavior of the bird during the simulation we developed a 3D model in *AutoCAD* inspired in a seagull as can be seen in Fig. 4. Each adjacent part (with different colors) corresponds to individual elements connected through joints. For simplicity, the structure of the wings is defined in the sense of a human arm, using the terms arm and hand accordingly. The corresponding wing joints will be denoted the shoulder and wrist.

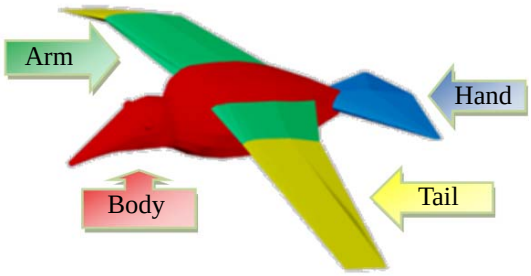


Fig. 4. 3D model of the robotic bird.

B. The Kinematics of the Robotic Bird

With the kinematic model we analyzed the bird flight movement and its behavior in different states such as taking off, flying with twists and turns, and others. Through this study, we obtained valuable specifications which helped choosing the initial mechanical design (Fig. 5).

The multi-link model is shown in Fig. 6. The number of joints is limited, when compared with a real bird, but this mechanical structure gives a good mobility. The joints are distributed as follows: two in the shoulder, one in the wrist and two in the tail. Differently from all the others, the wrist joint is not controlled. It consists in a mechanical spring mechanism that allows a movement of the wing similar to real birds. This structure provides a good mobility having a total of six controlled joints.

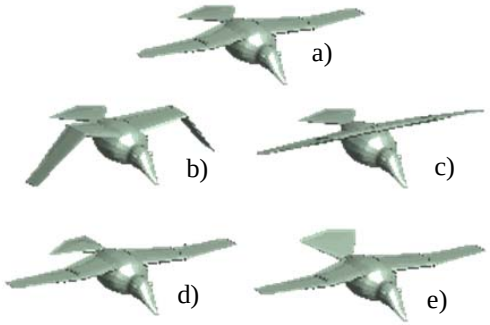


Fig. 5. a) Bird geometry, b) Wing flapping, c) Wing twisting, d) Tail twisting, e) Tail bending.

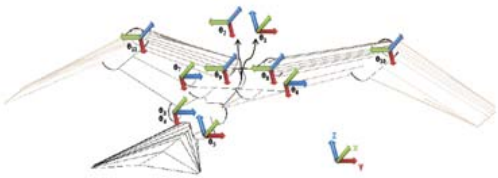


Fig. 6. Kinematic structure of the system.

In order to implement the animation of the bird in MatLab it is adopted the Denavit-Hartenberg (D-H) notation to represent frame (joint) coordinates in the robot kinematic chain. The next equation represents the homogeneous transformation A_i, namely a matrix constituted by a product of four fundamental transformations.

$$A_i = R_{\theta i} T_{di} R_{\alpha i} T_{ai} \quad (1)$$

where $R_{\theta i}$ and T_{di} are, respectively, the matrix rotation and matrix translation in *x*-axis and $R_{\alpha i}$ and T_{ai} are, respectively, the matrix rotation and the matrix translation in *z*- axis. With a series of D-H matrix multiplications, the final result is a transformation matrix from a given frame to the initial frame.

IV. SIMULATION PLATFORM

In the last decades robotics became a common subject in courses of electrical, computer, control and mechanical engineering. Progress in scientific research and developments on industrial applications lead to the appearance of educational programs on robotics, covering a wide range of aspects such as kinematics and dynamics, control, programming, sensors, artificial intelligence, simulation and mechanical design. Nevertheless, courses on robotics require laboratories having sophisticated equipment, which pose problems of funding and maintenance. The development of simulation platforms became an important ally of science, and today it is spread in the most varied sectors. The computer programs emphasize capabilities such as the 3D graphical simulation and the programming language giving some importance to mathematical aspects of modeling and control, [6]. However, undergraduate students with no prior experience may feel difficulties in getting into the robotics experiments before overcoming the symbolic packages procedures and commands. This state of affairs motivated the development of a computer program highlighting the fundamentals of robot mechanics

and control. The project leads to the *SIRB - Simulation and Implementation of a Robotic Bird* - program which was adopted as an educational tool in a flying robotics birds. This section introduces the package and discusses both basics and advanced aspects. The simulation platform was developed in *MatLab* (Matrix Laboratory) being a software for numerical computation and high performance visualizations. It allows to efficiently implement and solve mathematical problems faster than other languages such as *C*, *BASIC*, *PASCAL* or *FORTRAN*. The *SIRB* educational package (Fig. 7) was designed to take full advantage of the Windows environment. All the commands and the required parameters are entered through pull-down menus and dialog boxes. The software is intended to be self-explanatory to the extent possible to encourage students exploring the program. For the same purpose, help menus are available throughout the different windows. Several dialog boxes include figures to clarify context-dependent definitions.

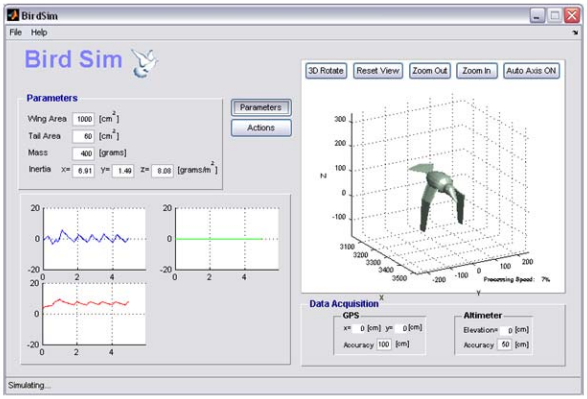


Fig. 7. *SIRB* Educational Package.

In order to demonstrate the capabilities of the package the paper shall now follow a typical classroom session. The tour begins with the definition of the type of the robotic bird and the numerical parameters, namely the wing and tail area, total mass of the bird, maximum flapping speed and the inertia of the bird (Fig. 8a). Later on, these numerical parameters can be viewed/changed and saved. As it can be seen in Fig. 8b, it will be possible to choose parameterized birds based in previous studies [7]. The primarily advantage inherent to the existence of a list birds, is to help users to setup the different parameters. Some bird parameters, like the inertia and the wing area, may be a little bit hard to choose. Therefore, selecting a bird with similar characteristics of the one desired can prove to be useful.

The parameters can easily be configured by the user. Even so, we will give some emphasis to the determination of the wing area and its inertia. To easily calculate the wing area based on the size of the wings (Fig. 9) we can use the following approximated method.

$$S = L_2\,(H_1 + H_2) + L_1\,(H_2 + H_3) \qquad (2)$$

The easiest way is to approximate the body of the bird to a rectangular parallelepiped as shown in Fig. 10.

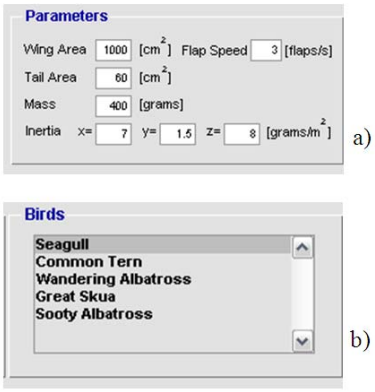


Fig. 8. Setup parameters for the robotic bird.

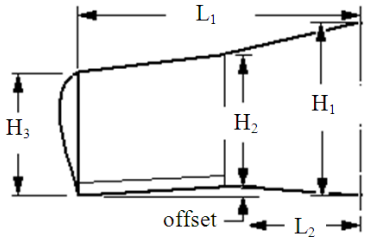


Fig. 9. Approximated method to calculate the wing area.

The inertia moment I in (x,y,z) axis, acting in the center of gravity of the rectangular parallelepiped, can then be calculated by the following equations:

$$I_x = m\frac{b^2+c^2}{12}; I_y = m\frac{a^2+b^2}{12}; I_z = m\frac{a^2+c^2}{12} \qquad (3)$$

After choosing the desired parameters, the user can choose the action that will be realized by the bird. After choosing the desired action some options will appear. Those options may vary depending on the action. For example, if the desired action is to go in a straight line then the bird initial velocity will be requested (Fig. 11). But, if the desired action it to go up or down, besides the initial velocity, the vertical distance to travel will also be requested (Fig. 12).

While the simulation is running the user will have the opportunity to see the charts of the velocities in x,y and z axis being constructed (Fig. 13) as well as the 3D animation of the bird.

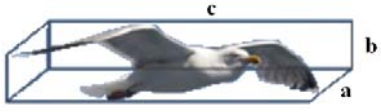


Fig. 10. Approximation of body for rotation.

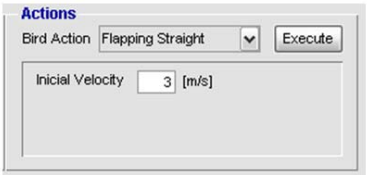


Fig. 11. Setup action to fly in a straight line with an initial velocity of 3 m/s.

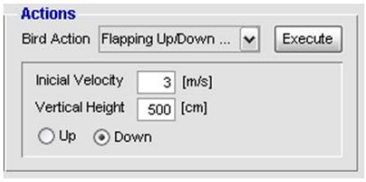


Fig. 12. Setup action to fly down a vertical distance of 5 m with an initial velocity of 3m/s.

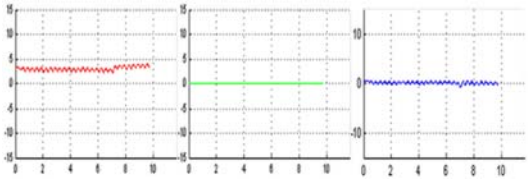


Fig. 13. Charts of the velocities in (*x*, *y*, *z*) axis.

To better watch the 3D animation the user can change the camera and can rotate and zoom while the simulation is running. Some actions, like flying up or down, will stop if the objectives are completed; others will never stop until the user wants to do so. After the action is realized the simulation can be saved to load it later in order to compare to other simulations. A grid line corresponding to the previously loaded simulation will appear in the charts of the velocities and the trajectory of the bird in the 3D animation (Fig. 14).

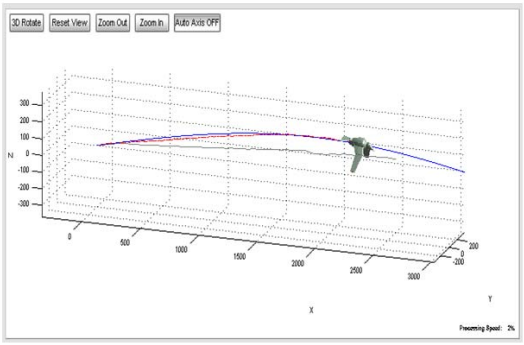


Fig. 14. Comparing trajectories made by different birds.

V. ELECTRONIC SYSTEM

Fig. 15 shows the electronic system implemented in the robotic bird. Basically the system consists on a electronic board with a *PIC18F258* microcontroller, a *LM2576-ADJ* chopper voltage regulator, an *ER400TRS* radio frequency module and the six servomotors corresponding to the six controlled joints.

A. Electric Actuators

The actuators used in the robot are servomotors. A servo is a small device that has an output shaft that can be positioned to specific angular positions by sending the servo a PWM signal. As long as the coded signal exists in the input line, the servo will maintain the angular position of the shaft. As the duty cycle changes, the angular position of the shaft varies. In practice, servos are used in radio controlled airplanes to position control surfaces

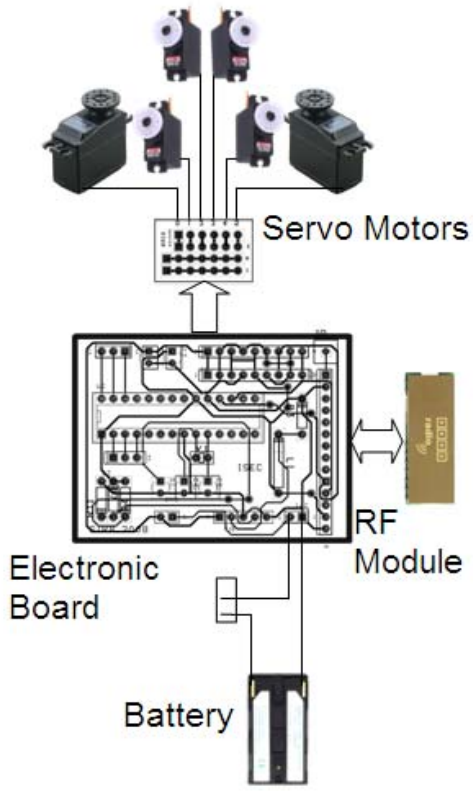


Fig. 15. Electronic System of the Robot.

like the elevators and rudders. They are also used in radio controlled cars, puppets, and of course, robots.

Servos are extremely useful in robotics. The motors are small with built in control circuitry and are extremely powerful for thier size. A standard servo such as the *HS-81MG* from *Hitec* has 2.6 Kg/cm of torque with a speed of 0.11 sec/60 with a small weight of 19 grams and measuring just 29.8x12x1.00 mm. Those were the servos used to control the tail and the angle of attack of both wings independently (Fig. 16a).

Fig. 16. Servo a) *HS-81MG* from *Hitec* b) *S9254* from *Futaba*.

A regular standard servo couldn't be used to control the flapping wings. Even having a great relation torque-speed it would not be enough. We then used digital servos. Digital servos are controlled no differently than analog servos. The difference is in how the servo motor is controlled via the circuit board. The motor of an analog servo receives a signal from the amplifier 30 times a second. This signal allows the amplifier to update the motor position. Digital servos use a high frequency amplifier that updates the servo motor position 300 times a second. By updating the motor position more often, the digital servo can deliver full torque from the beginning of movement and increases

the holding power of the servo. The quick refresh also allows the digital servo to have a tighter deadband. With the exception of a higher cost, there are only advantages for digital servos over analog servos. The digital micro processor is 10 times faster than an analog servo. This results in a much quicker response from the beginning with the servo developing all the rated torque 1 degree off of the center point. The standing torque of a digital servo is 3 times that of its analog counterpart. This means digital servos are typically smaller and have more torque. The digital servo used on the wings is the *S9254* from *Futaba*, with a speed of 0.06 sec/60, a torque of approximately 3.4 Kg/cm, a small weight of 49 grams and 41x20x36mm (Fig. 16b). This servo is very fast and commonly used in helicopters as well as other kind of applications that needs a very fast output speed.

B. Power Supply

To feed the whole system we need a battery with high durability, little, light and small costs. Lithium ion batteries are commonly used in consumer electronics. They are currently one of the most popular types of battery for portable electronics, with one of the best energy-to-weight ratios, no memory effect, and a slow loss of charge when not in use. In addition to uses for consumer electronics, lithium-ion batteries are growing in popularity for defense, automotive, and aerospace applications due to their high energy density. However certain kinds of mistreatment may cause Li-ion batteries to explode.

We used a Li-Ion 7.2V/1850mAh battery from Duracell commonly used in digital movie cameras. One of the primary advantages of using this battery is its weight being approximately 90 grams and its dimensions (70x38x20.5mm). Those batteries require periodically attention. They have a specific charging time with a specific battery recharger. In order to supply the regular 5V to the electronic system we used the step down switch voltage regulator *LM2576-ADJ* showing a great improvement in current consumption when comparing to the regular *LM7805* or other similar regulator.

VI. PHYSICAL STRUCTURE

A. Robot Construction

The construction of flying models should follow the principles of simplicity, slightness and robustness. Thus, the wooden raft, given to its low density and the enormous easiness which it can be worked out, is one of the basic materials in the construction of flying models. A main part of the bird is the wing. It is responsible for generating the forces that will raise the bird of the ground. It's in the construction of the wing, therefore, that becomes necessary to deposit a well-taken care and special attention. The wings had been made with wooden raft and carbon rods giving a good resistance and low weight. To give form to the wings, we connected the various airfoils made in raft with laths of raft and carbon tubes to strengthen the structure (Fig. 17).

To be able to have a wing movement similar to the one of real birds we used a set of springs and hinges in order to construct a mechanical spring mechanism in the wrist joint (Fig. 18). A thermal adhesive isolator was used to cover the structure of the wings.

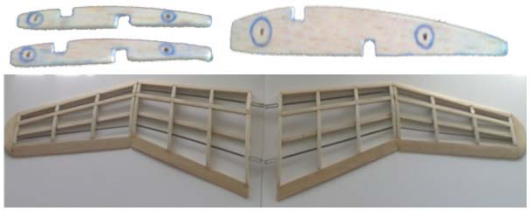


Fig. 17. Wings made with wooden raft and carbon rods.

Fig. 18. Mechanical spring mechanism in the wrist joint.

To connect the servomotor that will define the angle of attack of the wings, we used pine wood. Although is heavier than the raft, it was the indicated option to resist to the coupling between the engine and the wings (Fig. 19).

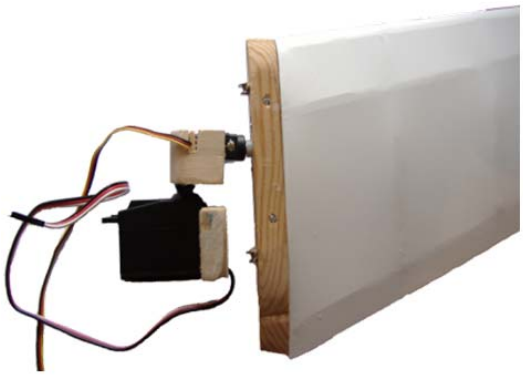


Fig. 19. Coupling between the servomotor and the wing.

Similarly to the wings, the tail was made in raft and afterward isolated with the same thermal adhesive material. The coupling between the tail and the servomotors was also similar to the one made in the wings using pine (Fig. 20).

The fuselage of the bird was made in fiberglass being a very resistant material and easy to be molded. Isolating everything with lycra and adding feathers made with a flexible plastic film to achieve a greater wing area we obtain the following result (Fig. 21).

B. Experimental Results

After the construction of the robot, we accomplished some experiences. We started with isolated tests for each motor to obtain the position and velocity limits. An initial position, the start position, is initially fixed for each one of

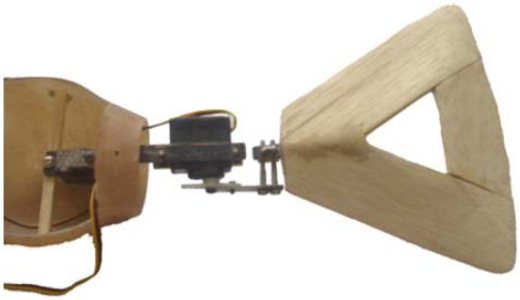


Fig. 20. Coupling between the servomotor and the tail.

Fig. 21. Robotic bird.

them. After receiving the start signal, all the motors will go to this position. As said before, contrarily to the standard servos used in the other joints, for the wing beat we used digital servos allowing a better relation force/speed. These servos can make, without any load, a rotation of 60 in 0.06 seconds. In the first test, we didn't use the flexible plastic film to simulate the effect of feathers. We made a great wing beat speed of approximately 640 ms per cycle. Fig. 22 illustrates an image sequence of one wing beat cycle.

Fig. 22. Image sequence showing one wing beat cycle.

However, the area of the wings was not enough when compared to the weight of the robot. In the second test we used the flexible plastic film to simulate the feathers getting a greater wing area. The application of the film took us to another problem. We used glue to adhere the film to the Lycra increasing the weight of each wing in 70 grams. The engine speed had then considerably decreased making it impossible to oppose the weight of the bird. Other solutions are still being implemented.

VII. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

Some satisfatory results were obtained with the simulation platform proving that the development of the kinematical and dynamic model can show us the behavior of the bird. It is possible to simulate all kind of closed loop actions like gliding, flapping wings, taking off, landing, following trajectories and others. Information relative to the physical nature of the flapping flight proved to be important to analyze solutions. The results had been evaluated using an intuitive analysis, and equally validated by other preceding analysis in this area. The robotic platform gave us some problems that could easily be avoided if the construction was implemented after all the dynamical analysis. Two servos where used in order to control the wing beat of each wing independently. However, such would not be necessary. To achieve identical movements, we can simply change the angle of attack of each wing and the tail rotations.

B. Future Works

We could change a lot of different characteristics in the physical structure of the bird such as using another kind of material instead of fiberglass to construct the body. The fiberglass, although relatively light, it still corresponds to almost one-third of the global weight of the robot. The body could be constructed using raft or another kind of light material as carbon fiber.

Relatively to the digital servomotors used in the wings, they could be substituted by a different system. Using only one DC engine connected to a V-system making both wings to flap simultaneously we could benefit of bigger force and speed.

REFERENCES

[1] Micael S. Couceiro, Carlos M. Figueiredo, N. M. Fonseca Ferreira, J. A. Tenreiro Machado, *Simulation of a Robotic Bird*, 3rd IFAC Workshop on Fractional Differentiation and its Applications, Ankara, Turkey, 05-07 November 2008.
[2] R. Zbikowski, *Fly Like a Fly*, IEEE Spectrum, 42(11), 46-51, 2005.
[3] J. Grasmeyer and M. Keennon, *Development of the Black Widow Micro Air Vehicle*, Proc. of the 39th AIAA Aerospace Sciences Meeting and Exhibit, NV, USA, AIAA Paper No. AIAA-2001-0127, 2001.
[4] A. Colozza, *Fly Like a Bird*, IEEE Spectrum, 44(5), 38-43, 2007.
[5] H. Rbinger *How Ornithopters Fly*, 2006.
[6] N. M. Fonseca Ferreira, J. A. Tenreiro Machado, *RobLib: An Educational Program for Robotics* , SYROCO'00, IFAC Symposium on Robot Control, vol.1 pg. 163-168, Vienna, Austria, 2000.
[7] Charles H. Blake *More Data on the Wing Flapping Rates of Birds*, 1948.

robotica 2009
festival nacional de robótica

Supervised Group Size Regulation in a Heterogeneous Robotic Swarm

Rehan O'Grady, Carlo Pinciroli, Anders Lyhne Christensen and Marco Dorigo

***Abstract*— In this study, we work with a heterogeneous swarm of wheeled and aerial robots. We present a self-organised approach inspired by the aggregation behaviour of cockroaches that allows each aerial robot to form a dedicated group of wheeled robots of a particular size. Our approach is based on simple probabilistic rules, but still proves robust and flexible. Different groups can be formed in parallel, and the size of the groups can be dynamically modified. Our work is based on a real robotic platform that is still under development—here, we present results and analysis of extensive simulation-based experiments. We also present a mathematical analysis of our system.**

I. Introduction

We consider a scenario in which a range of different tasks must be carried out by a heterogeneous swarm made up of flying robots (eye-bots) and wheeled robots (foot-bots). Each task is physically executed by a dedicated group of foot-bots. The eye-bots are responsible for exploring the environment, determining the optimal foot-bot group size for each task, aggregating foot-bots into groups of the relevant sizes, and finally for guiding the groups of foot-bots to the task sites.

In this study, we focus on the aggregation and group size regulation aspect of the above scenario. Aggregation is a fundamental process that has been studied in many different contexts such as biology [8], physics [16] and robotics [5]. The aggregation behaviour of cockroaches in an environment containing shelters is relatively simple and has been well explored by biologists—its dynamics can be accurately modelled at the individual cockroach level by mapping local environmental conditions to simple stop/go probabilities. Previous robotics studies have shown how this behaviour can be faithfully mimicked by a group of robots. Existing robotic implementations share key features of the cockroach model. In particular, the equilibrium distribution of agents depends passively on the initial configuration of the environment and on the static mapping of environmental conditions to stop/go probabilities.

In this paper, we take inspiration from the cockroach aggregation model, but extend it so that the equilibrium distribution of cockroach like agents (foot-bots in our case) can be dynamically controlled by the system. To do this, we treat our eye-bots like 'active' shelters under which the foot-bots aggregate. The eye-bots are active in the sense that they can dynamically alter the stop/go probabilities for the foot-bots aggregating underneath them.

We model our system mathematically, and then implement it using simulated eye-bots and foot-bots (the robots we simulate are based on a real robotic platform that is still under development). We demonstrate the feasibility of our system and show that it exhibits several desirable features, namely *stabilisation*, *redistribution* and *balancing* (these concepts are elaborated in Section III).

Carlo Pinciroli, Rehan O'Grady and Marco Dorigo are with Université Libre de Bruxelles, Brussels, Belgium ({cpinciro,rogrady,mdorigo}@ulb.ac.be). Anders Lyhne Christensen is with DCTI at Lisbon University Institute, Portugal (anders.christensen@iscte.pt).

II. Related Work

There is a large body of literature on heterogeneous robotic groups [7], [15]. However, to the best of our knowledge, no existing robotic study investigates group size regulation in heterogeneous robot groups. There is some literature on aggregation and group size regulation in homogeneous groups. Dorigo *et al.* [1] evolved two dimensional distributed aggregation in a swarm of embodied robots, Martinoli *et al.* [5] investigated the effects of probabilistic parameters on the size of object clusters collected by Khepera robots. Neither work provided an explicit group size control mechanism. Melhuish *et al.* [6] controlled group sizes in a swarm of abstracted agents using a firefly-like synchronisation mechanism. However, group size control was not fine grained to the level of individual robots, only one group was formed at a time and the physics of an embodied system was not taken into account.

In a series of experiments in a white circular arena, Jeanson *et al.* [3] derived a probabilistic behavioural model of the first instar larvae of the German cockroach *Blattella germanica*. They showed that individuals switch probabilistically between two alternative behaviours: random walk and resting. Analysis revealed that the probability for a cockroach to switch to (or remain in) one of these two states depends on the number of resting cockroaches in its neighbourhood (in direct antenna contact): as this number increases, the stopping probability also increases, while the probability of leaving an aggregate decreases. Furthermore, experimental evidence shows that cockroaches prefer to aggregate in dark places [13]. If multiple shelters are present in the environment, the majority of cockroaches aggregate under only one shelter rather than spreading evenly among the different shelters. A positive feedback mechanism ensures that this happens even when the shelters are perfectly identical [4].

Garnier *et al.* [2] used Jeanson's behavioural model to show that a group of cockroach-like robots can achieve a collective choice between two different shelters in the environment through simple local interactions.

III. Goals

Our system must be able to cope with the dynamic arrival and departure of eye-bots and foot-bots during system execution. It is also reasonable to assume that there will sometimes be insufficient foot-bots available to carry out all tasks at the same time, i.e., there may not be enough foot-bots to fill the quota of every eye-bot (*quota* refers to the desired group size of a single eye-bot). We have identified three key features that we believe

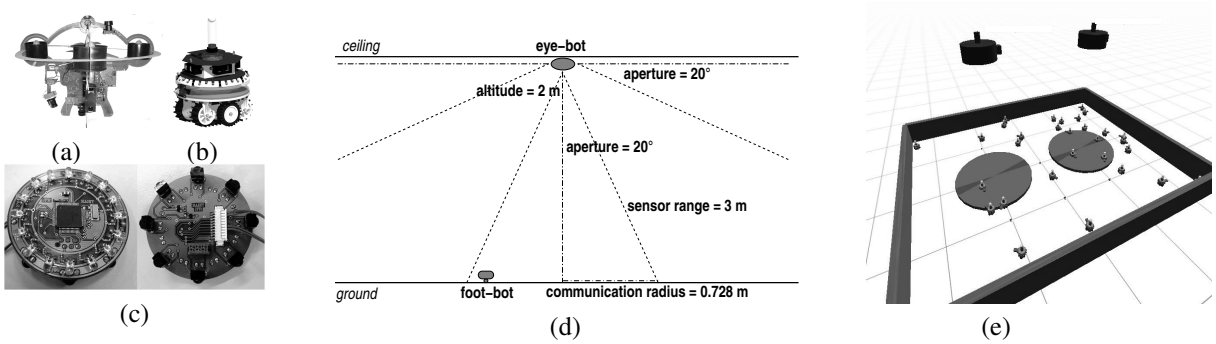


Fig. 1. The robots used in this study. At the time of writing, the robotic hardware is still under development. (a) The eye-bot (aerial robot). (b) The foot-bot (wheeled robot). (c) Hardware prototype of the range and bearing system. (d) The communication range of the eye-bot. (e) The simulation environment. The grey circles represent the signal range of the eye-bot's vertical range and bearing communication system—i.e., the area in which each eye-bot can aggregate foot-bots.

are necessary for a dynamic group size regulating system, namely *stabilisation*, *redistribution* and *balancing*.

Stabilisation means that the system settles down to an equilibrium state that is sufficiently stable to allow each eye-bot to independently determine that equilibrium has been reached (and therefore that it is reasonable to stop aggregating and initiate a subsequent behaviour). This is not trivial when there are insufficient foot-bots to meet the quotas of every eye-bot.

Redistribution means that a running system can respond to the dynamic addition or removal of robots of either type, and efficiently redistribute foot-bots accordingly. We would expect the natural dynamics of the system to encourage redistribution, rather than having to use communication to explicitly coordinate redistribution.

Balancing means that when there are insufficient foot-bots, the system stabilises in a state where each eye-bot has filled the same percentage of its quota of foot-bots (independently of the quota size). This is important as it ensures that no eye-bots arbitrarily take precedence over other eye-bots. Consider a future implementation of this system in which tasks have different priorities. In the case where there are not enough foot-bots, the balancing property will ensure that quotas with low priority are not filled at the expense of quotas with high priority[1].

IV. Robotic Platform and Simulation Environment

We consider a heterogeneous robotic system composed of two types of robots: aerial robots (eye-bots) and wheeled robots (foot-bots). Eye-bots are quad-rotor equipped robots capable of flying and attaching to the ceiling. For the purpose of this study, we assume that the eye-bots are always attached to the ceiling. Eye-bots are equipped with a high resolution camera which allows them to monitor what happens on the ground [12], see Figure 1(a). Foot-bots, on the other hand, move around on the ground. They are equipped with infrared proximity sensors, an omnidirectional camera, and an RGB LED ring that enables them to display their state to robots within visual range, see Figure 1(b).

The eye-bots communicate with the foot-bots using a range and bearing system [11] mounted on both robots see Figures 1(c) and 1(d). This system allows the eye-bots to locally broadcast and receive messages either from other eye-bots in the same plane, or to foot-bots in a cone beneath them. Furthermore, the system allows for *situated communication*. This means that recipients of a message know both the content and the physical origin of the message within their own frame of reference.

At the time of writing, the robotic hardware is still under development. For this reason, the results presented in this paper have been obtained in simulation. A custom physics based simulator called ARGoS [9] has been developed to reproduce the dynamics of the robots' sensors and actuators with reasonable accuracy. A screen shot of simulation environment is shown in Figure 1(e).

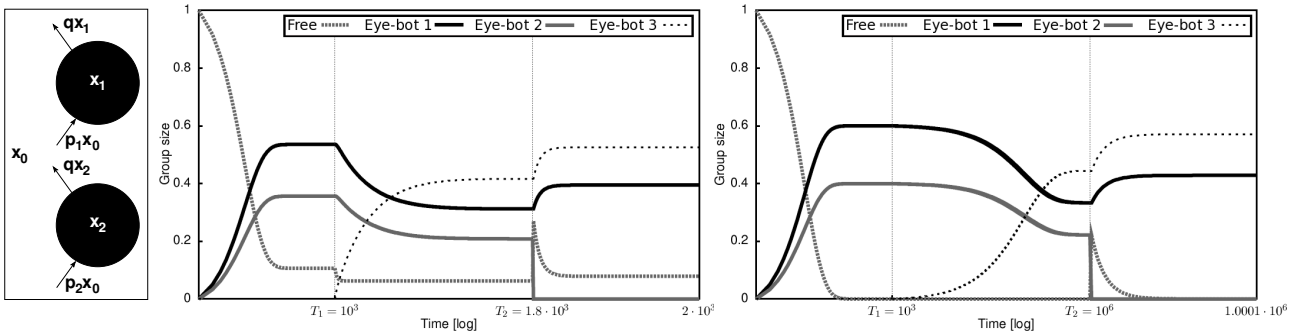
V. Methodology

In our biologically inspired system, we let foot-bots play the role of cockroaches, while eye-bots play the role of shelters. Unlike the static shelters in Garnier's previous system [2], the eye-bots in our system actively broadcast varying stop and go probabilities. By changing the stop and go probabilities that it broadcasts, an eye-bot can actively influence the number of foot-bots in its group. Like cockroaches, foot-bots can be moving (state FREE) or stationary (state IN_GROUP). A foot-bot is considered part of an eye-bot's aggregate if it is underneath it (i.e., within range of that eye-bot's range and bearing communication signal—see Figure 1(d)) and it is in state IN_GROUP.

Foot-bots in state FREE perform a random walk in the arena. At each time step, each eye-bot i sends a message containing two pieces of information (that will be received by any foot-bots underneath it): the probability for a foot-bot in state FREE to join the group (j_i) and the probability for a robot in state IN_GROUP to leave it (q).

Throughout this paper, we use a single experimental setup to test the various configurations of our system. We use this experimental setup to test both our abstract mathematical models and our concrete implementations on the robotic platform. In particular, we designed the

[1]In the future, an additional deadlock resolution mechanism will be needed to allow the system to take meaningful action when the system stabilises in a state where none of the eye-bot quotas have been filled. In the priority based system, for example, after stabilisation the eye-bots could each have a probability of releasing some of their aggregated foot-bots. These release probabilities could be linked to the priority of each eye-bot's task — the higher the priority of an eye-bot's task, the lower the probability of releasing foot-bots. Eye-bots with high priority tasks would thus be likely to fill their quotas at the expense of eye-bots with low priority tasks. In this study, we focus on the underlying balancing property that would enable this kind of probabilistic priority mechanism.



(a) (b) $q = 0.01$; Quotas: $[15, 10, 20]$ (c) $q = 0.00001$; Quotas: $[15, 10, 20]$

Fig. 2. (a) A schema of the mathematical model. (b,c) Experiments run using the mathematical abstraction of the *Constant-q* strategy for two different values of q. The eye-bot quotas for both experiments are: eye-bot 1: 15, eye-bot 2: 10, eye-bot 3: 20.

experimental setup to determine whether the different configurations of our system display the desirable properties discussed in Section III. In our setup, we use a rectangular arena containing 30 non-aggregated foot-bots and 3 eye-bots. Our experiment consists of three phases.

In the first phase, only eye-bot 1 and eye-bot 2 are active. The quotas of eye-bots 1 and 2 are always set so that their sum is less than 30. This first phase therefore allows us to test whether the system is capable of stabilising to the correct group sizes in the simple case where there are enough foot-bots to satisfy all eye-bot quotas. After a certain amount of time T_1, eye-bot 3 is activated. This initiates the second phase, in which all three eye-bots are active. This phase tests the redistribution property of the system in response to the arrival of a new eye-bot. It also tests the balancing property of the system, as eye-bot 3's quota is always chosen so as to make the sum of the quotas greater than the total number of foot-bots in the arena. We also always set eye-bot 3's quota to be greater than or equal to the quotas of eye-bot 1 and eye-bot 2, thus requiring the system to redistribute foot-bots efficiently. The third and final phase is initiated at time T_2 by the disactivation of eye-bot 2, and again tests both the redistribution and the balancing properties of the system, this time in response to the addition of more free foot-bots into the arena. The stabilisation property of the system can be tested in all three phases of the simulation based experiments—ideally, each eye-bot should be able to detect stabilisation in all phases.

VI. An Initial Implementation

In this section, we describe and analyse a simple implementation of our system. For each eye-bot, we set the join probability proportional to the eye-bot's quota of desired foot-bots. We keep the leave probability common for all eye-bots and constant over time. Therefore, we subsequently refer to this implementation as the *Constant-q* strategy.

Figure 2(a) illustrates the abstraction of the system that we use for mathematical analysis. In Figure 2(a), the arena is rectangular and the communication range of each eye-bot is drawn as a circular grey area. Let the total number of foot-bots in the arena be n and the number of foot-bots aggregated at time step t under eye-bot i be $g_i(t)$. Then, the fraction $x_i(t)$ of foot-bots aggregated under eye-bot i at time t is given by:

$$x_i(t) = \frac{g_i(t)}{n}.$$

Analogously, the fraction $x_0(t)$ of free (i.e. not part of any eye-bot's aggregated group) foot-bots present in the arena at time t is given by:

$$x_0(t) = \frac{n - \sum_i^n g_i(t)}{n} = 1 - \sum_i^n x_i(t).$$

We define p_i and q as constant probability parameters of our model. During a single time step, each free foot-bot (i.e., every foot-bot that is not part of any eye-bot's aggregated group) has probability p_i to join the aggregated group under eye-bot i. Each aggregated foot-bot has probability q of disaggregating (i.e., leaving the group of foot-bots which it had previously joined).

Foot-bots perform random walk with obstacle avoidance, thus spreading uniformly in the environment (see Section VII for details). When a foot-bot enters the communication range of eye-bot i, it joins the aggregate with a constant probability j_i. Under these assumptions, an easy way to calculate p_i is the following:

$$p_i = \frac{\text{Area(eye-bot)}}{\text{Area(arena)}} j_i$$

The analytical expression of the mathematical model is then:

$$\begin{cases} x_0(t+1) = x_0(t) - (\sum_i^n p_i)x_0(t) + q\sum_i^n x_i(t) \\ x_1(t+1) = x_1(t) + p_1 x_0(t) - q x_1(t) \\ \quad\vdots \\ x_n(t+1) = x_n(t) + p_n x_0(t) - q x_n(t) \end{cases}$$

and by imposing the steady-state condition

$$\forall k \in [0, n] \qquad x_k(t+1) \equiv x_k(t) \equiv x_k^*$$

we can derive the expression for x_k^* at convergence (see [10] for the full mathematical derivation):

$$x_0^* = \frac{q}{\sum_i^n p_i + q}$$
$$x_1^* = \frac{p_1}{\sum_i^n p_i + q}$$
$$\vdots$$
$$x_n^* = \frac{p_n}{\sum_i^n p_i + q}$$

Figures 2(b) and 2(c) illustrate the dynamics of our mathematical model of the *Constant-q* strategy for two different values of q. Notice that for $q = 0.01$ a significant portion of foot-bots is free when the system reaches its first equilibrium with two eye-bots, while for $q = 0.00001$ this portion is very close to zero. At $t = T_1$, the third eye-bot is added. Since $q > 0$, some foot-bots leave their aggregated groups and a new equilibrium is reached (redistribution). The value of parameter q affects the speed of convergence[2] to the new equilibrium: for $q = 0.01$ convergence in this second phase is three orders of magnitude faster than for $q = 0.00001$ because many more foot-bots are free to join eye-bot 3. At $t = T_2$, eye-bot 2 leaves the arena and frees all its foot-bots. A new equilibrium is quickly reached in about 100 time steps.

VII. Transferal onto Robotic Platform

A. Counting and Probabilities

In the mathematical model, there is no sense of convergence to a particular number of robots. Each abstract eye-bot converges to a fraction of the total number of robots proportional to its join probability (j_i). The first step in transferring our system onto an embodied platform is, therefore, to introduce the notion of counting.

Counting is achieved by each eye-bot monitoring the number of foot-bots underneath it using its camera. For each eye-bot i, when the aggregated foot-bot group size is smaller than its quota, it must have a non-zero join probability ($j_i > 0$). As soon as its quota is filled, the eye-bot prevents any more foot-bots from joining its aggregated group, by setting its join probability to zero ($j_i = 0$).

Clearly, we must somehow choose values for j_i and q. A natural choice for the value of j_i is to consider the fraction of the maximum foot-bot aggregated group size represented by the quota of eye-bot i. The maximum aggregated group size is determined by the physical size of the foot-bot and the communication range of the eye-bot[3]. We therefore set:

$$j_i = \frac{\text{eye-bot } i \text{ 's quota}}{\text{max foot-bot aggregated group size}}.$$

To find a sensible value of q is non–trivial—the impact of q may even depend on the density of robots in the arena. An incorrect choice of q can have dramatic effects on system performance. Note that the mathematical model treats probabilities p_i and q as the rates of free foot-bots joining and leaving aggregates respectively, while in reality this is only true as an average over time. In a realistic implementation, fluctuations are always present and perturb the expected dynamics of the system. We found that when q is only one order of magnitude smaller than p_i, the rate of robots leaving eye-bot i's aggregate over time is too high, thus making that aggregate unstable. On the other hand, choosing too low a value for q hinders redistribution. The role of q in this system is discussed in more detail in Section VII-F.

B. Physical Interference

The physical dynamics of embodied agents make the successful transferal of any model onto a real robotic platform non-trivial. We found that in a naïve implementation of our system, physical interference between robots disrupted the effects of the transmitted probabilities, catastrophically changing the resulting equilibrium. For example, a moving foot-bot would take a long time to traverse the space underneath an eye-bot that had already aggregated a large number of foot-bots, due to the overhead of using obstacle avoidance to thread a path through the aggregated foot-bots. In such cases, both arriving (non-aggregated) and leaving (disaggregating) foot-bots would repeatedly apply the join probability transmitted by the eye-bot. As a result, foot-bots tended never to leave large aggregates.

We also noticed other more direct physical effects. For example, to let disaggregating foot-bots leave an aggregate, it was necessary to make the other aggregated foot-bots get out of the way (using obstacle avoidance). However, this had the result that aggregated foot-bots positioned near the edge of an eye-bot's signal range often got pushed out of the aggregate by leaving foot-bots.

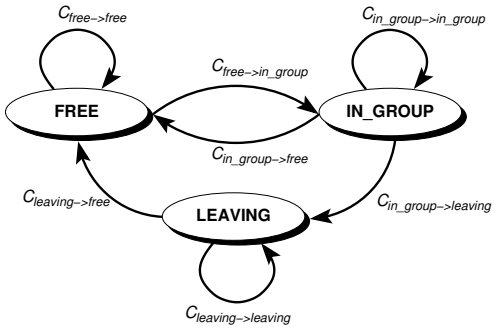
To get around these issues, we implemented a clustering technique, in which aggregating foot-bots were attracted to each other and to the centre of the eye-bot with which they were associated. Using this clustering algorithm as a base, our high level foot-bot state transition behaviour was a much closer match to the behaviour of the mathematical model presented in the previous section.

C. Clustering Algorithm

Our distributed clustering algorithm uses only local interactions. We consider each foot-bot to be immersed in two virtual potential fields. The minimum energy of the first potential field is at the point of the vertical projection of the centre of the eye-bot on the ground. The foot-bots calculate this potential field using information from the transmitted range and bearing messages coming from an eye-bot. The second potential field is analogous to the behaviour of molecules in physical systems. Using its camera, each aggregated foot-bot measures the distance to its neighbouring foot-bots and calculates the potential field whose minimum energy configuration is at distance σ_S to all neighbouring foot-bots. Each foot-bot superimposes its two fields to come up with a resultant force that it translates into appropriate angular velocities that it applies to its wheel actuators. The result is that the local system of aggregated foot-bots converges quickly towards its minimum energy configuration: a hexagonal lattice of foot-bots centred around the vertical projection of the eye-bot under which they are aggregating [14].

[2]Convergence in the mathematical model is the equivalent of the stabilisation property in the embodied robotic implementation (see section VII).

[3]By manually experimenting with the foot-bot hardware prototype, we came up with the (approximate) value of 25 as the number of foot-bots that could aggregate under an eye-bot without being so close as to collide.



State transition conditions	
$C_{free \to in\_group}$	`WithinRange()` = *true* and `Rand()` $< j_i$
$C_{free \to free}$	`WithinRange()` = *false* or `Rand()` $> j_i$
$C_{in\_group \to free}$	`WithinRange()` = *false*
$C_{in\_group \to leaving}$	`WithinRange()` = *true* and `Rand()` $< q$
$C_{in\_group \to in\_group}$	`WithinRange()` = *true* and `Rand()` $> q$
$C_{leaving \to free}$	`WithinRange()` = *false*
$C_{leaving \to leaving}$	`WithinRange()` = *true*

Fig. 3. State transition logic for foot-bots at each time step. `WithinRange()` is a function returning *true* when the robot is within the communication range of an eye-bot. `Rand()` is a function returning a random number in $\mathcal{U}(0,1)$. j_i is the join probability for eye-bot i, q is the common disaggregation probability. State transition conditions are represented be the symbol C and a subscript. For example, $C_{in\_group \to in\_group}$ represents the conditions under which an aggregated foot-bot will stay aggregated in its group (i.e., will not disaggregate) in a single time step.

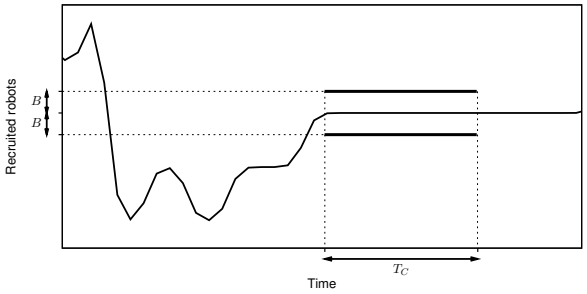


Fig. 4. Convergence detection is based on a tolerance boundary B which is a function of the leaving probability q, the current size of the aggregate g_i and the length of the monitoring period T_C.

Analytically, the potential field attracting a foot-bot towards the centre of its associated eye-bot at distance r_E is given by:

$$V_E(r_E) = \eta r_E^3$$

The equation for the second potential field that manages interactions between two aggregated foot-bots at mutual distance r_F is given by

$$V_F(r_F;\sigma_S) = \epsilon\left[\left(\frac{\sigma_S}{r_F}\right)^4 - 2\left(\frac{\sigma}{r_F}\right)^2\right]$$

Thanks to the fact that force $\vec{F}(r) = -\nabla V(r)$ and defining $\bar{r_E}$ ($\bar{r_F}$) as the normalised vector pointed from the centre of the foot-bot towards the centre of the eye-bot (foot-bot), we can derive

$$\vec{F}_E(r_E) = -3\eta r_E^2 \bar{r_E}$$

and

$$\vec{F}_F(r_F;\sigma_S) = \frac{4\epsilon}{r_F}\left[\left(\frac{\sigma_S}{r_F}\right)^4 - \left(\frac{\sigma_S}{r_F}\right)^2\right]\bar{r}_F.$$

Distance r_E is obtained from the range and bearing sensor, and distance r_F from the camera. At each time step, each aggregated foot-bot calculates $\vec{F}_E(r_E)$ and $\vec{F}_F^m(r_F^m;\sigma_S)$ for each neighbour m. The resulting force

$$\vec{F}_{IN\_GROUP} = \vec{F}_E(r_E) + \sum_m \vec{F}_F^m(r_F^m;\sigma_S)$$

is then directly transformed into wheel actuation on the foot-bot.

A simple extension to this system provides an elegant solution to allow foot-bots to leave an aggregated group while causing minimum disruption to the rest of the group. To differentiate leaving foot-bots, we make them illuminate their blue LEDs. It then suffices to define an interaction potential between red (aggregated) foot-bots and blue (leaving) foot-bots $\vec{F}_F(r_F;\sigma_L)$ with $\sigma_L > \sigma_S$. This new potential creates a 'bubble' in the hexagonal lattice of aggregated foot-bots around the leaving foot-bot. The choice of σ_L is done in such a way that the bubble is large enough for a leaving foot-bot to pass through. For leaving foot-bots we also invert the potential field that attracts it to the eye-bot, resulting in an applied repulsive potential of $-V_E(r_E)$. With this extension, the composite force for an aggregated foot-bot becomes:

$$\vec{F}_{IN\_GROUP} = \vec{F}_E(r_E) + \sum_m \vec{F}_F^m(r_F^m;\sigma_S) + \sum_l \vec{F}_F^l(r_F^l;\sigma_L).$$

For a leaving (blue) foot-bot, on the other hand, the composite force applied is:

$$\vec{F}_{LEAVE} = -\vec{F}_E(r_E) + \sum_m \vec{F}_F^m(r_F^m;\sigma_S) + \sum_l \vec{F}_F^l(r_F^l;\sigma_L).$$

D. Foot-bot state transition logic

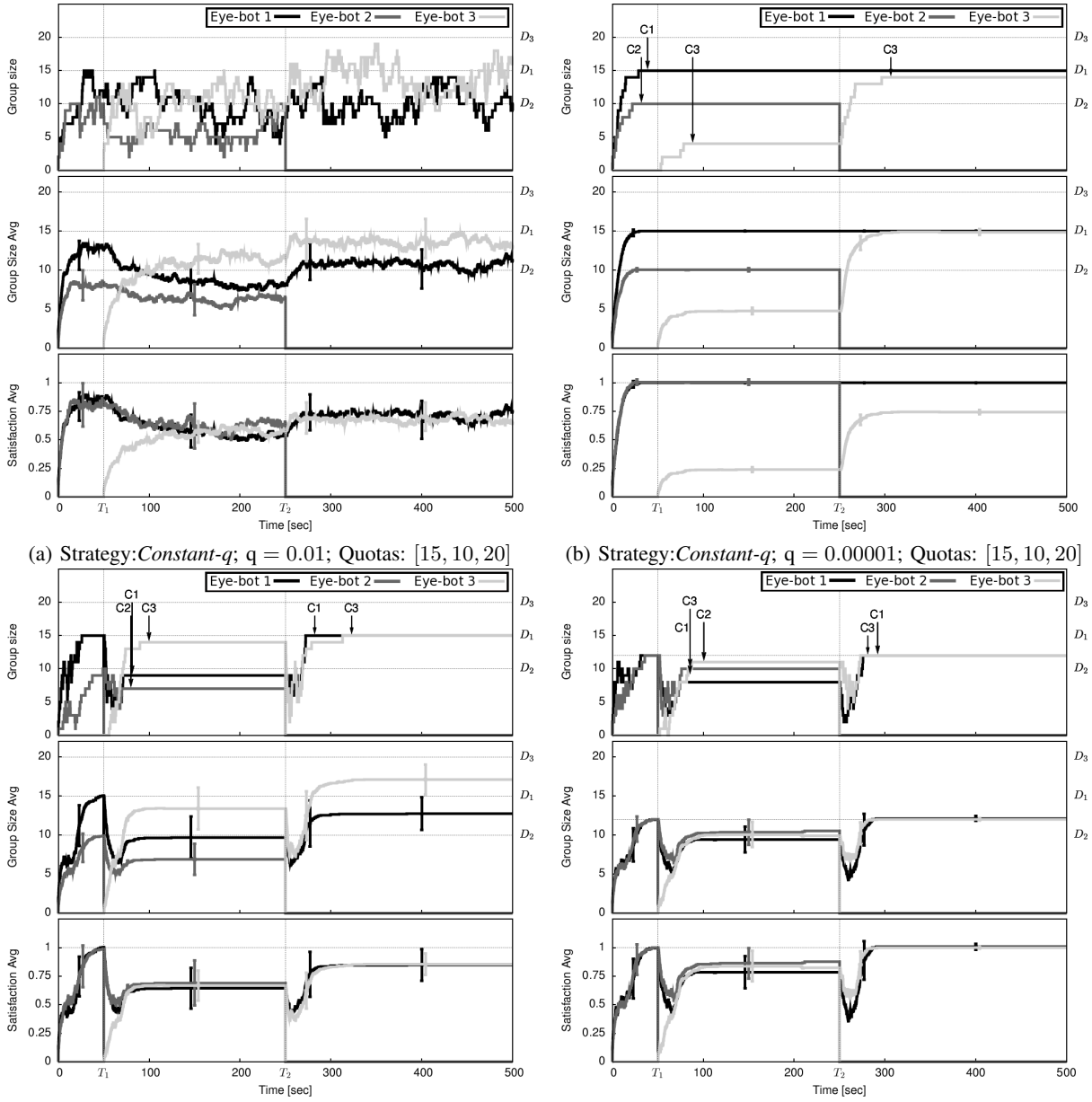
Using the clustering algorithm as a base, we define three states that the foot-bots can be in:

- state FREE: Foot-bot does not belong to any eye-bot's aggregated group. Foot-bot performs random walk with obstacle avoidance. This state is signalled with LEDs lit up in green.
- state IN_GROUP: Foot-bot is part of an aggregated group under an eye-bot. This state is signalled with LEDs lit up in red.
- state LEAVING: Foot-bot is leaving an aggregated group to which it previously belonged. This state is signalled with LEDs lit up in blue.

Eye-bots count the number of foot-bots aggregated beneath them by using their cameras to count the number of foot-bots in state IN_GROUP (i.e. lit up in red). Figure 3 shows the state transition logic of foot-bots upon receipt of a message from an eye-bot.

E. Stabilisation Detection

To detect stabilisation, each eye-bot monitors the fluctuations of the number of its aggregated foot-bots over a period T_C (see Figure 4). If fluctuations stay within some tolerance boundaries for the entire period, the eye-bot



(a) Strategy:*Constant-q*; q $= 0.01$; Quotas: $[15, 10, 20]$

(b) Strategy:*Constant-q*; q $= 0.00001$; Quotas: $[15, 10, 20]$

(c) Strategy:*Decaying-q*; Quotas: $[15, 10, 20]$

(d) Strategy:*Decaying-q*; Quotas: $[12, 12, 12]$

Fig. 5. Simulation results. Each group of 3 plots (a,b,c,d) describes a set of 20 experiments conducted with a particular system configuration. For each set, the top graph represents a single representative experiment that we selected from the set of 20, the middle graph shows the average results of all experiments (bars at selected times indicate std.dev.), the bottom graph shows the percentage of each eye-bot's quota that is filled. In every experiment, eye-bots 1 and 2 are active at the start of the experiment, eye-bot 3 is introduced at time $T_1 = 50$ s, eye-bot 1 leaves the experiment at time $T_2 = 250$ s. Quotas for eye-bots 1, 2 and 3 are given in correspondingly ordered square brackets, and shown by the correspondingly numbered symbols D_1, D_2, D_3. Symbols $C1, C2, C3$ indicate the moments at which the correspondingly numbered eye-bot detected stabilisation.

considers the system to have stabilised to its steady state. The tolerance boundary B is defined as a function of the leaving probability q, the current size of the aggregate g_i and the length of the monitoring period T_C:

$$B = \sqrt{T_C g_i q(1-q)}$$

B is derived by considering the changing number of aggregated foot-bots under an eye-bot as a time series. B is simply the formula for the standard deviation of such a time series over a given monitoring period.

F. Results

Figures 5(a) and 5(b) show the results of experiments using the *Constant-q* model with simulated robots. Looking at the sample run (top plots), we can clearly see that a lower value for q (0.00001) makes stabilisation possible (Figure 5(b)). With the higher value of q (0.01) (Figure 5(a)), there was too much noise in the system for stabilisation to be detected. However, using the lower value of q, the system does not display effective redistribution or balancing. Looking at the middle and bottom plots, we can see that for both q values, on average eye-bots 1 and 2 successfully aggregate the correct number of foot-bots between time 0 s and time T_1 (with the higher value of q displaying more noise). Once eye-bot 3 is introduced at time T_1, with $q = 0.00001$ the system does not display redistribution and balancing— the system arrives at equilibrium with eye-bot 3 having fulfilled a much smaller percentage of its quota than eye-bots 1 and 2. By contrast, with $q = 0.01$, the system displays effective redistribution

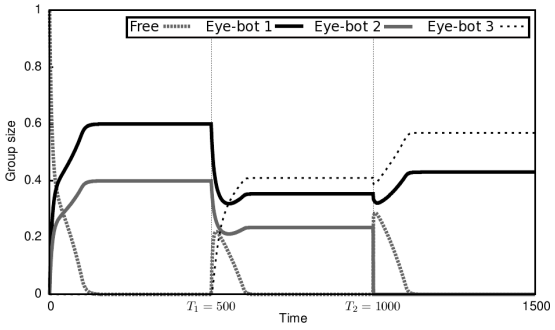


Fig. 6. *Decaying-q* strategy. Dynamics of the mathematical model. Eye-bots 1 and 2 are introduced at $t = 0$, Eye-bot 3 is introduced at $t = 500$, Eye-bot 1 leaves at $t = 1000$.

and balancing—after the introduction of eye-bot 3, the system on average quickly arrives at an equilibrium where all three eye-bots have filled the same percentage of their quota.

VIII. Decaying-q Strategy

Experiments in the previous section indicate that the *Constant-q* strategy allows the system to display the redistribution property or the stabilisation property but not both at the same time. To solve this problem, we modify our system so that individual eye-bots vary q between two values during system execution—a high q value ($q = 0.05$) that allows for efficient foot-bot redistribution, and a low q value ($q = 0.00001$) that encourages stabilisation. Parameter q is 'spiked' to the high value when the system starts, and whenever an eye-bot is activated or deactivated. After a spike, q is exponentially decayed to the low value over 20 seconds and then remains at the low value until the next spike.

The dynamics of a mathematical model of this strategy are shown in Figure 6. We can see that the new system shares the positive features of the previous linear model using both high and low q. After the introduction of eye-bot 3, the new system converges even faster than the previous *Constant-q* model with the high q of 0.01.

On the embodied robotic platform, at the moment that an eye-bot either enters the system or leaves the system, the eye-bot uses its horizontal range and bearing system (see Figure 1(d)) to communicate to local eye-bots that they should spike their q values. The results of running this model with the simulated robots are shown in Figure 5. We have run the system with three eye-bots, but with two different sets of target quotas—[15,10,20] (Figure 5(c)) and [12,12,12] (Figure 5(d)). In both cases, we can see that the system displays stabilisation, redistribution and balancing.

IX. Conclusions and Future Work

In this paper, we adapted an existing model for cockroach aggregation, in which the group sizes were determined a priori by the environment, and transformed it into an active model that could dynamically control group size. We demonstrated properties of our system with a mathematical analysis, then showed how the system could be implemented using simulated versions of a real-world robotic platform. We enhanced our system to show that the seemingly contradictory goals of redistribution and stabilisation could be achieved by a single system.

We believe that the underlying dynamics of our system are sufficiently simple that they could be implemented in other heterogeneous robotic platforms. To this end, we think it would be interesting to try other less explicit communication modalities (e.g. light intensity) as a means of transmitting probabilities.

We are currently investigating the scalability of the system as we introduce larger numbers of eye-bots. By leveraging local communication, we aim to restrict the effects of perturbations (introduction and removal of tasks and/or robots) to local regions of the system. We are also trying to embed our system as part of a more complete task execution scenario, of the type outlined in the introduction.

Acknowledgements. This research was carried out in the framework of Swarmanoid, a project funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission under grant IST-022888. Marco Dorigo acknowledges support from the Belgian F.R.S.-FNRS, of which he is Research Director.

References

[1] M. Dorigo, V. Trianni, E. Sahin, R. Gross, T.H. Labella, G. Baldassarre, S. Nolfi, J.-L. Deneubourg, F. Mondada, D. Floreano, and L.M. Gambardella. Evolving self-organizing behaviors for a swarm-bot. *Autonomous Robots*, 17(2-3):223–245, 2004.

[2] S. Garnier, C. Jost, R. Jeanson, J. Gautrais, M. Asadpour, G. Caprari, and G. Theraulaz. Aggregation behaviour as a source of collective decision in a group of cockroach-like robots. *Advances in Artificial Life*, 3630:169–178, 2005.

[3] R. Jeanson, C. Rivault, J.-L. Denebourg, S. Blanco, R. Fournier, C. Jost, and Guy Theraulaz. Self-organized aggregation in cockroaches. *Animal Behavior*, 69:169–180, 2004.

[4] A. Ledoux. Étude experimentale du grégarisme et de l'interattraction sociale chez les Blattidés. *Annales des Sciences Naturelles Zoologie et Biologie Animale*, 7:76–103, 1945.

[5] A. Martinoli, A.-J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: from probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29:51–63, 1999.

[6] C. Melhuish, O. Holland, and S. Hoddell. Convoying: Using chorusing to form travelling groups of minimal agents. *Robotics and Autonomous Systems*, 28:207–216, 1999.

[7] L.E. Parker. ALLIANCE: an architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[8] J.K. Parrish and W.M. Hammer. *Animal Groups in Three Dimensions*. Cambridge University Press, UK, 1997.

[9] C. Pinciroli. Object retrieval by a swarm of ground based robots driven by aerial robots. Mémoire de DEA, Universite Libre de Bruxelles, Bruxelles, Belgium, 2007.

[10] C. Pinciroli and R. O'Grady. A mathematical framework for symbiotic recruitment. Technical Report TR/IRIDIA/2009-006, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2009.

[11] J. Pugh and A. Martinoli. Relative localization and communication module for small-scale multi-robot systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA2006)*, pages 188–193, 2006.

[12] J. Roberts, T. Stirling, J. Zufferey, and D. Floreano. Quadrotor using minimal sensing for autonomous indoor flight. In *Proceedings of the European Micro Air Vehicle Conference and Flight Competition (EMAV2007)*, 2007.

[13] M.K. Rust, J.M. Owens, and D.A. Reierson. *Understanding and controlling the german cockroach*. Oxford University Press, UK, 1995.

[14] W. Spears, D. Spears, J. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2-3), 2004.

[15] G.S. Sukhatme, J.F. Montgomery, and R.T. Vaughan. Experiments with aerial-ground robots. In T. Balch and L.E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 345–367. AK Peters, 2001.

[16] A. Zangwill. Statistical physics: Advances in aggregation. *Nature*, 411:651–652, 2001.

This page is left blank intentionally

A slipping conditions observer in wheeled mobile robot traction

André Dias, José Miguel Almeida, Alfredo Martins, João Sequeira, Eduardo Silva

***Abstract*— In this work a discrete observer for slipping conditions in wheeled mobile robots is presented. The condition is detected at wheel level. Information about each motorized wheel adherence condition can thus be further used in higher levels of robot locomotion estimation and control. Slipping is detected only by motor current and wheel velocity measurement and analysis. Although with some limitations, it is easily applied to a wide range of systems and can be integrated in a more complex estimator using other types of information, such as global robot acceleration measurements. This approach does not depend on a priori knowledge of the operating surface or robot motor model. A computer vision based test setup is also presented. Results obtained with the observer in the ISePorto Robocup MSL robot are presented for a set of characterizing tests.**

I. INTRODUCTION

Wheeled mobile robot traction is a relevant problem for autonomous systems. The determination of slipping conditions at the wheel/surface interface without apriori knowledge of the surface characteristics has a very important role in the development of suitable locomotion control for mobile robots.

Minimization of robot slippage allows greater performance levels in actuation and improves odometric information. Early studies in autonomous vehicle motion control relied only in kinematic models [5] [7] [6]. With the development of new sensors allowing an better world perception, motion improvement was achievable [6] [8]. However part of this work focused in motion dynamics study for path tracking. Other line of development relied in the computing suitable trajectories under road holding constraints and varying terrain topography [4].

Terrain (irregularity) poses additional problems in autonomous vehicles subject to dynamic conditions. This can occur due to trajectory dynamics (induced moments and forces) or by variation in surface-wheel contact properties (due to terrain unevenness). Stringent performance requirements coupled with aforementioned traction problems introduce motion degradation capability and slippage. Thus, avoiding slippage conditions while preserving as much as possible motion within required performance is a necessary step. When unavoidable, traction loss must be dealt appropriately either by relaxing motion requirements or adapting motion planning strategy [2][3].

André Dias, José Miguel Almeida, Alfredo Martins and Eduardo Silva is with Autonomous Systems Laboratory, Instituto Superior de Engenharia do Porto, Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal adias,jma,aom,eaps@lsa.isep.ipp.pt

João Sequeira is with the Instituto Superior Técnico/ IST, Torre Norte, Av. Rovisco Pais 1, 1049-001 Lisboa,Portugal jseq@isr.ist.utl.pt

II. SLIPPAGE

A key issue in vehicle traction is the longitudinal force F_d (see figure 1). This force is due to the contact between the wheels of the vehicle and the surface where it moves on. In general, this friction force can be assumed proportional to the normal force N at the contact point.

For the sake of simplicity, in what concerns slippage analysis, the robot wheels are assumed rigid and maintaining their physical properties constant in time. The effect of passive wheels, e.g., castors, in the vehicle dynamics is also assumed to be negligible, their purpose being only to maintain the necessary static stability.

This simple model can be generalized by future work to allow complex dynamic conditions such as, for example, those arising when the inertia of each wheel varies along the trajectory, e.g., when the vehicle is requested to perform a tight curved trajectory that deforms the wheels or the structure of the robot, or when the type of contact between the wheels and the floor varies along the trajectory.

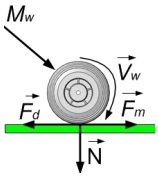


Fig. 1. Main forces acting in the wheel

Without loosing generality, for the purpose of designing a slippage detector/observer, the robot is modeled as a single wheel. The friction force (F_d in figure 1) is assumed to have the form,

$$F_d = N\mu_\lambda \tag{1}$$

where N and μ_λ respectively stand for the contact force between the robot and the ground (normal to the surface at the contact point), and the friction coefficient.

A straightforward balance equation yields

$$(F_m - F_d) = M\dot{V} \tag{2}$$

The force driving the wheel, F_m, is generated by an electrical actuator as

$$F_m = \frac{1}{r} K_m I_{motor} \tag{3}$$

with K_m, I_{motor}, and r representing, respectively, the motor intrinsic parameters, the electrical current, and the wheel radius.

Using (3) in (2) yields

$$F_d = \frac{1}{r} K_m I_{motor} - M\dot{V} \quad (4)$$

Assuming that the properties of the contact are kept constant during the movement, slippage is characterized by a sudden decrease in the friction force. Differentiating (4) thus provides the basic way to detect slippage

$$\dot{F}_d = \frac{1}{r} K_m \dot{I}_{motor} - M\ddot{V} \quad (5)$$

When $\dot{I}_{motor} < 0$ and $\ddot{V} > 0$ there is slippage occurring for sure. If $\ddot{V} < 0$ there may also be slippage, depending on the relative values of the terms in (5). However, such case is of no practical interest for our purpose as it indicates mainly a kind of slow slippage which seldom occurs in the MSL robot competitions.

Differentiating twice the observed velocity, V, may increase noise effects. Since we are interested mainly in the sign of the terms in (5), as a way to detect slippage a.s., the sign of the term $\ddot{V}$ can be replaced by the sign of the product $\dot{V}sgn(ref)$, where ref stands for the reference command signal applied to the wheel. Note that $\dot{V}sgn(ref) > 0$ means that the acceleration follows the reference signal (ref), with the velocity increasing or decreasing accordingly. Whenever $\ddot{V} > 0$ the acceleration is increasing and hence also the velocity, that is, $\ddot{V} > 0 \Rightarrow \dot{V}sgn(ref) > 0$ (a compatible reference sign is assumed). This means that testing $\dot{V}sgn(ref) > 0$ provides an optimistic form of checking for slippage (as the sufficiency condition does not hold in general) and provides the basis justification for the substitution in the slippage test that originates from (5).

III. Observer

For highly dynamic robot scenarios such as the robotic soccer environment, the terrain surface characteristics and robot dynamic conditions restricts the use of standard literature models for quantitative friction force description. These do not possess the necessary complexity to represent the robot dynamics richness and their use is also too expensive in relation to the provided information.

However, the models provide qualitative information allowing an interpretation for the robot dynamics evolution.

In view of this, using equation 4 and a state defined by motor current I_{motor} and wheel velocity V an algorithm (observer) is presented to determine the slipping condition. The algorithm observes the variables evolution and their transitions occurring in a topological frame (information hierarquization) detecting the diverse phases expressed in the friction force.

The information process hierarchy results in four discrete states determining the wheel friction condition: adherence, slipping, acceleration and breaking.

- *Slipping process* - characterized by high degree of motor current I_{motor} reduction and drastic increase of velocity V. At the time instant upon entering this state a reduction of the friction force occurs translated by a loss of adherence points to the surface resulting in an increase of velocity.
- *Adherence process* - characterized by adherence points to the movement base surface reacquiring resulting in a correspondent increase of friction force. In terms of velocity and current an increase of current I_{motor} and decrease of velocity V occurs.
- *Acceleration process* - characterized increase on both current I_{motor} and velocity V. This state happens by the transition between a static friction force condition to a dynamic friction force condition. In a accelerating stage the wheel can transit to either an adherence or slipping condition depending on the wheel/surface friction capability.
- *Breaking process* - characterized by current I_{motor} decrease and velocity V increase. In the breaking stage the wheel is transiting from a dynamic friction force condition to a static force condition. The capability to reacquire the connection points depends on the wheel/surface physics.

Algorithm 1 Observer

if $\dot{I}_{motor} < 0$ AND $\dot{V}sign(ref) > 0$ **then**
  Status: Slipping process
else if $\dot{I}_{motor} > 0$ AND $\dot{V}sign(ref) < 0$ **then**
  Status: Adherence process
else if $\dot{I}_{motor} > 0$ AND $\dot{V}sign(ref) > 0$ **then**
  Status: Acceleration process
else if $\dot{I}_{motor} < 0$ AND $\dot{V}sign(ref) < 0$ **then**
  Status: Breaking (deceleration)process
end if

The observer provides an index (*DATCOSTCS*) characterizing the friction condition and was implemented in the embedded distributed motion control system for the ISePorto mobile robots [1] .

A set of thresholds configured at the initialization stage are used to determine (when the *DATCOSTCS* exceed the threshold) the sending of a CAN bus message with the corresponding friction state and index value to the robot global motion controller (see figure 2).

Algorithm performance can be observed in figures 10, 12, 11 and 13. A qualitative analysis is provided in figures 16, 17 and 18 by the use of an external vision based motion measuring setup.

IV. ISePorto Robot

The slipping detection observer was implemented in the distributed embedded system motion control (DATCOS) on the ISePorto soccer robots (see figure 3).

The robots have conventional wheels in a differential traction arrangement with electrical DC motors. The traction control study for this traction configuration applied in the ISePortorobots, has the advantage (in comparison with the more popular omnidirectional wheel MSL robots) of being used in many other land based robots (with conventional

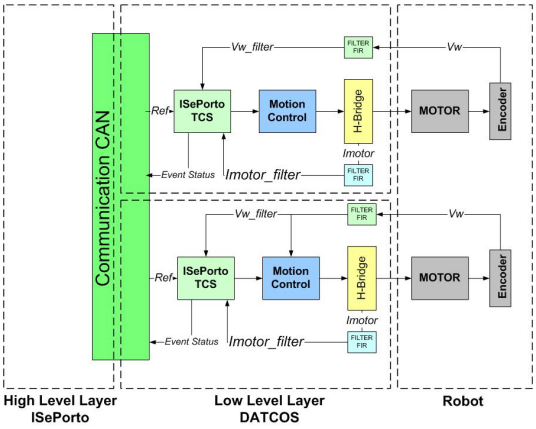


Fig. 2. Traction System Architecture

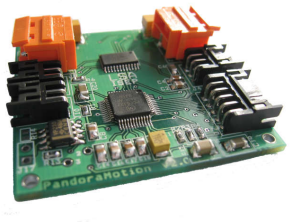


Fig. 3. Axis control node DATCOS

wheels) namely in outdoor terrains where the present observer will improve the navigation quality.

Fig. 4. ISePorto Robotic soccer player

Each driving wheel has a motion control unit performing the slipping estimation, implementing a wheel motion control loop and connected by a CAN bus [1]. In addition, this embedded controller also integrates the power driver for the motor. The main robot CPU controls the global motion issuing references to the control nodes and reading relevant information. The current ISePorto main computer is based on a single-board running a RT-Linux operating system and a navigation and coordination high level software.

V. Experimental Setup

In a previous experimental work[1] the slip occurrence in differential drive DC electrical powered mobile robots was studied allowing the understanding of the phenomenon associated with the traction process and provided insights to the traction control architecture envision and the development of a traction control system. Although the identification and validation of slipping in the robot motion required and external motion measuring source. An experimental setup was implemented combined an external computer vision process to obtain the required information as depicted in figure 5 and the axis control node DATCOS data information.

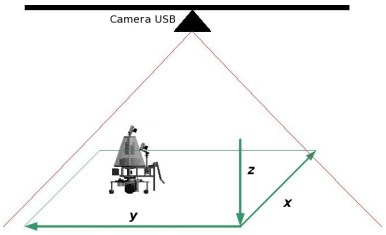


Fig. 5. Experimental setup

An ISePorto robot was used in the tests. The computer vision system provides robot position information thus allowing the determination of velocity and orientation. Simultaneously motor current and encoder odometry information is recorded for each traction motor/wheel. The synchronization between the two CPU's present in the experimental setup was achieved by using the Network Time Protocol (NTP). NTP was a requirement to the data timestamp accuracy and reliability from the experimental scenario.

Robot position and orientation was determined using two circle markers (orange and blue) with a fix distance detected in the overhead image (see figure 8).

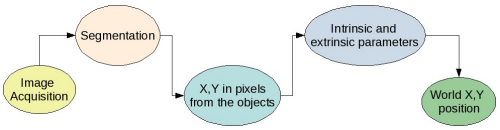


Fig. 6. Image processing

In order to achieve a good robot position accuracy by the vision system, it was necessary to determine the internal camera geometric and optical characteristics (intrinsic parameters), the 3D position and orientation of the camera frame relative to a certain world coordination system (extrinsic parameters) and precise color segmentation. Image processing steps are resumed in figure 6.

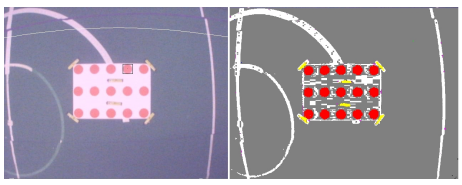


Fig. 7. Know target for accuracy test

A. Physical setup

- USB Philips camera at 15 FPS with 640x480 resolution;
- Blue and orange $15cm$ diameter circular markers;
- Image acquisition and processing in a Intel Pentium Dual Core $2GHz$ and $2GB$ memory;
- RT Linux operating system;

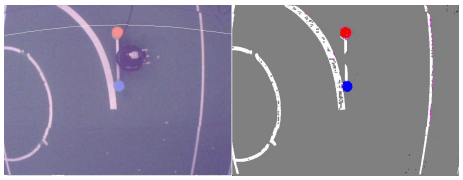


Fig. 8. Circular markers image segmentation

- ISePorto robot with a Intel Pentium Mobile 1.7GHz and 1GB memory;
- CAN bus with a baudrate of 1Mbit;
- Current and odometry acquisition with a sample time of 1kHz;
- Electrical current acquisition was done with a 12bit A/D converter and odometry with a quadrature decoder,giving 5000 ticks per wheel turn;
- Triaxial MicroStrain 3DM-GX1 accelerometer;
- Current and odometry were filtered with a Equiripple FIR filter of order 4. The values applied to the filter were calculated with the Remez Exchange algorithm.
- MATLAB software environment was applied in the logged data posterior analysis.

In order to verify the external vision system image acquisition quality (with particular emphasis on the color segmentation precision), a test target was used. This target had the following characteristics:

- Full 30cm circle markers
- Inter marker distances is 25cm in x and in y 33.5cm
- Target to camera distance 2.78m

By the analysis of the markers identification (figure 7) a medium error of 20mm was found. This error validates the external vision tracking system as a suitable measurement setup for the intended purpose.

VI. Results

A. Slipping

Slipping phenomena is characterized by the drastic decrease of motor current and the velocity increase. In the time interval observed this occurrence demonstrates an adherence loss by the wheel in relation with the supporting surface where the robot moves.

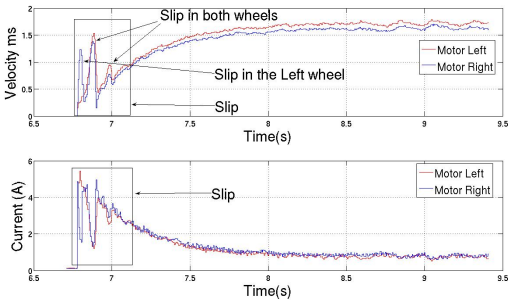


Fig. 9. Slip occurrence in the robot acceleration process

Figure 9 displays the slipping occurrence in both wheels with a reference command step.

B. Embedded control system slipping observer

In the follow section it will be presented the state observer performance implemented in the embedded system DATCOS motion for wheel state detection.

In figure 10 we can observe on the left graphs the velocity and current for each wheel and on the right the observer performance.

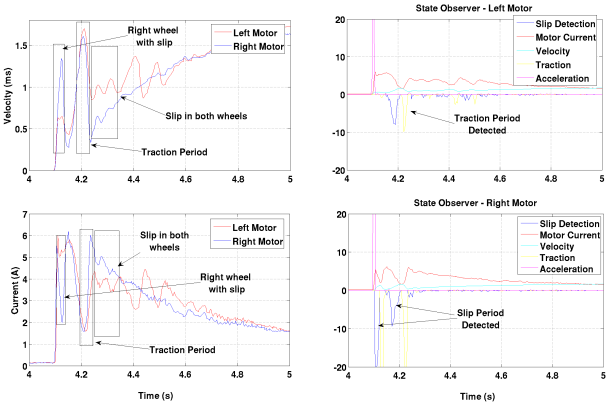


Fig. 10. Slipping observer performance

Analyzing the graphic it is shown the occurrence of slipping in the right wheel, with the corresponding identification by the observer at the instant 4.2s.

Information from the observer implemented in the local motion control board (DATCOS) is transmitted by CAN bus to higher processing (robot motion control level) level with the message *Event Status* (see figure 2).

In figure 11 is possible to observe the behavior embedded algorithm system detection transitions related to figure 10.

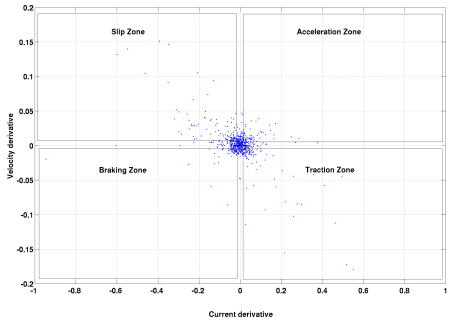


Fig. 11. Region transitions detected by the observer algorithm implemented in the DATCOS embedded system

The four regions represented in the figure 11 are the possible states identified by the implemented algorithm. High dispersion is observed in two regions, traction and slipping.

A movement without perturbations in terms of adherence loss implies a lower dispersion in the presented figure with a highly concentrated set of points around the origin.

The observer performance was measured for a different set of wheels with higher adherence capability. The data is presented in figure 12.

In the figure 12, slipping and traction instants are identifiable by the observer at $4.2/4.4s$ and $4.3s$ respectively.

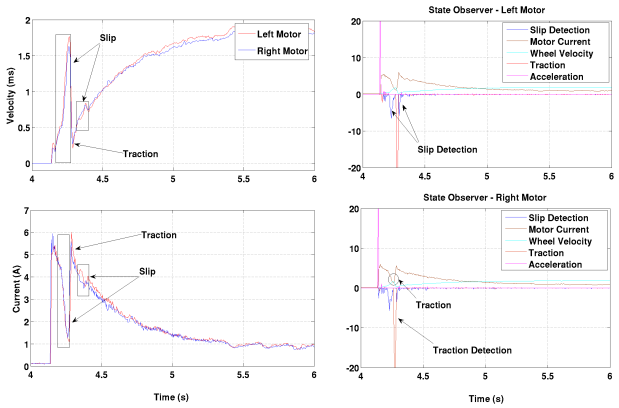


Fig. 12. Observer performance for higher quality (higher friction) wheels

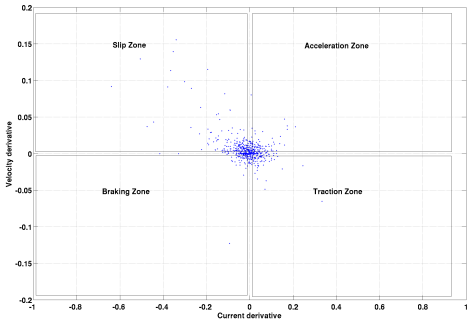


Fig. 13. Region transitions detected by observer algorithm in the embedded system for second wheel set

Transitions between regions detected in figure 13 present a higher dispersion in the slipping region versus the previous figure 11.

Comparing the dispersion with the one presented in figure 11 the higher wheel friction results in a lower dispersion translating in a locomotion increase performance by the robot.

This analysis allows us to determine robot movement "quality" (here defined loosely as without significant slippage) as a low dispersion in the points presented in the previous figure. The non existence of slipping instants (as detected in figure 12) would imply a further reduction of dispersion with a high concentration around the origin.

C. Traction tests

This section presents the results obtained by the external vision system, the triaxial MicroStrain 3DM-GX1 accelerometer and the embedded motion system data.

Comparing vision data with velocity measurements the results are highly coherent, validating the embedded system gathered information. In figure 14 slipping occurs with higher incidence in the right traction wheel with consequent robot motion perturbation at $0.8m$. In the breaking process

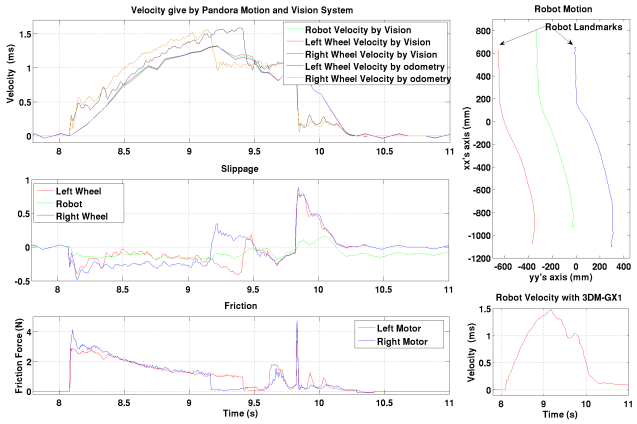


Fig. 14. Robot movement for a 50% PWM reference

friction force and wheel velocity variations are detected. Adherence and slipping instants identified in the figure 14 since $9.8s$ to $10.2s$.

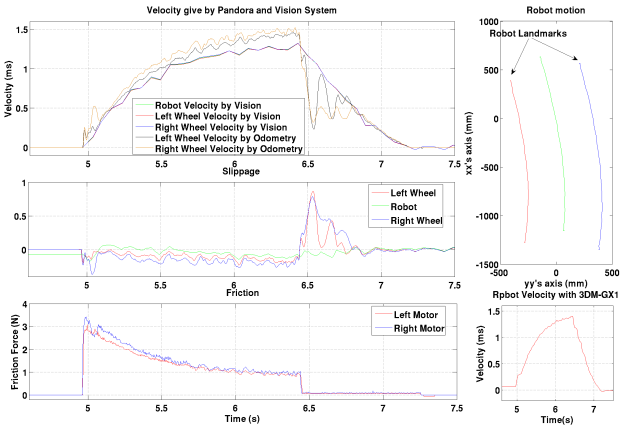


Fig. 15. Robot movement for 100% PWM reference

In figure 15 robot motion does not present slipping in almost all instants with the exception of the breaking process with the identified transitions between adherence and slipping. The external vision system does not allows the identifications of perturbations in each traction wheel, only presenting a global decrease of robot velocity. Through analysis of the three figures embedded system results are confirmed by the external gathered motion data.

D. Qualitative analysis

A qualitative analysis for the slipping observer is presented in this section with the data gathered by the external vision system. Figures 16, 17 and 18 demonstrate traction wheel behavior with information given by the embedded motion control system and external vision tracking.

In the figures plots identified by *Right wheel slip* or *Left wheel slip* present slippage obtained by the external vision system and by the embedded motion. Positive values in the vision system data represent a situation with the robot in an adherence process. Figure 16 shows that even for a low

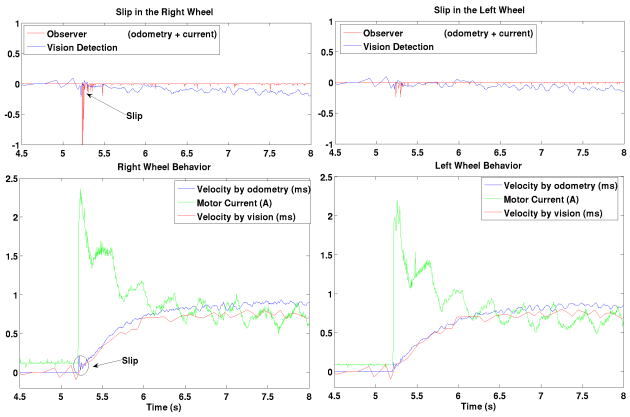


Fig. 16. Slipping obtained by vision comparing with the observer algorithm data for a 25% reference command

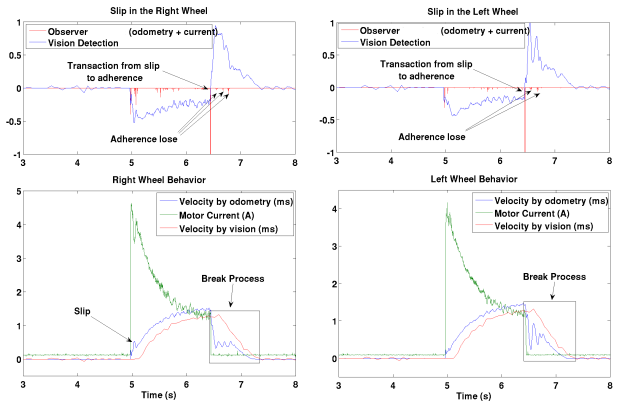


Fig. 18. Slipping obtained by vision comparing with the observer algorithm data for a 100% reference command

PWM reference commands slipping has ocurred with higher prevalence around time instant 5.25s.

This event was promptly identified by the observer algorithm implemented in the embedded system.

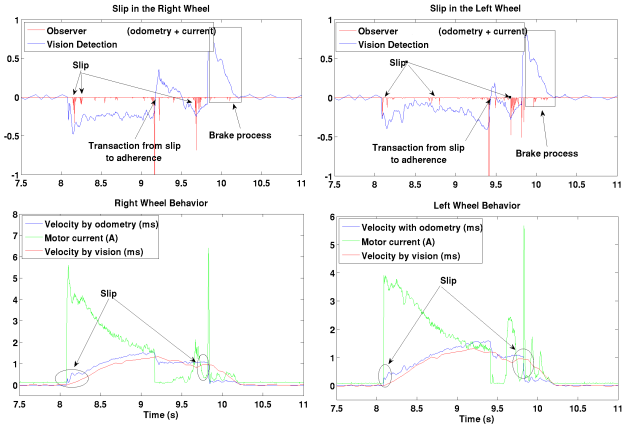


Fig. 17. Slipping obtained by vision comparing with the observer algorithm data for a 50% reference command

Figures 17 and 18 show that the embedded system detects slippage in agreement with slipping instants detected by the external system. In the breaking phase adherence loss was identified by the embedded system in agreement with the external vision. However a performance increase in the detection algorithm would be expected if the measurement of current including signal.

Presently only and absolute value for motor current is measured.

VII. Conclusions

The embedded slipping observer for the mobile wheeled robots was analyzed. An experimental setup based in an external computer vision system was implemented providing and independent source of motion data and allowing a qualitative analysis.

The overall motion control infrastructure, already implemented and tested in Robocup competitions allows local slip reduction (by detecting discrepancies in current and motor velocity and reducing reference) and relays to higher hierarchical levels information necessary to replanning when necessary.

Local slip control is currently implemented and must be validated and tested in operational conditions.

Comparing vision data with the slipping detection the results are highly coherent, validating the embedded system detection moments.

The present system represent the first approach for the traction control system by detecting the wheel slipping occurrence. In the future the system can be further developed by including dynamic robot model to an higher level response.

This work is also to be extended to other robotic motion control applications and in particular to marine robotics. Preventing effects such as cavitation, Wagner's effect and cross-coupling drag, or controlling active thrust and minimize thrust reduction due to propeller axial flow.

VIII. Acknowledgments

The authors would like to thank LSA team for their effort and dedication on robots development, experimental setup and support in competitions and events. This work is sponsored by IPP and ISEP.

References

[1] André Dias José Miguel Almeida Alfredo Martins and Eduardo Silva. Traction characterization in the Robocup Middle Size League *The 7th Conference on Mobile Robots and Competitions* 2007.

[2] Yasushi Toyoda Yoichi Hori and Yoshimasa Tsuruoka. Traction control of electric vehicle: basic experimental results using the test ev uot electric march. *IEEE Transaction on Industrial Applications*, 34(5):1131–1138, 1998.

[3] Albagul A. *Dynamic Modelling and Control of a Wheeled Mobile Robot*. PhD thesis, University of Newcastle upon Tyne, 2001.

[4] Shiller, Z. F.L: Dynamics Motion Planning of Autonomous Vehicles, IEEE Trans. on Robotics and Automation Vol. 7,No.2,1991

[5] R. M DeSantis. Modeling and path-tracking control of a mobile wheeled robot with a differential drive, robotica. *Robotica*, 1995.

[6] W. L. Nelson and I. J. Cox. Local path control for an autonomous vehicle.

[7] Yutaka. Vehicle path specification by a sequence of straight lines. *IEEE Trans. on Robotics and Automation*, 1988.

[8] G. T. Wilfong. Motion planning for an autonomous vehicle. *Proceedings of IEEE Intl. Conf. on Robotics and Automation*, 1988.

robotica 2009
festival nacional de robótica

Parallel Task Execution, Morphology Control and Scalability in a Swarm of Self-Assembling Robots

Anders Lyhne Christensen Rehan O'Grady Marco Dorigo

***Abstract*— We investigate the scalability of a morphologically flexible self-assembling robotic system by measuring task execution performance. We use a scenario consisting of three subtasks — gap crossing, bridge traversal and object pushing. Each subtask can only be solved by a dedicated self-assembled morphology. To successfully complete the scenario, individual robots must autonomously assemble and disassemble to form morphologies appropriate to the subtask at hand. Environmental cues tell the robots when they have encountered a particular task. Parallel execution of tasks is possible when there is a sufficient number of robots. With simulated robots, we perform a series of experiments demonstrating the feasibility and the scalability of our system. We implement our distributed control using the scripting language SWARMORPH-script that has been used in previous studies to form morphologies with up to nine real robots.**

I. Introduction

Self-assembling robotic systems are composed of multiple autonomous agents that can physically connect to each other to form larger composite robotic entities. Two of the key potential benefits of self-assembling robotic systems are morphological flexibility and parallelism. Morphological flexibility is important because any robotic entity must have a morphology that is in some way appropriate to the task it needs to perform. In theory, the ability to form a wide range of different morphologies should allow future self-assembling systems to tackle a wider range of tasks than conventional monolithic robots. Such self-assembling systems may well comprise thousands or even millions of individual agents. In such large systems, parallelism will be the key to efficiency—different self-assembled robotic entities will be able to carry out different tasks at the same time. A well-designed self-assembling system should thus allow for massively parallel task execution.

In this study, we explore a scenario designed to investigate morphological flexibility and large scale parallelism. In our scenario, a series of subtasks must be completed. Each subtask is solvable by a dedicated self-assembled morphology, which is incapable of solving the other subtasks. The robots start at one end of the arena and perform phototaxis towards a light source at the other end of the arena. As they proceed, environmental cues indicate the presence of particular subtasks to be solved. When they encounter a subtask, the robots must assemble into the appropriate morphology for the subtask at hand. Once that subtask is complete, the robots disassemble and continue phototaxis. They are thus ready to assemble into another morphology as soon as they encounter another subtask. The nature of the subtasks allows for a degree of parallel execution.

Anders Lyhne Christensen (anders.christensen@iscte.pt) is with DCTI, Lisbon University Institute, Portugal. Rehan O'Grady (rogrady@ulb.ac.be) and Marco Dorigo (mdorigo@ulb.ac.be) are with IRIDIA, Université Libre de Bruxelles, Belgium.

In previous studies, we pioneered a distributed technique for morphology control in self-assembling systems [5], [17] using both real-robots and a dedicated simulation environment. We developed a scripting language with primitives that would allow robots to self-assemble into particular shapes and to disassemble [6], [16]. However, the sequence of morphologies formed was determined in advance by the experimenter, and the self-assembled entities did not carry out any tasks.

In this study, we extend our previous work to apply particular self-assembled morphologies to specific tasks. The self-assembled morphologies are now formed on demand in response to environmental cues. We demonstrate the feasibility of our enhanced system in a dedicated simulation environment. Using our scenario, we explore the behavior of our system under different configurations. We investigate the negative influence of interference by increasing the number of robots while keeping the size of the arena and the number of tasks constant. We investigate how the system scales by concurrently increasing the size of the arena, the number of robots and the number of tasks. The verisimilitude of the simulation environment was verified in a previous study [17].

The paper is organized as follows: In Sect. II, we discuss related work. In Sect. III, we present the *swarm-bots* robotic platform on which this study is based and describe our simulation environment. In Sect. IV, we present the three different tasks that the robots must accomplish through self-assembly and disassembly in our experiments. In Sect. V, we provide an overview of SWARMORPH-script. In Sect. VI, we present the results of our experiments. We discuss our results and conclude the paper in Sect. VII.

II. Related work

There is a large body of scientific literature on the distributed creation and control of robotic morphologies using inter-connectable components. The two principle approaches are self-reconfigurable systems and self-assembling systems. In self-reconfigurable systems [20], the components tend to be incapable of independent motion. In self-assembling systems [11], the components are themselves independent robots that can autonomously form physical connections with one another. In the latter case, the individual robots can be either externally propelled or self-propelled. Several different hardware architectures and control mechanisms have been proposed respectively for self-reconfigurable robotics [3], [14], [15], [19] and for self-assembling robotics [2], [7], [8], [10], [12].

The advantage of morphological flexibility is that it potentially allows a robotic system to carry out a wider range of tasks. Somewhat surprisingly, little work has

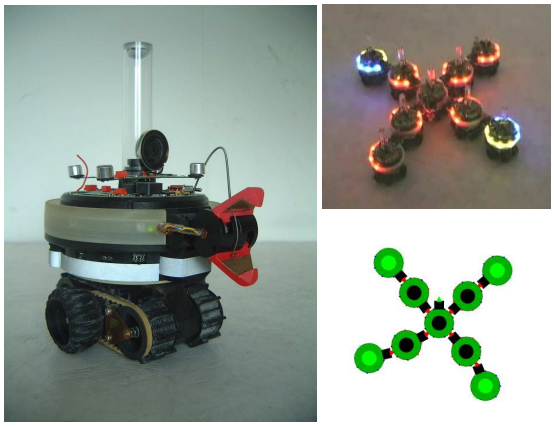


Fig. 1: Left: The *s-bot*. Top right: The star morphology formed with 9 real robots. Bottom right: The star morphology formed in our simulation environment.

directly focused on using self-reconfigurable systems to generate appropriate morphologies in response to task requirements [1] (and almost no work in the field of self-assembling systems). The advantage of self-assembling systems is that, as well as morphological flexibility, they offer the potential for parallel task execution. White *et al.* [18] used mathematical and simulation based models to analyze the scalability of their self-assembling system. However, in common with other works that consider scalable self-assembling systems with larger numbers of robots [13], the focus is on the ability of the system to self-assemble ever larger structures.

In this paper, our approach is different in that the robots form specific morphologies to solve different tasks, and that they carry out the tasks in parallel. We measure scalability, not by an internal measure of self-assembling efficiency, but rather by the external measure of task completion efficiency.

III. Robotics Platform

We conduct our experiments using a simulated version of the *swarm-bots* robotic platform. The platform consists of a number of mobile autonomous robots called *s-bots* (see Fig. 1) that are capable of forming physical connections with each other. Each *s-bot* is equipped with an XScale CPU running at 400 MHz, a number of sensors including an infrared ground sensors, proximity sensors, and light sensors. Physical connections between *s-bots* are established by a gripper-based connection mechanism. Each *s-bot* is surrounded by a semi-transparent ring that can be grasped by other *s-bots*. *S-bots* can advertise their location and/or internal state by means of eight sets of RGB-colored LEDs distributed around the inside of their semi-transparent ring.

The *s-bots* have an omni-directional camera that points upwards at a hemispherical mirror mounted above the *s-bot*'s turret in a transparent perspex tube. The camera records the panoramic images reflected in the mirror. Depending on light conditions, the camera can detect illuminated LEDs on other *s-bots* up to 50 cm away. The combination of the camera and the LEDs thus provides the *s-bots* with local, situated communication capabilities.

The experiments in this study were conducted in a simulation environment consisting of a specialized software simulator with a custom dynamics engine tailored to our robotic platform [4]. All the sensors and actuators that were used are simulated with reasonable accuracy by our simulation environment. We developed a control interface abstraction layer that allowed us to transfer our control programs between the simulator and the real robots without any modification. The control abstraction layer allowed us to run and test the same SWARMORPH-based control programs both in simulation and on real robots.

IV. Tasks and Morphologies

We have chosen three tasks: gap crossing, bridge traversal, and object pushing. None of these tasks can be solved by a single robot operating alone. Instead, the robots have to self-assemble and cooperate in order to accomplish each of the three tasks. Based on trial and error experimentation with real robots, we have designed the three tasks so that each task requires the robots to self-assemble into a dedicated morphology. Each morphology can solve one task and one task only, that is, the dedicated morphology that succeeds in solving one of the tasks will fail to solve if applied to either of the other two tasks. The tasks and their associated morphologies are shown in Fig. 2 and described in detail below.

A. The Gap Crossing Task

In this task, the robots must cross a 22 cm wide rectangular hole that runs the width of the arena. An *s-bot* can detect the gap based on readings from its infrared ground sensors. Of the *s-bot*'s four ground sensors, one points slightly forwards and one points slightly backwards. This allows an *s-bot* to detect a gap before falling into it. A gap of 22 cm was chosen because it is reliably passable by four real *s-bots* connected in linear morphology, while a three *s-bot* linear morphology will fail unless it is perfectly aligned (any smaller morphology always fails).

B. The Bridge Traversal Task

In this task, the robots must use a bridge to cross a 50 cm wide rectangular hole that runs the width of the arena. The bridge is made of two pipes spaced 17.5 cm apart, each with a diameter of 8 cm. The curvature of the pipes is sufficient that a moving *s-bot* cannot balance on a single pipe. The two pipes are also sufficiently far apart that the wheels of a single *s-bot* cannot make contact with both pipes at the same time. Thus, a single *s-bot* cannot traverse a bridge alone. However, a composite robotic entity comprised of two physically connected *s-bots* (appropriately oriented) can traverse a bridge, since it can make contact with both pipes at the same time—each *s-bot* touches one of the pipes. The curvature of the pipes does not cause the constituent *s-bots* of such an entity to topple, as the *s-bots* mutually support each other, see Fig. 2 (middle).

The on-board computer vision software does not enable the robots to estimate the width of a gap or to see the bridge. We have therefore placed a special reflective material before the bridged 50 cm gap to distinguish it from the 22 cm gap. The reflective material can be detected by an *s-bot* using its infrared ground sensors: readings

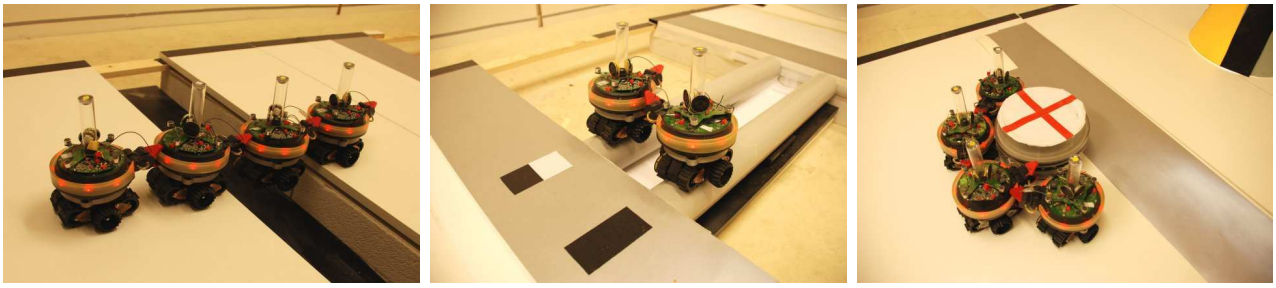


Fig. 2: The three tasks and the appropriate morphology for each tasks. Left: The gap crossing task (line morphology). Middle: The bridge traversal task (support morphology). Right: The object pushing task (shovel morphology).

are higher than for the normal arena floor. In order to determine the position of the bridge, we have put a distinct simple bar code in front of each pipe, see Fig. 2 (middle). The bar code is made up of different materials that can be detected by an *s-bot*'s ground sensors. Whenever a robot detects a bar code, it can use the bar code information to determine which pipe it is facing (left pipe or right pipe) and build the morphology to cross the bridge accordingly. We have also added reflective material on the far side of the bridge to allow the robots to detect when they have successfully crossed the bridge.

C. The Object Pushing Task

In this task, the robots have to perform cooperative transport by pushing two or more objects 30 cm towards the light source. The objects have a dimension and weight that prevents a single *s-bot* from pushing them. In fact, a shovel shape formed by four robots is necessary to reliably shift an object, see Fig. 2 (right). We use objects with a diameter of 20 cm positioned in front of a 30 cm expanse of reflective material. The robots are programmed so that when they have reached the end of the reflective material, they disassemble and move back across the reflective material to search for more objects. The objects are wrapped in the same reflective material. An object that should be shifted can thus be detected by an *s-bot* based on proximity sensor readings—because of the reflective material, the readings for the object are higher than those for either other *s-bots* or for walls.

V. Methodology

We have developed a distributed control scheme that allows *s-bots* to respond to the obstacles described in Sect. IV and to self-assemble into specific morphologies[1]. Each *s-bot* is autonomous and only local, situated, color-based communication is used between the *s-bots*. Whenever an *s-bot* detects the presence of a task that requires a larger robotic entity to be self-assembled, it starts a new self-assembly process by illuminating its LEDs in a particular color configuration. The color configuration indicates a point on the *s-bot*'s body where another non-attached *s-bot* should grip and a corresponding orientation which the gripping *s-bot* should assume. We term such a color configuration a *connection slot* [5].

When an *s-bot* has gripped another *s-bot*, the two *s-bots* initiate communication by changing the color configuration of their LEDs. The communication system allows for the transmission of strings. Through this communication, the newly connected *s-bot* receives instructions on how to extend the local structure. Following these instructions, the newly connected *s-bot* in turn attracts other *s-bots* by opening a new connection slot itself. When a subsequent new *s-bot* attaches, it once again initiates communication, and is told in turn how to extend the structure. As this process repeats itself, the morphology grows accordingly.

A. The SWARMORPH-Script Language

We abstracted basic behaviors such as *phototaxis*, *invite connection*, *send rule ID*, and *disconnect*, into a set of control primitives. We used these control primitives to build a morphology creation language (SWARMORPH-script) that can be executed on real *s-bots* [6]. The language allows for explicit high-level expression of distributed rules for morphology growth. Below, we provide a summary of some of the primitives available in SWARMORPH-script:

- `Phototaxis`: Perform phototaxis until an obstacle has been encountered or overcome.
- `OpenConnSlot`: Invite a connection at a certain location.
- `Connect`: Find and connect to an *s-bot* inviting a connection.
- `SendRuleID`: Send the ID of a rule.
- `ReceiveRuleID`: Receive the ID of a rule.
- `Notify`: Notify a physically connected *s-bot*.
- `Disconnect`: Open the gripper to disconnect from the morphology.
- `Retreat`: Retreat for a certain amount of time.
- `if, then, end`: Branch based on the type of obstacle encountered or based on the rule ID received.

B. The Script

In this section, we describe the script that is used to solve our three task scenario. We describe the overall functioning of the script, and for illustrative purposes present a section of the script, see Script 1. We show the global structure of the script, and focus on the part of the script that builds the

[1]Note that the sensory equipment available on the *s-bot* platform is not sufficiently sophisticated to allow for a truly adaptive morphological response mechanism. Instead, as discussed in Sect. IV, we place cues in the environment that are detectable by the s-bots. The cues uniquely identify the different tasks, and trigger the formation of the appropriate morphology.

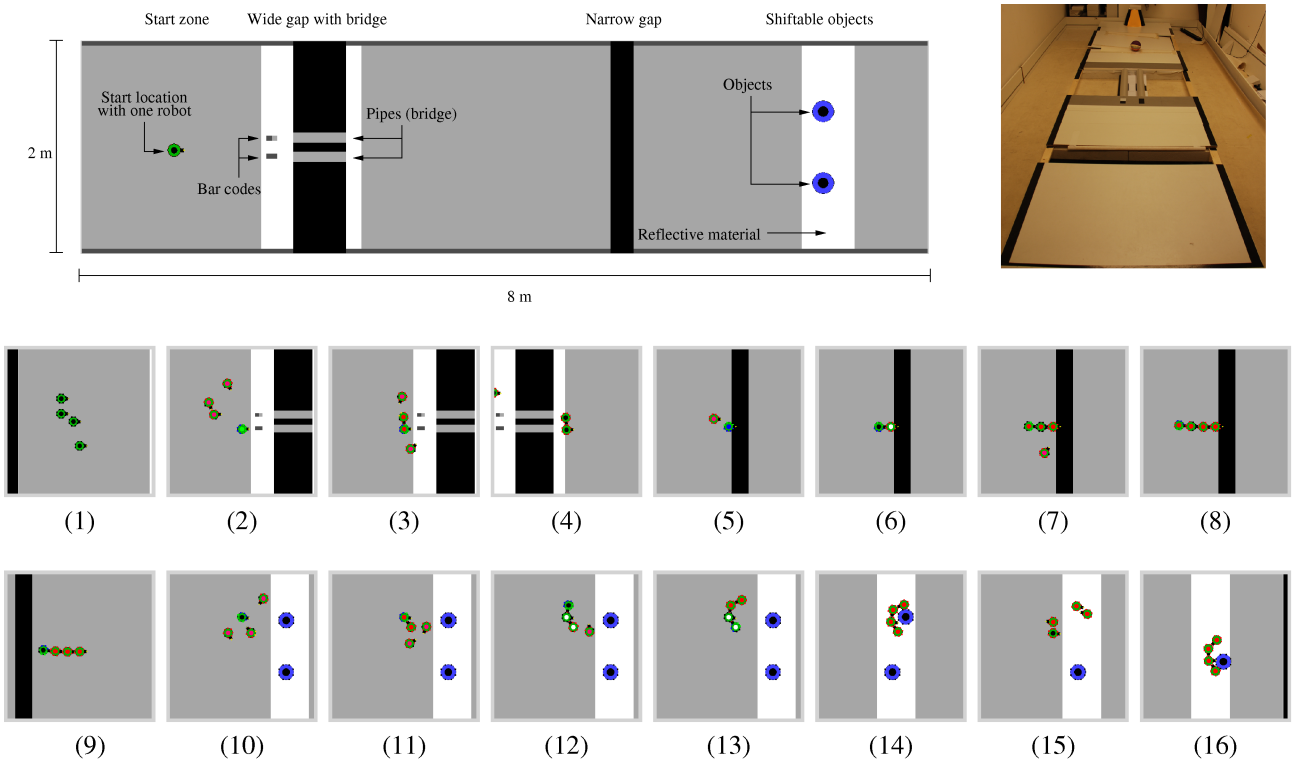


Fig. 3: Top left: A simulated arena (8 m x 2 m). Top right: The real arena (5.0 m x 1.8 m). Bottom rows: Four *s-bots* completing the scenario by first crossing the bridge, then the narrow gap, and finally shifting the two objects. In simulation, the light source is placed on the far right of the arena (not shown). For debugging purposes, the centers of the *s-bots* visually indicate the current controller state of the *s-bots*.

two-*s-bot* morphology necessary to cross the bridge. Due to space considerations we do not show the full script. The full script can be found in the accompanying online material:

`iridia.ulb.ac.be/supp/IridiaSupp2009-002/index.html`

All *s-bots* execute this SWARMORPH-script. Initially, the *s-bots* perform phototaxis individually. When one of the *s-bots* encounters an obstacle, the *s-bot* illuminates its LEDs in order to invite another *s-bot* to connect (it opens a connection slot). When an *s-bot* performing phototaxis sees that another *s-bot* is inviting a connection, it ceases to perform phototaxis and instead tries to physically connect to the inviting *s-bot*. When a successful connection has been formed, communication is initiated.

Fig. 3 shows an example of an arena used in simulation, the real arena, and an example run in which the script presented above is executed on four *s-bots*. A light source is placed on the far right of the arena (not shown), grey indicates normal arena floor, black indicates a hole, while white indicates reflective material. In Fig. 3(1), the four *s-bots* start by performing phototaxis. One of the *s-bots* detects a bar code indicating the presence of a bridge; the *s-bot* retreats off the reflective material and opens a connection slot (Fig. 3(2)). The other *s-bots* detect the invitation to connect and one of them manages to attach (Fig. 3(3)). The *s-bot* that detected the bar code communicates with the newly attached *s-bot* and instructs it to start crossing the bridge. When the two *s-bots* have crossed the bridge (Fig. 3(4)), they disassemble and continue performing phototaxis. One of the two *s-bots* to first cross the bridge then detects the narrow gap and initiates the formation of a line morphology. The other two *s-bots* have also crossed the bridge and contribute to the line morphology (Fig. 3(5-8)). After crossing the gap (Fig. 3(9)), the *s-bots* disassemble and continue moving towards the light source. One of the *s-bots* detects an object (Fig. 3(10)) and starts the formation of a shovel morphology (Fig. 3(11-13)). When assembled into the shovel morphology, the *s-bots* then push the object across the reflective material (Fig. 3(14)), disassemble and retreat back across the reflective material (Fig. 3(15)). The other object is encountered, a morphology is assembled, and the other object is pushed (Fig. 3(16)).

VI. Experiments and Results

In this section, we describe a series of experiments that we carried out in different arenas. We first test our script in a basic task execution experiment, where four robots carry out the three subtasks in sequence. We then test the negative influence of interference in our system, by increasing the number of robots while keeping the arena configuration constant. Finally, we test the scalability of our system by creating a series of progressively larger arenas that allow the tasks to be carried out in parallel, and conduct experiments with correspondingly larger numbers of robots.

A. Basic Task Execution

We conducted 100 experiments with 4 *s-bots* in a 8 m x 2 m arena containing a bridged gap, a narrow gap, and two pushable objects. We used 4 *s-bots*, as this is the minimum number of robots able to complete the scenario

Script 1: Solve three subtasks in an unknown order.

```
Label: "PhototaxisAndLookForTasks"
Phototaxis();
if right-bridge-cue-detected then
    # Retreat off the reflective material
    Retreat();
    # Invite new connection from the left
    OpenConnSlot(left);
    # Send instructions to the connected s-bot
    SendRuleID(1);
    # Cross bridge
    Phototaxis();
    # Restart script
    Jump(PhototaxisAndLookForTasks);
end
else if left-bridge-cue-detected then
    # Retreat off the reflective material
    Retreat();
    # Invite new connection from the left
    OpenConnSlot(right);
    # Rest of the code is identical to code above
    ...
end
else if hole-detected then
    # Start a line morphology
    ...
end
else if object-detected then
    # Start a shovel morphology
    ...
end
else if conn-slot-detected then
    # Connect to a connection slot
    Connect();
    # Receive instructions
    ReceiveRuleID();
    # If a bridge is ahead
    if receivedruleid = 1 then
        # Phototaxis across the bridge
        Phototaxis();
        # Disconnect from the seed
        Disconnect();
        # Restart script
        Jump(PhototaxisAndLookForTasks);
    end
    # Logic for the other morphologies
    ...
end
```

— four *s-bots* are needed both to cross the narrow gap (line morphology) and to push the object (shovel morphology). In each experiment, we recorded the time it took the four *s-bots* to navigate through the arena and to push both of the two objects 30 cm.

At the start of each experiment, the *s-bots* were placed in the starting zone and oriented to face the light source. We let each experiment run for 6,000 simulated seconds (= 100 minutes). The results are summarized in Tab. I. In ninety-four of the experiments, the four *s-bots* succeeded in navigating the arena and pushing both of the objects the required distance.

We witnessed two types of failure that prevented one or both of the objects from being pushed in six of the experiments. Firstly, there are sometimes 'robot casualties' during task execution. We consider a robot to be a casualty if it falls into one of the gaps. When one or more robots fall into a gap before both objects have been pushed the requisite distance, there are then insufficient remaining *s-bots* to complete the scenario. Secondly, if the *s-bots* form

TABLE I: Results summary of experiment with 4 *s-bots* in an 8 m x 2 m arena and two objects to push.

0 objects pushed	1 experiments
1 object was pushed	5 experiments
2 objects were pushed	94 experiments
Average time, 1st object	1,150 s (st.dev 310 s)
Average time, 2nd object	1,803 s (st.dev 332 s)

a slightly misaligned shovel morphology, the object can slide off the side of the shovel before it has been shifted 30 cm. As a result, the object remains in the center of the reflective band and can no longer be detected by the *s-bots*.

In one experiment, neither of the two objects were successfully pushed. This occurred due to a misaligned shovel morphology in both cases. In another five experiments, only one of the two objects was pushed (see Tab. I). One of these experiments failed due to robot casualties, and four of these experiments failed due to misaligned morphology growth.

B. Negative Influence of Interference

Both types of failure that we saw in the previous section are caused by interference (for more details on interference and its potential role in controller design, see [9]). Interference occurs when a high local density of robots results in collisions (although the robots perform obstacle avoidance, this mechanism is overwhelmed when the density is sufficiently high). Collisions lead to robot casualties when one of the colliding robots is pushed into a gap. Collisions lead to misalignment when a robot that is inviting a connection is displaced or rotated by a collision with another *s-bot*.

To determine the influence of interference on task completion performance, we ran an additional set of experiments with a varying number of *s-bots* in the same 8 m x 2 m arena that we used in Subsect. VI-A (see Fig. 3(top left)). In each experiment, the *s-bots* were initially placed in the starting zone and oriented to face the light source.

The results are summarized in Fig. 4. For each experimental setup with a given number of *s-bots*, we performed 100 replications. In each replication, we varied the initial placements and initial seed for the random number generator. The results for each set of experiments are summarized by two bars. The wide bars indicate the average task completion time and standard deviation observed in 100 replications of the experimental setup. The narrow bars denote the percentage of robot casualties.

As the results indicate, the average performance initially increases as more *s-bots* are added. However, at a group size of 18 *s-bots*, the average performance begins to decrease. Furthermore, the percentage of robot casualties (the narrow bars in Fig. 4) increases monotonically with the robot density. In the four *s-bots* experiments, we observed one robot casualty in a single experiment, yielding a robot casualty percentage of 0.25%. When 30 *s-bots* are present in the same arena, the robot casualty percentages is 20.90% ($\approx$ 6 *s-bots*/experiment on average).

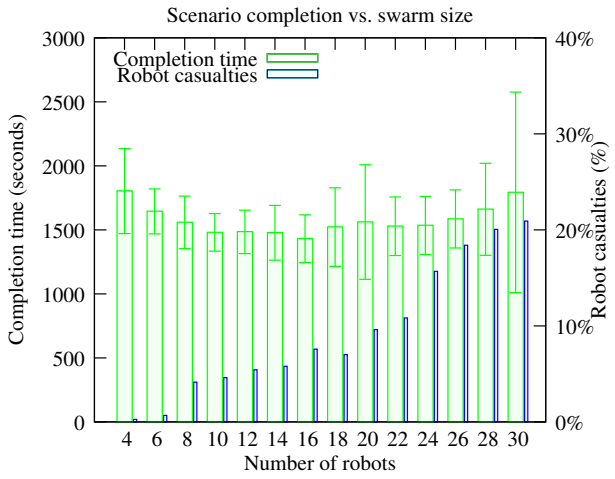


Fig. 4: Scenario completion times and robot casualties for swarms of different sizes in an 8 m x 2 m arena with pushable objects.

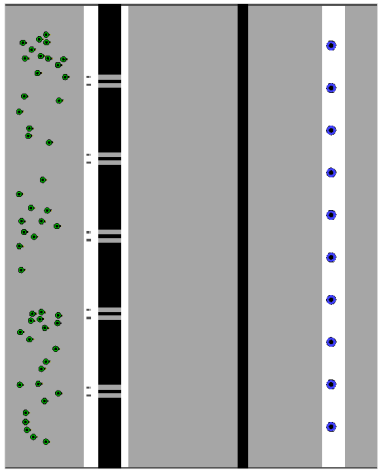


Fig. 5: An example of an arena for scalability experiments with 50 *s-bots*. The width w of the arena is 10m, the number of bridges b is 5, and the number of pushable objects o is 10.

C. Scalability

In order to evaluate the scalability of our approach, we ran a series of experiments with progressively larger numbers of *s-bots* and correspondingly larger arenas. We varied the size of the initial robot population from 100 robots to 1,000 robots in increments of 100. We performed 100 replications for each population size. Each experiment was run for 1,500 simulated seconds (= 25 minutes).

For each population size, we set the width of the arena, the number of bridges and the number of objects as a function of the number of robots in the population. If n is the number of *s-bots* in a given experimental setup, the arena is 8 m long, $w = n/5$ meters wide, contains $b = n/10$ bridges, and $o = n/5$ pushable objects. The bridges and pushable objects are uniformly distributed along two lines running the width of the arena. Note that to obtain the arena that was used in the experiments of the previous two sections, we would need an initial population size of 10 robots ($n = 10$). An example of an arena corresponding to an initial robot population of 50 *s-bots* ($n = 50$) is shown in Fig. 5.

The results are summarized in Fig. 6. Each bar denotes the average number of objects pushed by a swarm of a fixed size over the 100 replications of the experiment. Each bar is annotated with the standard deviation for the result set.

In Fig. 6, we have added a least squares fit line ($y = 0.117 \cdot x$). As the results show, the task execution performance scales linearly with the number of *s-bots*. Linear scalability should not come as a surprise: the control is completely decentralized and each *s-bot* acts based only on what it senses in its immediate vicinity. We therefore expect that the linear scalability trend would continue beyond swarms of 1,000 *s-bots* (however, given the computational resources required, we have been unable to confirm this).

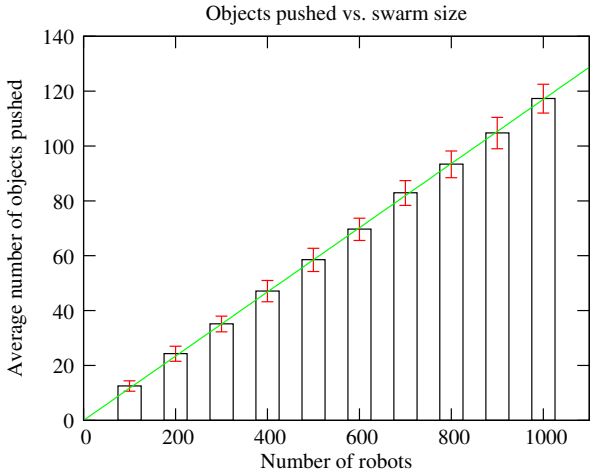


Fig. 6: Scalability

VII. Conclusions

We have presented a scenario in which robots have to solve three different tasks. In order to solve the different tasks, the robots have to cooperate by self-assembling into specific morphologies appropriate to each task.

We conducted extensive simulation-based experiments to investigate issues related to interference between robots. We found that high robot densities resulted in a lower performance and an increase in robot casualties. In another set of experiments, we investigated scalability by increasing the number of robots and tasks in the scenario. We found that the task execution performance scales linearly with the number of robots and number of tasks — at least up to 1,000 robots. Given our decentralized control approach, we expect this trend to continue for even larger swarms.

We are currently conducting experiments on real robotic hardware using the same SWARMORPH-script based control program that we have used in the simulation-based experiments presented in this study. Our ongoing research concerns the cooperation between meta-entities, that is, cooperation between two or more self-assembled robotic entities.

Acknowledgements

This work would not have been possible without the innovative robotic hardware developed by Francesco Mondada's group at the Laboratoire de Systeme Robotiques

(LSRO) of EPFL. This work was supported by the *SWARMANOID* project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-022888 and by the *VIRTUAL SWARMANOID* project funded by the F.R.S.-FNRS. Marco Dorigo acknowledges support from the F.R.S.-FNRS, of which he is a Research Director.

REFERENCES

[1] H. Bojinov, A. Casal, and T. Hogg. Emergent structures in modular self-reconfigurable robots. In *IEEE International Conference on Robotics and Automation (ICRA 2000)*, pages 1734–1741. IEEE Computer Society Press, Los Alamitos, CA, 2000.

[2] H. B. Brown, J. M. V. Weghe, C. A. Bererton, and P. K. Khasla. Millibot trains for enhanced mobility. *IEEE/ASME Transactions on Mechatronics*, 7(4):452–461, 2002.

[3] Z. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *International Journal of Robotics Research*, 23(9):919–937, 2004.

[4] A. L. Christensen. Efficient neuro-evolution of hole-avoidance and phototaxis for a swarm-bot. Technical Report TR/IRIDIA/2005-14, IRIDIA, Université Libre de Bruxelles, Belgium, 2005. DEA Thesis.

[5] A. L. Christensen, R. O'Grady, and M. Dorigo. Morphology control in a multirobot system. *IEEE Robotics & Automation Magazine*, 14(4):18–25, 2007.

[6] A. L. Christensen, R. O'Grady, and M. Dorigo. SWARMORPH-script: A language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence*, 2(2-4):143–165, 2008.

[7] R. Damoto, A. Kawakami, and S. Hirose. Study of super-mechano colony: concept and basic experimental set-up. *Advanced Robotics*, 15(4):391–408, 2001.

[8] T. Fukuda, M. Buss, H. Hosokai, and Y. Kawauchi. Cell structured robotic system CEBOT: Control, planning and communication methods. *Robotics and Autonomous Systems*, 7(2-3):239–248, 1991.

[9] D. Goldberg and M. J. Matarić. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings of the National Conference on Artificial Intelligence*, pages 637–642. John Wiley & Sons, Hoboken, NJ, 1997.

[10] R. Groß, M. Bonani, F. Mondada, and M. Dorigo. Autonomous self-assembly in swarm-bots. *IEEE Transactions on Robotics*, 22(6):1115–1130, 2006.

[11] R. Groß and M. Dorigo. Self-assembly at the macroscopic scale. *Proceedings of the IEEE*, 96(9):1490–1508, 2008.

[12] S. Hirose, T. Shirasu, and E. F. Fukushima. Proposal for cooperative robot "Gunryu" composed of autonomous segments. *Robots and Autonomous Systems*, 17:107–118, 1996.

[13] K. Hosokawa, I. Shimoyama, and H. Miura. Dynamics of self-assembling systems: analogy with chemical kinetics. *Artificial Life*, 1(4):413–427, 1994.

[14] E. Klavins, R. Ghrist, and D. Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 51(6):949–962, 2006.

[15] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-tran: Self-reconfigurable modular robotic system. *IEEE-ASME Transactions on Mechatronics*, 7(4):431–441, 2002.

[16] R. O'Grady, A. L. Christensen, and M. Dorigo. Autonomous reconfiguration in a self-assembling multi-robot system. In *Ant Colony Optimization and Swarm Intelligence, Sixth International Conference, ANTS 2008*, pages 259–266. Springer-Verlag, Berlin, Germany, 2008.

[17] R. O'Grady, A. L. Christensen, and M. Dorigo. SWARMORPH: Multi-robot morphogenesis using directional self-assembly. *IEEE Transactions on Robots*, 2009. In press.

[18] P. J. White, K. Kopanski, and H. Lipson. Stochastic self-reconfigurable cellular robotics. In *Proc. of the 2004 IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 2888–2893. IEEE Computer Society Press, Los Alamitos, CA, 2004.

[19] M. Yim, K. Roufas, D. Duff, Y. Zhang, C. Eldershaw, and S. B. Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003.

[20] M. Yim, W. M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian. Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14(1):43–52, 2007.

This page is left blank intentionally

Maze Solving with Enhanced Map Representation

João Lobato Oliveira[1], João Certo[2], and Luis Paulo Reis[3]

***Abstract*—This paper presents an agent based maze solving problem facing two distinct situations: a known and an unknown map. For the known map experiment different methodologies were implemented, namely an odometry based localization system, target localization through adaptive triangulation, a regressive obstacle distance function for mapping, a quad-tree strategy for map representation, and the A* algorithm for path planning. We argue that the integration of these methods together with augmented representation of obstacles and a flexible map depth representation grants the design of an efficient deliberative agent for a known map experiment. For solving unknown maps a wall-following scheme was implemented with a quasi-reactive agent. A simulation environment called ciber-rato was used to test the different agent solutions through different complexity mazes, to prove the original hypotheses. After evaluating the experiments, with different parameters, the robot's deliberative performance was compared to the reactive agent's, for proof of concept.**

I. Introduction

CYBER-MOUSE (Ciber-Rato) is a modality included in the "Micro-Rato" competition, directed to teams interested in the algorithmic issues and software control of mobile autonomous robots [1]. This modality is supported by a software environment, which simulates both robots and a labyrinth [2]. The Cyber-Mouse has seven sensors but only two, user selectable, are available at any given time. The final purpose is to reach the cheese, identified by a ground sensor and detectable through a direction providing beacon sensor visible trough low walls. Mouse's performance is evaluated through success on reaching the cheese, the time it took and the number of collisions.

The cyber-mouse competition has been used, amongst other applications as a testbed for long-term planning [3], as a scenario for the detection and avoidance of dangerously shaped obstacle [4], or even as a tool for the teaching of Artificial Intelligence and Robotics [5]. In this paper we evaluate the problems of mapping, localization and path planning by building a deliberative agent that can find its way from the starting position to the target without prior knowledge of the maze. Ultimately this architecture's performance is compared to a reactive approach.

Manuscript received February 15, 2009. This work was supported in part by FEUP Faculty of Engineering of the University of Porto under the Intelligent Robotics class of the Doctoral Program in Informatics Engineering, LIACC – Intelligence and Computer Science Laboratory in the Distributed Artificial Intelligence and Robotics (NIAD&R) research group and the ACORD project -Adaptative Coordination of Robotic Teams (PTDC/EIA/70695/2006).

[1] FEUP – Faculty of Engineering of the University of Porto, Portugal; e-mail: ee03123@fe.up.pt

[2] LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal; FEUP – Faculty of Engineering of the University of Porto, Portugal; e-mail: joao.certo@fe.up.pt

[3] LIACC – Artificial Intelligence and Computer Science Lab., University of Porto, Portugal; FEUP – Faculty of Engineering of the University of Porto, Portugal; e-mail: lpreis@fe.up.pt.

The paper structure is as follows. This section introduced the Cyber-Mouse environment. The subsequent section discusses the problems of mapping and self-localization, navigation and path planning together with some related state of the art algorithms leading to the chosen approach. Section 3 presents the two implemented architectures: one based on several deliberative functionalities, and other essentially based on reactive fundamentals. Section 4 contains a description of the testing environments and the respective results. Finally in section 5 we conclude this paper and point to future work.

II. Robotic Mapping and Planning Overview

Mapping is the process of building an internal estimate of the metric map of the environment [6].The mapping problem is generally regarded of most importance in the pursuit of building truly autonomous mobile robots, but still mapping unstructured, dynamic, or large-scale environments remains largely an open research problem.

Planning is the process of deciding which route to take based on and expressed in terms of the current internal representation of the terrain. Typically this process calculates the cost of each motion decision towards the target, based on a given heuristics, and chooses the "cheapest" one.

In this section we present an overview on robotic mapping and planning and we introduce some state of the art algorithms in these fields. Based on this study, in the next section we explain our approach.

A. Mapping and Localization Problem

Robotic mapping addresses the problem of acquiring spatial models of physical environments through mobile robots, which are then used for robot navigation. To acquire a map, robots must possess sensors that enable it to perceive the outside world. Sensors commonly brought to carry out this task include cameras; range finders (using sonar, laser or infrared technology), radars, tactile sensors, compasses, and GPS. However, all these sensors are subject to errors, often referred to as measurement noise, and to strict range limitations.

So, considering these issues several different challenges can arise to robotic mapping: noisy sensor's measurements (with different error probabilities); high dimensionality of the entities that are being mapped; data association problem (problem of determining if sensor measurements taken at different points in time correspond to the same physical object in the world); environments changing over time; and robot exploration.

The motion commands issued during environment exploration carry important information for building maps, since they convey information about the locations at which different sensor measurements were taken. Robot motion is also subject to errors, and the controls alone are therefore insufficient to determine a robot's pose (location

and orientation) relative to its environment. If the robot's pose was known all along, building a map would be quite simple. Conversely, if we already had a map of the environment, there are computationally elegant and efficient algorithms for determining the robot's pose at any point in time. In combination, however, the problem is much harder.

Considering the map representation problem, which has a significant impact on robot control [7], we can account for three main methods: free space maps (road mapping), as spatial graphs, including Voronoi diagrams, and generalised Voronoi diagrams; object maps; and composite maps (cell decomposition) as point grids, area grids and quad trees.

Virtually all state-of-the-art algorithms for robotic mapping in the literature are probabilistic. They all employ probabilistic models of the robot and its environment, and they all rely on probabilistic inference for turning sensor measurements into maps [6].

Our mapping algorithm is based on a deterministic model representing the distance to an obstacle given the obstacle sensor value. The map is represented using quad-tree decomposition since this algorithm grants good performance for this application, with low processing cost. A more detailed explanation of our strategy is given in Section 3.

B. Navigation and Path Planning

In artificial intelligence, planning originally meant a search for a sequence of logical operators or actions that transform an initial world state into a desired goal state [8]. Robot motion planning focuses primarily on the translations and rotations required to navigate, considering dynamic aspects, such as uncertainties, differential constraints, modelling errors, and optimality. Trajectory planning usually refers to the problem of taking the solution from a robot motion planning algorithm and determining how to move along the solution in a way that respects the mechanical limitations of the robot.

The classic path planning problem is then finding a collision-free path from a start configuration to a goal configuration, in a reasonable amount of time, given the map representation, retrieved in the mapping process, and the robot's body constitution.

In an unknown environment the mapping and motion planning must be processed in parallel through exploration and dynamic navigation decisions. This structure requires plans updating. A natural way of updating plans is to first select a path based on the present knowledge, then move along that path for a short time while collecting new information, and re-planning the path based on new findings.

Considering the application many algorithms have been proposed for path planning: A and A Star (A*), Dijkstra, Best-First, Wavefront Expansion, Depth-First Search, Breadth-First Search. Our strategy uses the A* algorithm with a quad-tree representation of the map, as it will be explained in the next section. The decision was made by balancing implementation cost with a guarantee of a solution.

III. Architecture

Our architecture is presented in four independent modules, concerning the self-localization, target (goal) localization, mapping and navigation, and path planning problem. These modules were integrated to solve various mazes facing different conditions: in a known map, with knowledge of the start and target positions; and in an unknown environment without any previous knowledge.

A. Self-Localization

The self-localization is based on the robots' odometry which is defined by a dynamic movement model [9]. Considering the robot diameter of 1 mouse unit (Um), Eq. 1 represents the power of each motor considering the robot's inertia; Eq. 2 models the linear velocity, and Eq. 3 the rotation, which represents the angle with the North. Finally, Eq. 4 and Eq. 5 results in the X and Y axis value of the robot, relative to its starting position:

$$\begin{cases} lOutPow_t = (lOutPow_{t-1} + lInPow_t)/2 \\ rOutPow_t = (rOutPow_{t-1} + rInPow_t)/2 \end{cases} (Um) \quad (1)$$

$$lin_t = \frac{|lOutPow_t + rOutPow_t|}{2} \quad (Um/cycle) \quad (2)$$

$$\begin{cases} rot_t = compass\ direction, & t\ mod\ 50 = 0 \\ rot_t = rot_{t-1} + (rOutPow_t - lOutPow_t), & t\ mod\ 50 \neq 0 \end{cases} (rad) \quad (3)$$

where mod is the remainder operator

$$X_t = X_{t-1} + lin_t * \cos(rot_t) \quad (Um) \quad (4)$$

$$Y_t = Y_{t-1} + lin_t * \text{sen}(rot_t) \quad (Um) \quad (5)$$

As evinced in Eq. 3, the rotation initial value equals the compass direction. Afterwards, in order to correct cumulative odometry rotation errors, the same adjustment is done every 50 cycles, always accounting for the compass sensor latency of 4 cycles.

Due to Gaussian noise, the model presented induces a linear motion maximum error given by Eq. 6.

$$\delta \leq \frac{Max(MotorPow) * NoiseDeviation + MotorResolution/2}{Max(MotorPow)} \ (\%) \quad (6)$$

As such, for each position estimate there is a maximum δ deviation for the Cartesian coordinates and 2*δ for the rotation angle. The simulator defines *Max(MotorPow)=0.15*, *NoiseDeviation=1.5%* and *MotorResolution=0.001*; which infers δ≈1.83% and a rotation error of 3.66%, acceptable for this application.

B. Target Localization

For the beacon localization a two steps method consisting on triangulation and recursive adjustments was implemented.

A first step, depicted in Fig. 1, triangulates the beacon position by intersecting two lines given by two reference points (two different robot positions). The target is visible when the beacon is within the angular sensing range of the mouse and there are no high obstacles between the mouse location and the beacon. In this situation, while exploring the map, a first point (*A*) is traced by memorizing the robot's position, the beacon direction (α) and the current rotation angle (β). Then within a Euclidean distance of 3*um* and an angle difference of at least 30º, to minimize the error, a new point (*B*) is memorized along with the beacon and rotation angle. Given the beacon sensor

latency of 4 time units (*ut*) and its Gaussian noise, in each point the mouse stops for 20*ut* and the beacon final angle is given by the average of the values retrieved from the 5th to the 20th cycle. Given these parameters the target position (X_C, Y_C) is obtained, in Eq.9, by the following deduction:

$$\begin{cases} Y_A = m_A * X_A + b_A \\ Y_B = m_B * X_B + b_B \end{cases} (Um)\ ,\ m = \tan\theta \qquad (7)$$

$$\begin{cases} \theta = \alpha + \beta + 2\pi, & if\ \theta < -\pi \\ \theta = \alpha + \beta - 2\pi, & if\ \theta > \pi \\ \theta = \alpha + \beta, & else \end{cases} (rad) \qquad (8)$$

$$\begin{cases} X_C = (b_B - b_A)/(\tan\theta_A - \tan\theta_B) \\ Y_C = \tan\theta_A * X_C + b_A \end{cases} (Um) \qquad (9)$$

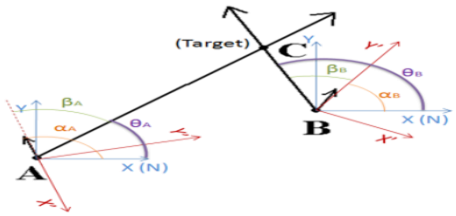


Fig. 1. Triangulation scheme.

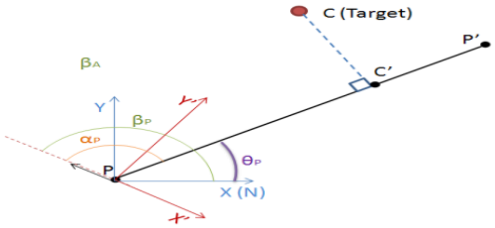


Fig. 2. Target position adjustment.

The second step consists on adjusting the target point with recursive new measures (each 25 cycles). Given a new mouse position (*P*) and a correspondent beacon and rotation angles, a new line (linear equation) is traced and the former beacon estimate is compared with the closest point ($X_{C'}$, $Y_{C'}$) on this new line (see Fig. 2), considering:

$$\begin{cases} Y_{P'} = any \\ b_P = Y_P - \tan(\theta_P) * \ _P \\ X_{P'} = (Y_P - b_P)/\tan(\theta_P) \end{cases} \qquad (10)$$

$$new\_line = P + u(P' - P) \qquad (11)$$

$$u = \frac{(X_C - X_P)(X_{P'} - X_P) + (Y_C - Y_P)(Y_{P'} - Y_P)}{||P' - P||^2} \qquad (12)$$

$$\begin{cases} X_{C'} = X_P + u(X_{P'} - X_P) \\ Y_{C'} = Y_P + u(Y_{P'} - Y_P) \end{cases} \qquad (13)$$

The adjustment is then weighted considering the confidence of the current target estimate position. So depending on the number of previous adjustments, *l*, the new target location ($X_C(t)$, $Y_C(t)$) is given in Eq. 14:

$$\begin{cases} X_C(t) = X_C(t-1) * \left(1 - \frac{1}{n}\right) + X_{C'}(t) * \left(\frac{1}{n}\right) \\ Y_C(t) = Y_C(t-1) * \left(1 - \frac{1}{n}\right) + Y_{C'}(t) * \left(\frac{1}{n}\right) \end{cases}, \qquad (14)$$

where $n = min(3 + l\ ,\ 5)$

C. Mapping

The navigation and consequent mapping is based on the obstacles disposition along the map. To calculate the robot's distance to an obstacle relative to its sensor values, a series of successive experimental measurements were taken.

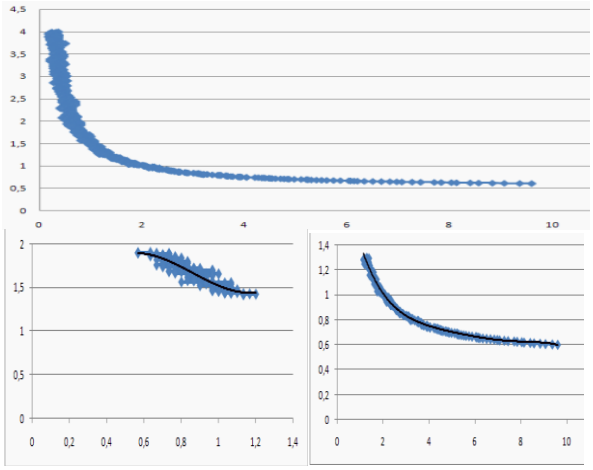


Fig. 3. Obstacle distance distribution: obstacle sensor values in horizontal axis (units); obstacle distance in vertical axis (mouse units). the full distribution ((a) - up); distribution for sensor values ranging from 0.9 to 1.0 ((b) - down-left); distribution for sensor values ranging from 1.1 to 4.5 ((c) - down-right).

In the experiments in Fig. 3 the distance, *d* (vertical axis), is in function of the given sensor values, *x* (horizontal axis). Through linear regression it was possible to obtain the following equation (Eq. 15).

$$\begin{cases} d = 4{,}1981x^3 - 10{,}84x^2 + 8{,}1978x - 0{,}0403, \\ \qquad if\ 0.9 < x < 1.0 \\ d = -0{,}0001x^5 + 0{,}0046x^4 - 0{,}0561x^3 \\ \qquad +0{,}3435x^2 - 1{,}095x + 2{,}2027, \\ \qquad if\ 1.1 < x < 4.5 \end{cases} \qquad (15)$$

These functions estimate the obstacle frontal distance with a low error, δ, to a maximum of 0.213um for distances in the 0.9-1.0 sensor value range, and 0.189 in the 1.1-4.5 range.

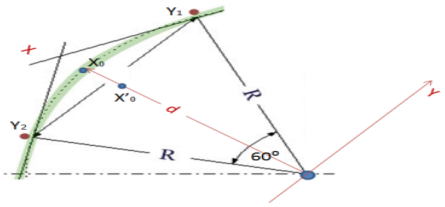


Fig. 4. Obstacle sensor coverage.

Equation 16 is valid for a sensor aperture angle of 60º as depicted in Fig. 4. The total sensor coverage is mapped in Eq.16 and 17.

$$|Y_2| = |Y_1| = X'_0 * sen(30) \qquad (16)$$

$$X'_0 = Xo - Y_1 * sen(30) \qquad (17)$$

C.1. Quad-Tree Map Representation

In the presence of an obstacle we used a Quad-Tree gridding to subdivide each of the obstacle cells. Our quad-tree strategy uses an adaptive division depth to a deepest cell size (granularity) of 0.1*um*. Fig. 5 and Fig. 6 illustrate different granularities for different known maps.

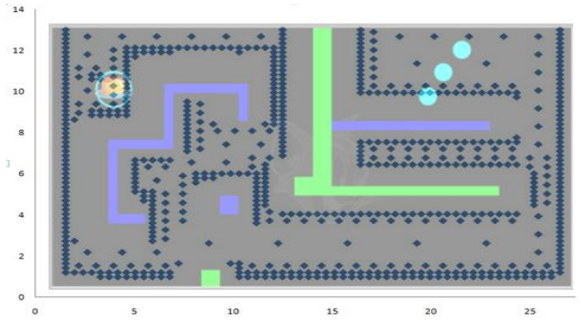


Fig. 5. Real map overlapped with Quad-Tree map representation: RTSS06Final with 0.7*um* depth. Diamonds represent cell centers, high wall are green and low walls are blue.

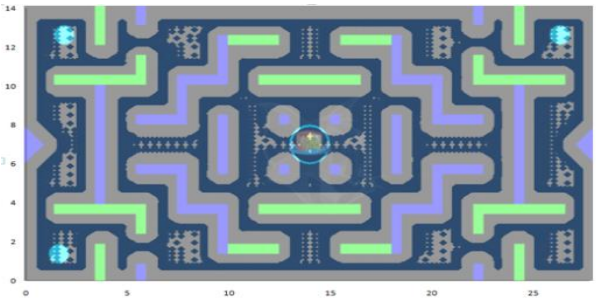


Fig. 6. Real map overlapped with Quad-Tree map representation: 2005Final with 0.1*um* depth.

As the robot has a body with a radius of 0.5 *um*, simply navigating through empty cells represents a problem as, although the center of the robot would pass through an empty cell, its body could collide with an obstacle. In order to coup with this problem the map representation was enhanced with additional cells. Considering the minimum spacing between obstacles of 1.5*um* we implemented a method to grant the robot passage. This method consists on, after dividing each obstacle to the minimum defined cell size, also dividing each adjacent cell (see Fig. 7c)) within a Euclidean distance of the mouse's radius plus 15% for safety. Ultimately, the map outer walls were also subdivided to account for their presence as an obstacle, when calculating the passable cells (see Fig. 7a)).

Quad-Tree implementations often have a pre-defined maximum depth. We argue that this depth should be adapted to each map. As a quad-tree grows deep, the possibility of solving a maze increases along with the improvement of an optimal path. However a deep tree brings two problems: an increasing cost on computing a path and robot control issues regarding the efficiency of navigating through close waypoints. In this implementation, the grid granularity (Quad-Tree depth) is user definable for testing different values in order to optimize the maze-solving performance, verifiable in the experiments (section *4*).

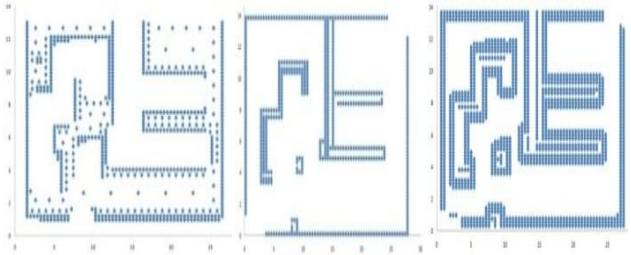


Fig. 7 Quad-Tree cells representation, for RTSS06Final map, with 0.7*um* depth (from left to right): passable cells (a); obstacle cells (b); adjacent cells (c).

C.2. Path Planning

To find a path between a previously computed objective and the agent, the A-Star (A*) algorithm is used. The following function represents the cost between the source point and the target point, which passes through node n:

$$f(n) = g(n) + h(n) \quad (18)$$

Here *g(n)* is the real cost from the source to node *n*, and *h(n)* is the estimated cost between node n and the target. f(n) is the total cost of the path that passes through node *n*. The used heuristic function is the Euclidean distance between the source and the target position. This function is implemented over the Quad-Tree map representation, by defining the shortest path towards the target by marking waypoints in the correspondent map cell centres.

C.3.

The robot is controlled by following each waypoint centre given by the A* algorithm towards the target. The robot rotates to each waypoint centre and accelerates in that direction. The waypoint centre is considered reached if the robot's coordinate values are within a certain error margin of that centre.

The agent navigation speeds are dynamically adjusted and are dependent on several factors. Simple control optimizations include a speed increase in rotation if there is big differences between the current angle and the waypoint direction or increasing spend if a waypoint is far away. More advanced implemented speed optimizations take into account subsequent waypoints and their relative direction in order to further increase the mouse' performance.

IV. Experiments and Results

This section comprises two maze solving experiments: a deliberative agent in a known map, by using a specific map representation and path planning; and a quasi-reactive agent in an unknown map, which reacts in accordance to sensor events.

A. Maze Solving in a Known Map

In this experiment each tested map is parsed from its XML file to retrieve its exact representation along with the robot's starting position. Then the quad-tree method is applied by subdividing each obstacle cell to the minimum required size, delimiting passable areas, as previously showed in Fig. 5, Fig. 6, and Fig. 7, for different maps and granularities (Quad-Tree depths). Using these areas the robot is controlled, as explained in section C3, to follow each waypoint given by the A* algorithm, considering the self-localization method (see III-A) or the GPS for positioning. This procedure is done recursively until reaching the target area.

B. Reactive Agent Architecture

The designed reactive agent consists on a behavior-based state-machine. The main reasoning is done by a state-machine, depicted in Fig.8 in which, when the simulation starts, the robot begins with the state *Find Beacon*, rotating the robot around itself until he finds the beacon or walking randomly until a wall is found. When the beacon is found, the robot changes its state to *Follow Beacon* and goes forward until it reaches the ground beacon area or finds a

wall. If the beacon is reached, the state changes to *Beacon Area Reached* and the simulation ends. On the other hand, if a wall is found instead, the robot changes state to *Change Direction*, rotating itself to the side which has no detectable walls. Once the robot stops detecting a wall directly in front, it changes to the state *Follow Wall*. On this state the robot simply goes forward until it stops detecting the side wall. When it stops detecting the side wall, or detects another wall in front it changes back to the *Find Beacon* state.

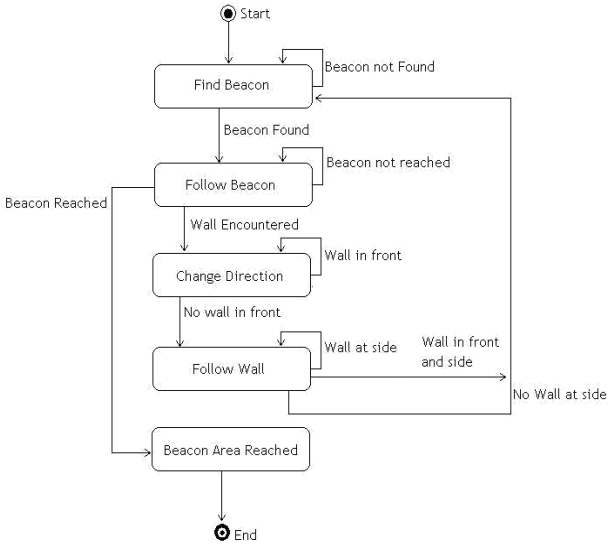


Fig. 8. State Machine for Smart-Follower Agent.

Besides the state, two additional non-reactive elements were included: the time since the mouse was near a wall; and a dual wall-sensor decision model. The notion of time allowed the mouse to wander randomly to a wall when no beacon is found (high-walls) while maintaining a direction for a short time after leaving the wall. Furthermore, this time notion allows to coup with the delay in receiving the beacon's direction, ensuring that no more than a fixed number of cycles are spent in robot's orientation.

The reactive dual sensor wall approach is based on two distance sensors, the frontal IR sensor and one of the lateral IR sensors depending on the relative side of an obstacle. This mechanism allows a closer approach to a wall when the robots' direction is relatively oblique.

The intelligent sensor selector mechanism deals with the limitations of reading two sensors at any given cycle. The mechanism chooses the appropriate sensors depending on the next state and, if more than two sensors are needed, the current cycle time. Depending on sensor's importance, one of the sensors requested can be fixed, and the other (or others) is swapped each cycle within relevant sensors.

C. *Evaluation Scenarios*

In order to evaluate each experiment the following, gradually increasing difficult scenarios, were chosen: *Basic* with a small wall between mouse and beacon (Fig. 9a); *MicRato98*, an easy map with only low walls (Fig. 9b); *2001Final*, a medium difficulty map with only low walls (Fig. 9c); *RTSS06Final*, a hard map with low and high walls (Fig. 9d); *2005Final*, a very hard map with low and high walls (Fig. 9e).

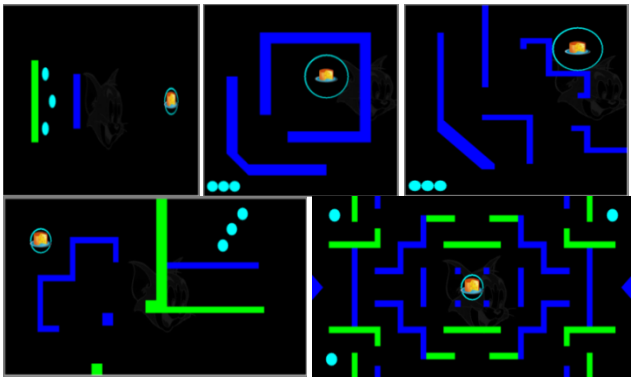


Fig. 9. Evaluation maps (from top-left to bottom-right): Basic (a); MicRato98 (b); 2001Final (c); RTSS06Final (d); 2005Final (e).

The evaluation was done by observing if the mouse reached the cheese or not and the time it took to do it. Since collisions impose errors in the self-localization procedure which would make the robot fail the waypoints (given by A*) towards the target, the number of collisions weren't considered. Additionally, when relevant, an observational description of the mouse' behaviour during the experiment may be included to further evaluate and compare the approaches.

D. *Results*

Each map was tested with the two maze solving experiments. For results comparison, in both experiments the tests were made with two self-localization systems: through odometry measurement (see section III *A*) and with GPS (a debugging sensor providing accurate Cartesian coordinates for robot's position). Different deepest cell's maximum sizes (considered in the quad-tree decomposition) were used, a fixed resolution of 0.1*um* that guarantees map solving (for maximum 1.5*um* obstacle distance) and one variable, granting the best performance for each map. Since the simulator adds some noise in the sensors and actuators, three different runs for each map and agent were performed. As such, conclusions can be made from averaging the results and thus overcoming the stochastic nature of the simulator.

C.1. *Maze Solving in a Known Map*

In this experiment we used the implementation described in 4.1. The achieved results, for each localization method and for the two considered minimum cell sizes, are found in the followings Table I and Table II.

Table I. Maze solving in a known map results for the minimum cell maximum size of 0.1*um*.

Known Map		Experiments						
		Minimum Cell Max. Size	1 Time	2 Time	3 Time	Average Time	Success. Exp.	Observations
Map1 - Basic	GPS	0,1	766	744	796	768	3	NA
	Odometry	0,1	626	626	NA	626	2	Miss Target-Loc. Errors
Map2 - MicRato98	GPS	0,1	1842	1791	1654	1762	3	NA
	Odometry	0,1	1510	1510	1510	1510	3	NA
Map3 - 2001Final	GPS	0,1	1746	1758	1871	1791	3	NA
	Odometry	0,1	1522	NA	1522	1522	2	Collision
Map4 - RTSS06Final	GPS	0,1	3133	3184	3265	3194	3	NA
	Odometry	0,1	2465	NA	2465	2465	2	Collision
Map5 - 2005Final	GPS	0,1	1897	2012	2347	2085	3	NA
	Odometry	0,1	1676	1676	NA	1676	2	Collision

Known Map		Experiments						
		Minimum Cell Max. Size	1 Time	2 Time	3 Time	Average Time	Success. Exp.	Observations
Map1 - Basic	GPS	0,5	738	739	575	684	3	NA
	Odometry	0,5	586	NA	586	586	2	Miss Target-Loc. Errors
Map2 - MicRato98	GPS	0,2	1531	1364	1585	1493	3	NA
	Odometry	0,2	1366	1366	NA	1366	2	Miss Target-Loc. Errors
Map3 - 2001Final	GPS	0,2	1462	1500	1553	1505	3	NA
	Odometry	0,2	NA	1313	1313	1313	2	Collision
Map4 - RTSS06Final	GPS	0,5	2165	2384	2169	2239	3	NA
	Odometry	0,5	1968	1968	NA	1968	2	Miss Target-Loc. Errors
Map5 - 2005Final	GPS	0,2	1860	1860	1860	1860	3	NA
	Odometry	0,2	NA	NA	1532	1532	1	Collision

As observable the unsuccessful tests were provoked by collisions or target missing when using the odometry self-localization method, due to the consequent positioning errors. Each map has a correspondent minimum cell maximum size value for best performance, depending essentially on the spacing between walls and the target area.

C.2. Maze solving in an Unknown Map - Reactive Agent Evaluation

In this experiment we tested our (quasi-)reactive agent, *Smart-Follower*, for paradigm comparison. The results are shown in Table III. The experiment and correspondent results related to our deliberative implementation in unknown environments will be evaluated as future work, as exposed in *4.2* and in section *4*.

Table III. Experimental results for the *Smart-Follower* agent.

Follower	Experiments									
	1		2		3		Average			Observations
	Time	Collisions	Time	Collisions	Time	Collisions	Successful Exp.	Time	Collisions	
Map1 - Basic	1260	0	228	0	1244	0	3	911	0	NA
Map2 - MicRato98	950	0	450	1	886	0	3	762	0	NA
Map3 - 2001Final	638	0	790	0	NA	NA	2	714	0	Closure Conflict
Map4 - RTSS06Final	NA	NA	1366	0	NA	NA	1	1366	0	Wall-Beacon Conflict
Map5 - 2005Final	NA	NA	NA	NA	NA	NA	0	NA	NA	Wall-Beacon Conflict

Here, the (*en*)*closure* conflict happens when the mouse is surrounded by walls on both side sensor and front, and an obstacle on the back. Although the side sensors detect an obstacle there was enough room for the mouse to pass. The *wall-beacon* conflict noted on the observation row happens when the target area is impossible to reach due to entrapment between walls, caused by the conflict of following the beacon and avoiding obstacles at the same time.

V. Conclusions and Future Work

As observed, in the known map experiment the use of the self-localization method granted better results than the GPS, since the exactitude of the GPS positioning makes the robot constantly adjust its position towards each waypoint centre. This effect is also evident through the discrepancy between GPS runs with the same conditions. In contrast, tests using odometry always achieved the same time results, as it discredits errors imposed by the motor's noise and consequently accounts for the motion inaccuracy. Yet this localization error makes the robot collide or miss the target in about one third of the runs, with an exception to the *2005Final* map where the exclusive presence of 1.5um spacing demands great positioning accuracy (with a maximum error of about 1.5%), only granted by GPS. Still the majority of successful tests validate the method.

The deepest cell's maximum size can be adjusted to improve the performance (Table II vs Table I). For each map there is an optimum value which is dependent of the minimum spacing between obstacles and by the target area. This way the obstacle cells must be small enough to grant the robot's passage between walls and to grant a waypoint in the target area centre, and big enough to keep a good performance.

As observable in the second experiment, quasi-reactive approaches, featuring some deliberations, can quite effectively resolve most of the simpler maps (first 3 in Table III) and situations with simple algorithms. When comparing its results with the ones achieved by our deliberative agent, within known maps, we can infer that the reactive implementation simplicity granted better timing performances. Nevertheless the deliberative agent proved to be goal-effective with an accuracy of approximately 75%, only failing the target due to an error-prone odometry, self-localization method. When using GPS the target was always reached, independently on the map. As a final remark one might refer that the *Smart-Follower* senses space and time, by sensing the world with its multidisciplinary sensors, while our deliberative agent is deaf, blind and mute, successfully planning its navigation solely on an internal map representation.

In the future, in known map environments, the proximity sensors can be used in order to avoid obstacles collisions or even to correct odometry errors ensuring a success rate of 100% in odometry based systems. A greedy type approach can also be used for determining the optimal deepest cell's maximum size for solving a known map. Regarding unknown maps, the models here described should be integrated to build a deliberative agent capable of mapping the obstacle dispositions while planning its way to the target.

References

[1] Almeida, L., Fonseca, P., Azevedo, J.L.: The Micro-Rato Contest: a popular approach to improve self-study in electronics and computer science. SMC'2000, IEEE Int. Conference on Systems, Man and Cybernetics, Vol. 1, Nashville, USA (2000) 701 – 705.

[2] Lau, N., Pereira, A., Melo, A., Neves, A., Figueiredo, J.: Ciber-Rato: Um Ambiente de Simulação de Robots Móveis e Autónomos. Revista do DETUA **3** (2002) 647 - 650.

[3] Ribeiro, P.: YAM (Yet Another Mouse) - Um Robot Virtual com Planeamento de Caminho a Longo Prazo. Revista do DETUA **3** (2002) 672-674

[4] Luís, P., Martins, B., Almeida, P., Silva, V.: Detecção de Configurações de Obstáculos Perigosas: Aplicação no Robô EnCuRRalado. Revista do DETUA **3** (2002) 659-661.

[5] Reis, L.P.: Ciber-FEUP - Um Agente para Utilizar o Simulador Ciber-Rato no Ensino da Inteligência Artificial e Robótica Inteligente. Revista do DETUA **3** (2002) 655-658.

[6] Thrun, S. – Robotic Mapping: A Survey CMU-CS-02-111 (2002).

[7] Murphy, R.R., Introduction to AI robotics. Cambridge, MA : MIT Press. (2000) p.466.

[8] LaValle, S.: Planning Algorithms. Cambridge University Press, (2006).

[9] Lau, N., CiberRato 2008 Rules and Technical Specifications, online at: http://microrato.ua.pt/main/Docs/RegrasMicroRato2008_EN.pdf accessed 15 December 2008.

Real-time path planning using a modified A* algorithm

Pedro Costa, A. Paulo Moreira, Paulo Costa

***Abstract*— Real-Time path planning is a key issue for the performance of a mobile robot. In this paper, a modified A* algorithm that can plan in real-time the best path is presented. The suggested modifications to the A* algorithm enable it to deal with non static obstacles in an efficient way. It is shown that the proposed algorithm produces better results when used with moving obstacles.**

I. Introduction

Trajectory planning [1] has always been a problem through the times in mobile robotics. This problem can be classified in two cases: when we have a static environment or a dynamic one. The case where the environment is dynamic has an additional difficulty if we can't know the future position of the moving objects.

For a static environment the full trajectory can be planed in advance. For the case where the environment is dynamic and there is some uncertainty on the future position and velocity of some of the obstacles, the trajectory must be re-planed as new information is gained.

The Small Size League of the Robocup Federation (SSL) is an excellent tested for this problem. When two teams of five robots each compete in a robotic soccer game there is a very dynamic environment. While the position, at each instant, for all the robots can be known, the future position for the robots from the opposing team is uncertain. This means that a lot of the obstacles are moving and their future position can not be pre-computed.

In this case we have robots that can achieve speeds above 2 m/s. So, the path planning algorithm has also some very hard real-time constraints.

There are many possible approaches to this problem. Amongst the most popular are the potential field methods [2] [3] [4] [5], where the robot behaves like a particle immersed in a potential field. The target point acts as an attractive force while the obstacles act as repulsive forces. The combination of this influence should lead the robot to the target while avoiding the obstacles. The biggest problem with this approach is that in a cluttered environment it can result in a impossible or time consuming solution.

There are also the probabilistic approximations like the Rapidly-exploring random tress (RRT) [6] where the roadmap is randomly explored. An improved version ERRT [7] tries to achieve a better performance.

Grid based methods, can, with the movement restricted to the grid slots, find an optimal solution. A few of those algorithms: Dijkstra, A * [8] [9] [10], Wavefront , can find the solution quite efficiently but can only deal with a static environment. There is a optimized variant of the A*, the D * [11] [12] that allows to recalculate less than the entire path in response to discovery of new information.

Here, a different approach is attempted by incorporating some of the dynamics in the way that the obstacles are represented in the cells. The standard path optimization is done following the A* algorithm but the cell representation is modified to incorporate some knowledge about the dynamics associated with the moving obstacles.

First, the standard implementation of the A* algorithm is presented, and then the improved obstacle cell representation is proposed. Finally, the results that show the improved performance and generated trajectory are shown.

Trajectory planning

A*

The A* algorithm works with a cell based map. (fig 1)

For the SSL Robotic field, if the cell size is set to 4cm, as the field dimensions are 4.9 m by 3.9, the grid will have 123 by 98 cells.

Each cell represents a node. Each node can be connected to other nodes and moving from one node to the other has an associated cost (fig 2). In this case, the cost is the metric distance between the cell centers. The A* can calculate the path that minimizes the cost from moving from the starting cell to the target cell.

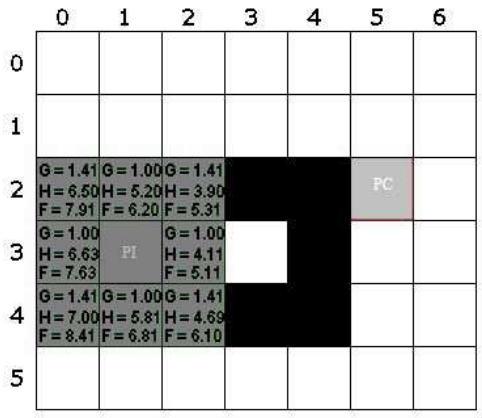


Figure 1 – Cell partitioned environment

Pedro Costa is a PhD Candidate, Paulo Costa and António Paulo Moreira are Assistant Professors with the Department of Electrical and Computer Engineering, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias s/n, 4200 465 Porto, Portugal pedrogc@fe.up.pt

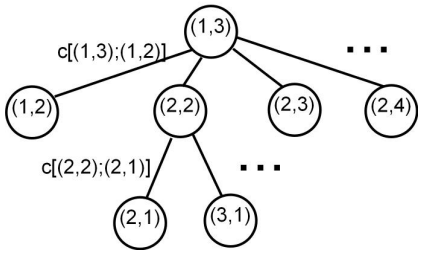


Figure 2 – Associated cost for moving between connected nodes.

The A* Algorithm

There are two lists:

The open list, known as the O-list, that contains the nodes that are candidates for exploration.

The closed list, known as the C-list, that contains the already explored nodes. This nodes where previously in the O-list but as they where explored, they were moved for the C-list.

The nodes in these lists store the "parent" node, which is the node that was used to optimally reach them. This is the node that lies in the shortest path from the origin to current node .

Star(n) – represents the set of neighbors to node n

C(n1,n2) – Cost from going from node n1 to node n2

F(n)=g(n)+h(n) estimate for the lowest cost of going from the origin to the target while passing through node n

g(n) – Cost from the origin to node n

h(n) – An heuristic to estimate the cost of the path from node n to the target node

Algorithm

1. Add origin node to O
2. Repeat
3. Choose $nbest_r$ (best node) from O so that $f(n_{best}) <= f(n)$ $\forall n \in O$
4. Remove nbest from O and add it to C
5. if nbest = taget node then end
6. For all $x \in Q(n_{best})$ which are not in C do:

6.1. if $x \notin O$ then

6.1.1. Adiciona o nó x a O

6.2. else if g(nbest) + c(nbest, x) < g(x) then

6.2.1. Change parent of noce x to n_{best}

7. until O is empty

The basic idea is to choose the best node (lowest cost function) from the O-list. That node is then moved to the C-list and all of its neighboring nodes are processed and inserted in the O-list if they aren't already there. If they are already there, the cost associated with the path from the origin to them is compared with the new one, if the new one is lower, the parent is changed.

This procedure is repeated until the target node is reached or the O-list becomes empty which means that there isn't a feasible solution.

The new cell map construction

The cell map must reflect the possible obstacles affecting the trajectory that the robot must perform. The other robots' velocities can be used to estimate possible collision points.

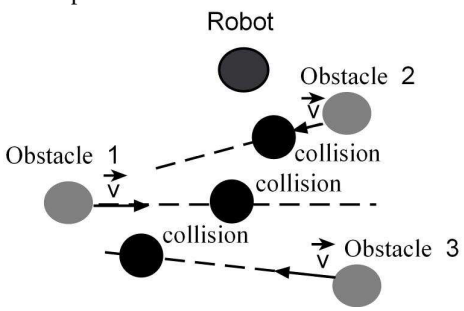


Figure 3 – Collision Points

In this case, while the trajectory must be fully planned, only the first steps are taken before new information arrives and a new calculation is performed. For the SSL team the interval between measures of the robots position is 40 ms. That is also the period of the control loop.
For each calculation the speed of each robot is assumed to be constant. Under that assumption the possible collision point between the robot and an adversary can be estimated. That is where the obstacle will be placed, as it is shown in fig 3

To try to approximate the inherent environment dynamic in a static map there were some techniques that are proposed. They are called:

- Distance
- Slack
- Trail
- Direction

Distance

This change makes the obstacle smaller as the possible collision point is further away from the robot. As the distance increases the relative importance of that obstacle vanishes as is can be seen in figs 4,5.

A distant obstacle mostly does not affect the immediate trajectory points. That can speed up the A* calculation because fewer obstacles will lead to less visited cells and a lower time to find a solution.

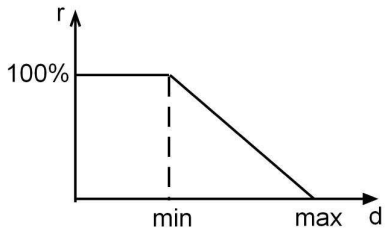


Figure 4: Obstacle size versus distance

In Fig 4 d is the distance between the robot and the estimated collision point, r is the radius of the obstacle, min is the distance above which the obstacle starts losing importance and max is the distance where the obstacle can be discarded.

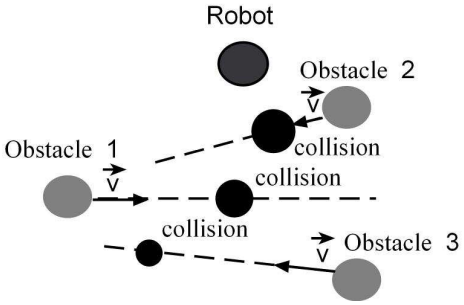


Figure 5: Collision points

Slack

This changes the way an obstacle is represented in the cells. A security area is created around the obstacle. This area is built by setting the cost for those cells above the free ones but still allowing the robot to choose a path through those cells, if the algorithm finds it optimal. This does not make the obstacle bigger but creates a security zone that should be avoided if that does not impact the optimal path. Of course, it can be optimal to use that zone instead of choosing a longer path.

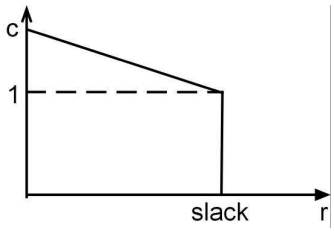


Figure 6 – Slack Zone Cost

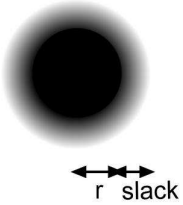


Figure 7 – Obstacle with a slack zone. The black intensity means a higher cost

Trail

A moving obstacle can obstruct the robot for a longer period if the trajectory to avoid the obstacle ends moving the robot parallel to the obstacle movement.

This change creates a certain dynamic awareness to an otherwise static map. It creates an additional zone where the cost to travel there is increased. This zone is created around the projected future positions for the obstacle. The size of this zone depends on the speed of the obstacle. As it is shown in fig 8.

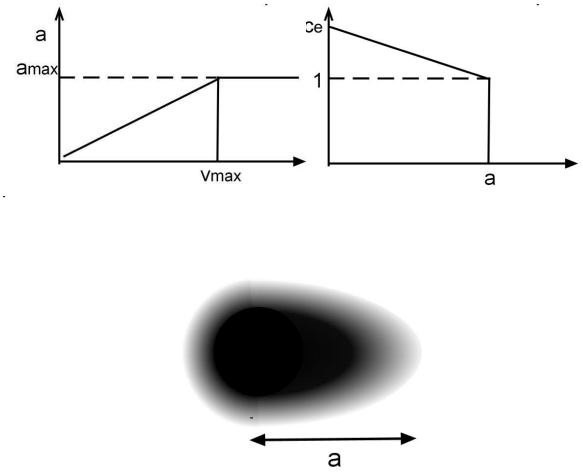


Figure 8 – a) Determination of the size b) Trail Cost
c) Obstacle shape change due to its motion

Direction

This change tries to set the required direction used by the robot as it approaches the target. Without it the robot will hit the target destination from any direction.

There are cases when the approach direction is mandatory. For this case a restriction like in fig 9 is used.

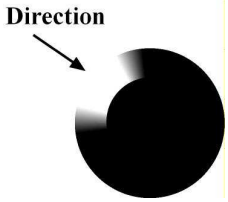
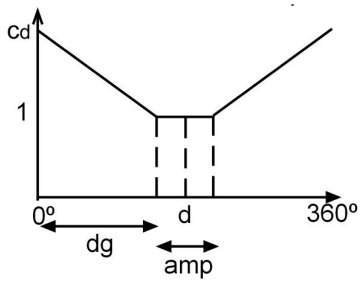


Figure 9 – Target point with mandatory approach direction

Sometimes there is preferred direction but that restriction is not hard. It can violated if the gain in the arrival time is significant. To achieve this, a softer version of the extra obstacle is used, as it can be shown in fig 10



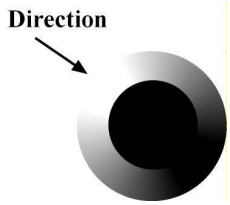


Figure 10 – Target point without hard restriction on the target point

Results

A series of simulations were performed to optimize the parameters for this modification.

A cost function that weights the different performance targets is shown

$$F(x)=0.7*a*T + 0.2*b* P + 0.1*c*C$$

Em que:

T – Time to reach the target.
P - Processing time for the A*.
C – number of colisions
a, b e c – Normalizing factors for different setups

The optimal solution has the robot reaching the target in the shortest time while the algorithm completes in the shortest time also and keeping the collisions as low as possible. The different weighting represents the compromises that are necessary to make. Of course, having the robot reach the target in the shortest time is the most important issue. The processing time must be kept low because that means an extra delay in the control loop and the overall control stability can be compromised. Keeping the collision count low is also desirable.

TABLE I
OPTIMIZED PARAMETERS

Distance	
max	1.25 m
min	0.75 m
Slack	
slack	0.15 m
c	5
Trail	
Ce	5
a	0.65 m
Direction	
amp	60
Cd	5

Two examples were created to test the gains that the new algorithm can yield.

For the first case (fig 11),we have a robot and a obstacle that crosses its path.

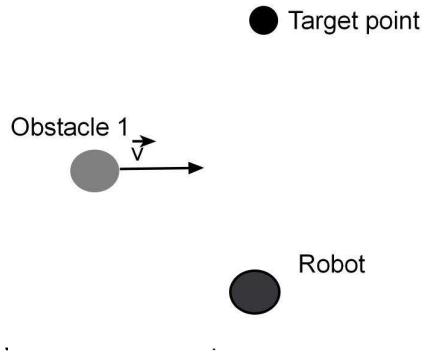
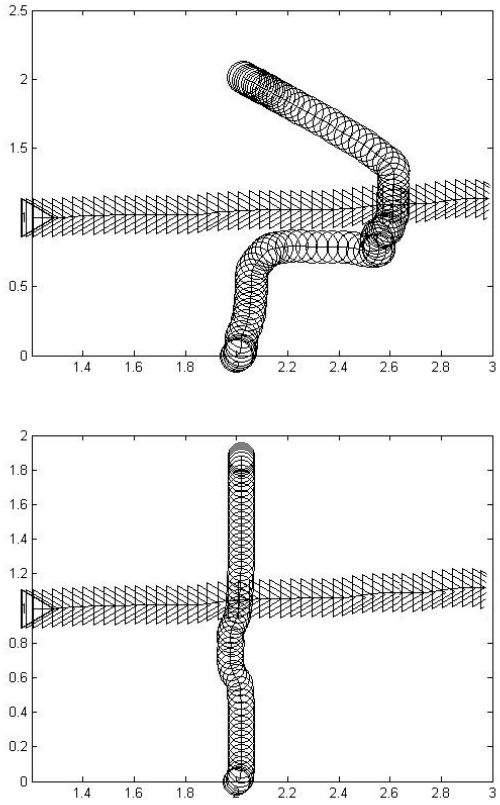


Figure 11 – An obstacle that crosses the robot's path

What happens here is that, without using the obstacle speed, as the robot tries to avoid the obstacle it is dragged in the directions of its movement this happens because the solution where the robot goes around the obstacle choosing to pass in front of it, is the one that seems optimal.



Δ Obstacle
○ Our Robot

Figure 12 – a) Standard b) modified A* results for the case presented in fig 11

TABLE II
RESULTS FOR SITUATION PRESENTED IN FIG. 11

	Standard A*	Modified A*
Time to reach target	2.53s	1.92s
Processing time	0.46ms	0.55ms
Collisions	0	0

In another case that could be taken from a robotic soccer game there are several robots traveling in the field.

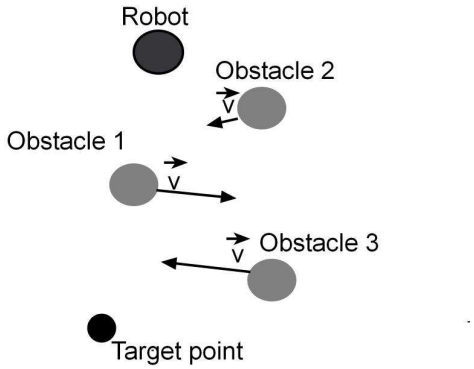
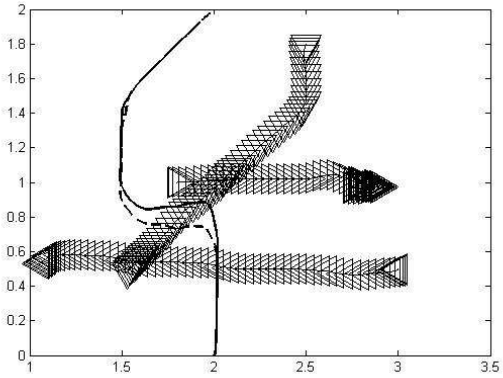


Figure 13 – Several obstacles crossing the robot's path.

For this case the optimal path isn't affected by obstacles 1 and 3, only obstacle 2 will matter.



Δ Obstacles
—— Standard A*
----- Modified A*

Figure 14– Standard and modified A* results for the case presented in fig 13

TABLE III
RESULTS FOR THE CASE PRESENTED IN FIG. 13

	Standard A*	Modified A*
Time to reach target	2.88s	2.76s
Processing time	0.72ms	0.71ms
Collisions	0	0

Comments

In both cases the modified algorithm found a solution where the robot reached the target in less time. The main reason was the drag effect was avoided. In the first case, we were able to avoid the drag of the robot, with the introduction of the trail effect the path chosen no longer to try to go ahead but to pass behind. In the second case, as the obstacle comes into the robot, the path chosen is the one that makes the robot turn up earlier with a smoother path. This is the main gain from using the modified version. Naturally, these solutions are more efficient

The processing time increased in one case and decrease in the other. Three factor work here: the first is to create the modified map increases processing time; second is that the trail effect creates larger obstacles and could increase the number of cells to expand and thus increase the total processing time. The other factor is the shrinkage of obstacles considered far away, that will reduce the processing time.

Both algorithms achieved a trajectory without any collisions.

While this improvements where presented in a robotic soccer setting the advantages that the modified algorithm shows can be found in other cases not specific to robotic soccer competitions. There are many situations where the obstacles are known but their future movement can only be predicted.

The direction restriction was inspired in a typical robotic soccer problem and its contribution is not related with the execution time of the algorithm.

A future improvement should be a way to reflect some dynamical restrictions that the robots have in the obstacles shape to achieve trajectories better suited to the high trajectory speeds.

REFERENCES

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kandor, W. Burgard, L. Kavraki, and S. Thurn , " Principles of robot motion theory, algorithms, and implementation " 2005.

[2] J. Borenstein and Y. Koren, " Real-time obstacle avoidance for fast mobile robots," IEEE Transactions On Systems, Man, And Cybernetics, Vol. 19, No. 5, September/ October, pp. 1179-1187 1989.

[3] Hwang, Yong K., "A potential field approach to path planning" IEEE Transactions On Robotics And Automation, Vol. 8, No. 1, February, Pp. 23-32, 1992

[4] E. H. Miller, "Exact Robot Navigation Using Artificial Potencial Functions," *IEEE Trans. On Robotics and Automation, Vol 8, No. 5, October, pp. 501-518, 1992.*

[5] S.S. Ge, Y. J Cui, "Dynamic Motion Planning for Mobile Robots Using Potential Field Method" *Autonomous Robots 13, pp. 2007-222, 2002*

[6] S. M . LaValle " Rapidly-exploring randon trees: A new tool for path planning" , In Technical Report Nº 98-11, October 1998.

[7] James Bruce and Manuela Veloso. Real-Time Randomized Path Planning for Robot Navigation. In *Proceedings of IROS-2002*, Switzerland, October 2002.

[8] T. Goto, T. Kosaka, and H. Noborio, "On the Heuristics of *A** or *A* Algorithm in ITS and Robot Path-Planning" *IEEE Trans.* on Intelligent Robots and Systems, October, pp. 1159-1166, 2003

[9] P. Melchior, B. Orsoni, O. Lavialle, A. Poty, A. Oustaloup, "Consideration of obstacle danger level in path planning using A$*$ and Fast-Marching optimisation: comparative study," Signal Processing 83 pp. 2387 – 2396, 2003.

[10] K. Yamaguchi and S. Masuda, "An *A*$*$ Algorithm with a New Heuristic Distance Function for the 2-Terminal Shortest Path Problem," *IEEE Trans.* Fundamentals, Vol.E89–A, No.2 February, pp. 544-550, 2006

[11] Karen I. Trovato and Leo Dorst, "Differential A*," IEEE Transactions On Knowledge And Data Engineering, Vol. 14, No. 6, November/December, pp. 1218-1229, 2002

[12] D. Cagigas, "Hierarchical *D** algorithm with materialization of costs for robot path planning", Robotics and Autonomous Systems 52, pp. 190–208, 2005

A competitive dynamic model for decision making in autonomous robots performing cooperative tasks

Flora Ferreira, Estela Bicho and Wolfram Erlhagen

***Abstract*— Efficient team performance in a joint action task requires that each agent takes into account the behaviour of the other teammates when making a decision. In this paper we report results of our ongoing work on endowing autonomous robots with the cognitive capacities that allow them to coordinate their decisions and actions in space and time, and without explicit communication. As a specific example we have chosen the task in which two robots jointly assemble a toy vehicle from its components without direct communication. In order to model the decision processes of each agent we propose a competitive dynamical system. The focus of the paper is on the mathematical analysis of this dynamic model and on its validation in computational simulation for the joint construction task.**

I. INTRODUCTION

A recent trend in robotics is to build autonomous robots capable of interacting cooperatively in a useful and intelligent way with other agents, humans or robots. Successful joint action requires some cognitive capacities. Most importantly, the robot has to be able to predict the actions of others. Taking into account the inferred goals of the teammates in a cooperative task, the robot should decide about the most adequate complementary behaviour. To illustrate the coordination of actions and decisions we have chosen a joint construction task in which two robots assemble a toy vehicle from its components. The vehicle consists of a round platform with an axle on which two wheels have to be mounted and fixed with a bolt (see Fig. 1).

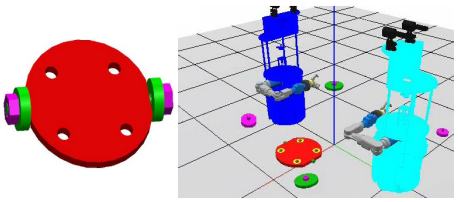


Fig. 1. Toy vehicle to be build (left side) and an example of a possible scenario for the joint construction task (right side).

The pieces are distributed among the workspaces of the robots and it is assumed that both robots known the construction plan. Although the number of individual decisions to be made by each agent is limited, the joint decision process is complex since the desired end state is not defined by a single logical sequence of construction steps. The complexity of the joint task appears to be further increased due to the fact that the working areas of the two agents are separated. This obliges each agent to hand over components like wheels or bolts to the partner or request them. Thus this construction scenario is rich enough to show intelligent decision making in a social context.

In the field of classical artificial intelligence, a standard mechanism for representing assembly/construction plans is AND/OR graphs [1]. A graph allows to access the current steps in the plan and to update the state of the world following agents'actions. There have been various attempts to design robots able to perform construction tasks in collaboration with peers and/or humans. The robot(s) may be controlled, for instance, by a negotiating multi-agent system (see e.g. [2]) or a collaborative problem-solving model of dialogue [3]. In either case, the success of the task strongly relies on explicit exchange of information (communicated information in case of robot-robot(s) or verbal communication in case of human-robot), between the agents is required. An additional drawback is that there is no underlying model for action dynamics representation that can evolve in a goal-directed way. To overcome these drawbacks, there has been recently, a growing interest in the neuroscience based approach, which is based on neuropsychology and neuro-modeling [4]. The aims is to discover the neuro-cognitive mechanisms involved when humans are engaged in joint action and try to implement these in robot(s) performing collaborative work with other agents.

In [5] a cognitive architecture for the joint construction task has been presented, that takes into account several neuro-cognitive mechanisms that are believed to underlie successful interaction in social contexts. The architecture implements the joint coordination of actions and goals as a dynamic process that integrates contextual cues, shared task knowledge and the predicted intention of the partner. In previous work, the architecture has been formalized by a coupled system of dynamic neural fields (DNFs), each implementing a specific functionality. The work reported here is based on the same cognitive architecture, but we explore the extent to which we may simplify the architecture by modelling each layer as coupled system of non linear ordinary differential equations, not by DNFs. The motivation of this simplification is to decrease the computational load for the robotic systems.

The rest of the paper is organized as follows: in section II we give an overview about the cognitive control architecture. The description of the proposed competitive dynamic system are covered in section III. The analysis of the fixed point, their stability and parameters setting are described in section IV. The results of the simulation are presented

This work was supported by the COOPDYN project financed by the FCT and the FEDER (POSI/SRI/38051/2001), by the EU-project JAST financed by the EC (IST-2-003747-IP), and by the FCT fellowship SFRH/BD/42375/2007.

Flora Ferreira is with Dept of Industrial Electronics, University of Minho, Guimarães, Portugal fjferreira@dei.uminho.pt

Estela Bicho is with Dept of Industrial Electronics, University of Minho, Guimarães, Portugal estela.bicho@dei.uminho.pt

Wolfram Erlhagen is with Dept of Mathematics for Sciences and Technology, University of Minho, Guimarães, Portugal wolfram.erlhagen@mct.uminho.pt

in section V. We finish with conclusions and future work.

II. COGNITIVE CONTROL ARCHITECTURE FOR JOINT ACTION

In Fig. 2 we present the schematic view of the cognitive control architecture. The observation layer (OL) represents information about the hand motion of the partner in terms of a specific goal (e.g. 'reaching towards a wheel/bolt'), and the classification of the grasping behaviour in terms of the grip type (top grip, side grip or bottom grip). In addition, gestures like pointing or hand-out are represented in this layer. The action simulation layer (ASL) contains the representation of goal-directed action sequences in the motor repertoire of the observer (e.g., reaching-grasping-attaching a wheel to the axle) . A particular sequence may become activated by observed motor acts (e.g., reaching or grasping) represented in OL. The motor sequences in ASL are linked to representations in layer IL that encode the intentions underlying the observed motor behaviour (e.g., my partner grasps a bolt with a side grip to fix the wheel on the axle, hand me over a wheel etc.). During action observation, these representations become automatically activated once the associated action sequences are triggered (motor simulation) by input from the action observation layer OL and additional environmental cues (e.g., the distribution of components in the two working areas). The object memory layer (OML) represents the pieces that are in the workspace of each agent (e.g. 'there is one wheel in the workspace of my partner'). In the common sub-goal layer (CSGL) the currently active sub-goals for the team are encoded (e.g. 'insert the left bolt'). Finally, the action execution layer (AEL) contains representations of all possible complementary action sequences (e.g. 'reach and grasp wheel with top grip and mount it on the axle'). The most adequate complementary behaviour is selected in a competition process that integrates the information about the common sub-goals (CSGL), the inferred intention of the other robot (IL) and the spatial distribution of objects in the two working areas (OML). For more details see [5].

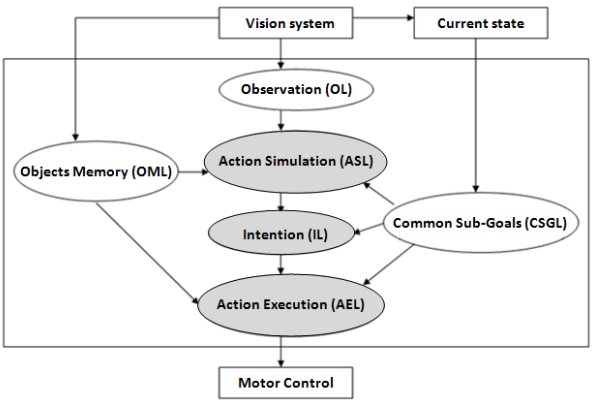


Fig. 2. The schematic view of the cognitive control architecture for the joint construction task.

The core part of the cognitive architecture, i.e., layers ASL, IL and AEL, can be seen as a distributed network of decision making systems (e.g. "What is the goal-directed action of my partner?", "What is the intention of my partner?", "Which action should I select?"). In [5] these layers have been formalized as DNFs that describe the dynamics of neural population activity. DNFs implement a decision making process based on a combination of recurrent excitatory interactions within a local pool of neurons and lateral inhibitory interactions between neural populations. Here we assume instead that each layer is modelled as a set of individual neurons that mutually compete with each other. In the following we describe what is represented by the neurons in the different layers from the perspective of robot R1 which takes into account the actions and inferred goals of its partner R2. As shown in Fig. 3, the ASL layer has eight neurons that represent the possible actions sequences of robot R2: 'RW-TG-I, Reach to Wheel in workspace with Top Grip and Insert wheel on base'; 'RW-SG-H, Reach to Wheel in workspace, grasp it with Side Grip and Hold it out for R1'; 'RWhR1-TG-I, Reach Wheel from R1's hand with Top Grip and Insert wheel on base'; 'RB-SG-I, Reach to Bolt in workspace, grasp it with Side Grip and Insert bolt on base'; 'RB-TG-H, Reach to Bolt in workspace, grasp it with Top Grip and Hold it out for R1'; 'RBhR1-BG-I, Reach Bolt from R1's hand with Bottom Grip and Insert bolt on base'; 'RhtoR1EW, Reach empty hand toward R1 Expecting a Wheel'; 'RhtoR1EB, Reach empty hand toward R1 Expecting a Bolt'. The IL layer is formed by four neurons that represent the possible intentions of R2: 'IW, Insert Wheel'; 'HW, Handover Wheel'; 'IB, Insert Bolt'; 'HB, Handover Bolt'. The layer AEL eight neurons represent the possible complementary action sequences the observing robot R1 may execute. The sequences are similar to the ones represented in the action simulation layer ASL: 'RW-TG-I', 'RW-SG-H', 'RWhR2-TG-I', 'RB-SG-I', 'RB-TG-H', 'RBhR2-BG-I', 'RhtoR2EW', 'RhtoR2EB'.

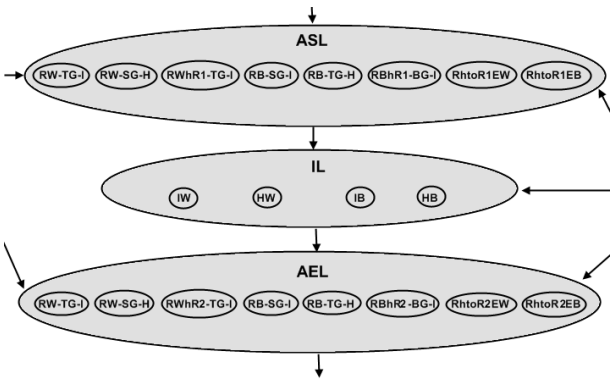


Fig. 3. The labels of the different neurons in layers ASL, IL and AEL are shown. For more details see the text.

III. DYNAMICAL COMPETITIVE SYSTEM

Model layers ASL, IL and AEL implement a competitive dynamics defined as a continuous system of first order differential equations:

$$\alpha_i \frac{dx_i}{dt} = \beta_i x_i - |\beta_i| x_i^3 - \sum_{j \neq i} \gamma_{ji} x_j^2 x_i + f_{stoch}, \text{where} \quad \beta_i \in \mathbb{R},$$
$$\alpha_i, \gamma_{ji} \in \mathbb{R}^+, x_i mod[-1,1],\ i,j = 1,...,N,\ j \neq i \quad (1)$$

with one independent variable t, N dependent variables $x_1, x_2, ..., x_N$, and the parameters α_i, β_i, and γ_{ij}.

This nonlinear dynamical system, with adequately tuned parameters, can be interpreted as a set of N neurons, x_i, which receive external input modelled by parameters β_i. All neurons in a layer compete with each other and the winning neuron x_i that becomes activated close to the maximal activation level 1 represents a unique decision. At any given time, the state of the dynamical system is given by a point $x(t) = (x_1(t), x_2(t), ..., x_n(t))$ in the state space. The neural activation x evolves continuously in time, as determined by the vector field $\dot{x} = f(x, parameters)$. A point that does not change in time is called fixed point. The fixed points are constant solutions of the dynamical system, i.e., are the points at which the vector field is null,

$$\dot{x}|_{x_{fixed\,point}} = f\left(x_{fixed\,point}\right) = 0. \quad (2)$$

Consider that P_i is a point of $\mathbb{R}^N$ whose components are all zero except the i-th component that is ± 1. It is desired that, when existing, the points $(0,...,0)$ and $P_i \in \mathbb{R}^N$ are fixed points of the system. The fixed point $(0,...,0)$ represents the state where all neurons are inactive. Since we want the system to represent a decision, this is an undesired state, that is, this fixed point should be a reppeler. The fixed point P_i represents the desired state where neuron x_i is active and all others are silent, so it must be an attractor (asymptotically stable fixed point[1]).

The parameters α_i, β_i, and γ_{ij} control the behaviour of the system. Parameter α_i defines the time scale of the system, β_i external input to neuron x_i, and γ_{ji} controls the inhibition of neuron x_j over neuron x_i. The stochastic force, $f_{stoch} = \sqrt{Q}\xi_n$ where ξ_n is the Gaussian white noise of unit variance and Q is the effective variance of the force, ensures that the system escapes from repellers within a limited time.

Next we analyze the contribution of the terms in (1) to the behavioural dynamics of the system. Consider the first part of the system (1)

$$\alpha_i \frac{dx_i}{dt} = \beta_i x_i - |\beta_i| x_i^3, x_i mod[-1,1]. \quad (3)$$

The components of the fixed points, $x(t)$, are 0 or ± 1, when $\beta_i > 0$, and are all 0, when $\beta_i < 0$. Therefore the equilibrium value of each neuron x_i is 0 or ± 1. In this situation there is no competition, each neuron is either inactive or active. Competition is introduced by the following term

$$-\sum_{j \neq i} \gamma_{ji} x_j^2 x_i. \quad (4)$$

If neuron x_i is inactive its value is 0 and the value of product $\gamma_{ji} x_j^2 x_i$ is also 0 for any x_j, i.e., the neuron remains inactive. If neuron x_i is active we have to distinguish two cases:

- when x_j is inactive the value of $\gamma_{ji} x_j^2 x_i$ is 0 and the value x_i continues active;
- when x_j is active the value of x_j is 1 or -1, so $x_j^2 = 1$ and $\gamma_{ji} x_j^2 x_i > 0$ if $x_i = 1$ or $\gamma_{ji} x_j^2 x_i < 0$ if $x_i = -1$.

Note that, if $x_i = -1$ or $x_i = 1$, the value of term (4) is 0 if all neurons x_j are inactive or is a positive value or negative value respectively, i.e., the signal of term (4) is symmetric of the signal of x_i so that there is inhibition from neuron x_j to neuron x_i.

[1]If a variable is slightly displaced from a fixed point, it may move back to the fixed point. For a description of how the fixed points are analytically determined and for a study of their stability see [6].

IV. FIXED POINTS, THEIR STABILITY AND PARAMETERS SETTINGS

For determining the values of the parameters we have analyzed the existence and stability of the fixed points $(0,...,0)$ and $P_i, i = 1,...,N$. This analyses has been based on the qualitative theory of Dynamical System [6] [7].The conclusions of this study, considering $\gamma_{ji} > 0$ e $\alpha_i > 0$, $i, j = 1,...,N$, $i \neq j$, are presented in the following table: The parameters values β_i are determined in ASL, IL, and

TABLE I
EXISTENCE AND STABILITY

Fixed points	$(0,...,0)$	P_i
Existence	Always	If $\beta_i > 0$
Asymptotically stable	If $\beta_i < 0, \forall i$	If $\beta_j < \gamma_{ij}, \forall j$
Unstable	If $\exists i : \beta i > 0$	If $\exists j : \beta_j > \gamma_{ij}$

AEL layers from information of the other connected layers. In the following we present the expressions that give the value β_i in each of this three layers:

$$\text{ASL: } \beta_i = \sum_{j=1}^{4} W_{ji} * CSGL_j + \sum_{k=1}^{10} W_{ki} * OL_k + \sum_{l=1}^{4} W_{li} * ML_l, \quad i = 1,...,8; \quad (5)$$

$$\text{IL: } \beta_i = \sum_{j=1}^{4} W_{ji} * CSGL_j + \sum_{k=1}^{8} W_{ki} * ASL_k, \quad i = 1,...,4; \quad (6)$$

$$\text{AEL: } \beta_i = \sum_{j=1}^{4} W_{ji} * CSGL_j + \sum_{k=1}^{4} W_{ki} * IL_k + \sum_{l=1}^{4} W_{li} * ML_l, \quad i = 1,...,8; \quad (7)$$

where W is the weight matrix that represents the connection strengths between the neurons of two layers. $CSGL_j$, OL_k, ML_l, ASL_k, IL_k are the values of the neuron's activation in each layers, which is 1 if the neuron is active and 0 if is inactive. Neuron x_i is active if $P_i \in \mathbb{R}^N$ is a fixed point of the system and this occurs if $\beta_i > 0$. So we conclude that x_i is active if $\beta_i > 0$.

The parameters value γ_{ji} are determined as a function of β_i and β_j, for $\beta_i, \beta_j > 0$, and are 0 otherwise. Note that, competition between the neurons x_i and x_j exists only if they are active, i.e., if β_i and β_j are positive. From Table I we conclude that the value of the parameter γ_{ji} must satisfy two conditions:

- the larger the difference $\beta_j - \beta_i$ the larger must be the inhibition of neuron x_j to neuron x_i;
- the value of γ_{ji} should be larger than β_i to guarantee that the fixed points are stabel.

So, γ_{ji} is given by the following expression:

$$\gamma_{ji} = \begin{cases} \beta_i + e^{-\frac{\beta_i}{\beta_j}}, & \text{if } \beta_i, \beta_j > 0 \\ 0, & \text{if } \beta_i < 0 \vee \beta_j < 0 \end{cases}. \quad (8)$$

Note that $e^{-\frac{\beta_i}{\beta_j}}$ is a decreasing function that varies between 0 and 1 for positive values of $\frac{\beta_i}{\beta_j}$. If $\beta_j - \beta_i > 0$, the larger this difference the smaller is the value of $\frac{\beta_i}{\beta_j}$ and the larger is the value of $e^{-\frac{\beta_i}{\beta_j}}$. If $\beta_j - \beta_i < 0$, the larger this difference the larger is the value of $\frac{\beta_i}{\beta_j}$ and the smaller (close to zero) is the value of $e^{-\frac{\beta_i}{\beta_j}}$.

The velocity with which the system is converging or diverging from the fixed point is defined by the real part of the eigenvalues, λ_i, of the Jacobian matrix [2]. The larger $Re(\lambda_i)$, the larger is this velocity. The local time constant near a fixed point is given by the expression $\tau = min\{\tau_i\}$, where $\tau_i = \frac{1}{|Re(\lambda_i)|}$. The system becomes the faster the smaller the value of τ. The τ's values of the fixed points $(0,...,0)$ and P_i are shown in table II: To guarantee

TABLE II
τ'S VALUES

Fixed points	$(0,...,0)$	P_i
Values of τ	$min\left\{\frac{\alpha_i}{\mid \beta_i \mid}\right\}$	$min\left\{\frac{\alpha_i}{2\beta_i}, \frac{\alpha_j}{\mid \beta_j - \gamma_j \mid}\right\}$

numerical stability of the Euler method, which we used for the model simulations, the following condition must hold:

$$dt \ll \tau_{i_{min}}. \tag{9}$$

Taking into account the values of Table II we can compute $\tau_{i_{min}}$:

$$\tau_{i_{min}} = \min\left\{\frac{\alpha_i}{\mid \beta_i \mid}, \frac{\alpha_i}{2\beta_i}, \frac{\alpha_i}{|\beta_i - \gamma_{ji}|}\right\}. \tag{10}$$

Finally setting

$$\alpha_i = 10\ dt\ \max\left\{|\beta_i|, 2\beta_i, |\beta_i - \gamma_{ji}|\right\} \tag{11}$$

guaranties the desired numerical stability of the system.

In order to illustrate the behaviour of the dynamical system for different parameter values, we discuss here two simple examples with two neurons x_1 e x_2. In the first example, we assume $\beta_1 = 2$, $\beta_2 = -1$, $\gamma_{12} = \gamma_{21} = 0$, $\alpha_1 = 4$, and $\alpha_2 = 1$, so we have the following dynamical system:

$$\begin{cases} \frac{dx_1}{dt} = \frac{2x_1 - 2x_1^3}{4} \\ \frac{dx_2}{dt} = \frac{-x_2 - x_2^3}{1} \end{cases}. \tag{12}$$

Fig. 4 [3] shows the phase space of the system (12). We can see the fixed points and understand qualitatively the dynamic behaviour of the system in the vicinity of these points.

The system (12) has three fixed points: $(0,0)$ and $(\pm 1,0)$. As shown in phase space diagram, the fixed point $(0,0)$ is unstable (repeller), and $(\pm 1,0)$ are asymptotically

[2]The matrix of the partial derivatives of the $f_i = \left(\beta_i x_i - |\beta_i| x_i^3 - \sum_{j \neq i} \gamma_{ji} x_j^2 x_i\right)/\alpha_i$ with respect to state variables x_i.

[3]Phase space were obtained from function "pplane7" developed by Polking(2003).

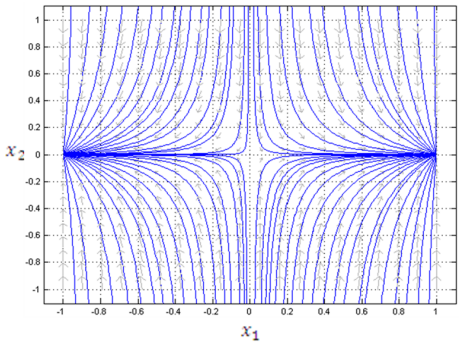


Fig. 4. Phase space of the dynamical competitive system for $\beta_1 = 2$, $\beta_2 = -1$, $\gamma_{12} = \gamma_{21} = 0$, $\alpha_1 = 4$, and $\alpha_2 = 1$.

stable (attractors). In this example, neuron x_1 is active. Neuron x_2, however, could never become active because the fixed point$(0,\pm 1)$ does not even exist. This happens to occur because $\beta_2 < 0$.

In the second example, we assume $\beta_1 = 2$, $\beta_2 = 1$, $\gamma_{12} = 1 + e^{-\frac{1}{2}}$, $\gamma_{21} = 2 + e^{-2}$, $\alpha_1 = 4$, and $\alpha_2 = 2$, so we have the following dynamical system:

$$\begin{cases} \frac{dx_1}{dt} = \frac{2x_1 - 2x_1^3 - \left(2 + e^{-2}\right) x_2^2 x_1}{4} \\ \frac{dx_2}{dt} = \frac{x_2 - x_2^3 - \left(1 + e^{-\frac{1}{2}}\right) x_1^2 x_2}{2} \end{cases}. \tag{13}$$

The system (13) presents nine fixed points:$(0,0)$, $(\pm 1,0)$,

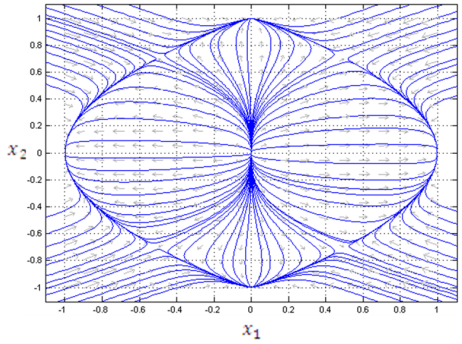


Fig. 5. Phase space of the dynamical competitive system for $\beta_1 = 2$, $\beta_2 = 1$, $\gamma_{12} = 2$, $\gamma_{21} = 3$, $\alpha_1 = 4$, and $\alpha_2 = 6$.

$(0,\pm 1)$ and $(\pm 0.3,\pm 0.9)$ approximately, as we can observe in phase space in Fig. 5 [3]. The fixed points $(\pm 1,0)$ and $(0,\pm 1)$ are asymptotically stable (attractors) and the others are unstable (repellers). This implies that there are two potential alternatives represented and therefore a competition process takes place (cf. Section IV, first situation) which defines the winning neuron representing the final decision.

V. SIMULATION RESULTS

The presented dynamical competitive system has been tested through computer simulations. The simulations were made in Matlab using M-files created for layer ASL, IL and AEL. The ordinary differential equations were integrated using the progressive Euler method with fixed time step [8]. The runnig of the M-files give the activation in the layers, over time. The images used for illustrating the behaviour of the robots comes from a robot simulator

developed in our lab [9][4]. Next, we present three situations in some more detail that representative for the joint construction task at hand. Initially, the round platform is placed between the two robots, R1 (blue on the left) and R2 (cyan on the right). The components are distributed in the two working areas of the teammates. R1 is supposed to act as a social companion serving the needs of its partner R2, whereas R2 is supposed to decide what to do next without taking into account inferred goals of R1.

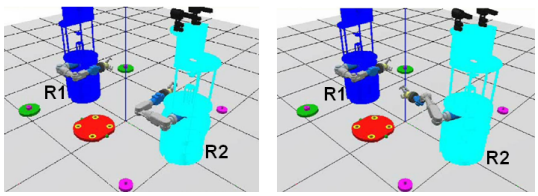


Fig. 6. Snapshot of the robots engaged in the construction task. The sub-goals are "insert the right wheel" and "insert the left wheel". There are two wheels in the workspace of R1 and two bolts in the workspace of R2. R2 reaches its hand towards R1.

In the first situation, the two wheels are in the workspace of R1 and the two bolts are in workspace of R2. When the task starts, the current sub-goals for the team are "insert left wheel" and "insert right wheel". Fig. 6 illustrates that

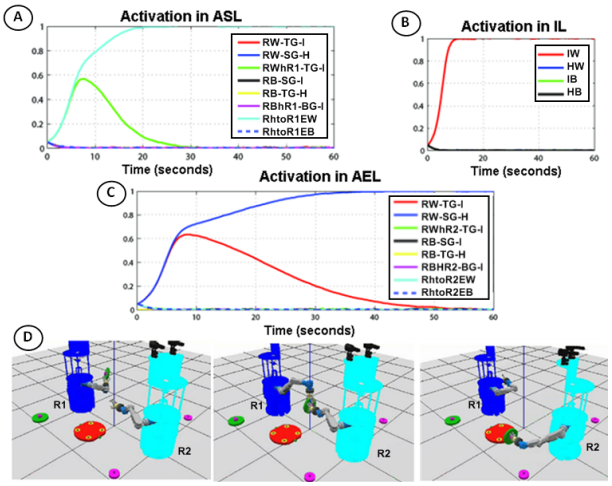


Fig. 7. Panel A,B,C shown the activation of each neuron, over time, in layers ASL, IL and AEL, respectively. Panel D illustrates the overt behaviour R1 grasping and handing over a wheel to R2, and R2 grasping the wheel and attaching it to the platform.

there are two potential decisions that R1 can make: grasp a wheel and insert it on the platform or grasp a wheel and hand it over to R2. Fig. 7 shows that R1 understand the gesture of R2 as demanding a wheel (activation of the neuron 'RhtoR1EW' in ASL, panel A), and infers that the intention of R2 is to insert a wheel (activation of the neuron 'IW' in IL, panel B). R1 decides to satisfy the request (activation of the neuron 'RW-SG-H' in AEL, panel C). We can see in panel C that initially the two neurons ('RW-TG-I'and 'RW-SG-H') that represent the two possible decisions ('reach and grasp wheel with top grip and mount it on the axle' or 'reach and grasp wheel with side grip and hand over a wheel to R2') compete for expression in overt behaviour. The decision of R1 to serve R2 first is the most appropriate for the team performance since R2 has no wheel in its workspace. The overt bahaviour of the two robots is illustrated in panel D.

In the second example, the construction task is in the final stage. Both agents are aware that only a bolt on the side of R2 is still missing. The bolt is in the workspace of robot R1. However R1 cannot see it from its present position since it is placed behind it. As we can be seen in Fig. 8, R2 hands out its hand in the direction of R1 (Panel A) and R1 infers that R2 is expecting to receive a bolt (activation of the neuron 'RhtoR1EB' in ASL, panel B) to attach it on its side of the construction (activation of the neuron 'IB' in IL, panel C). R1 decides to grasp a bolt for handing it over to the teammate (activation of the neuron 'RB-TG-H' in AEL, panel D), but has first to look for it. The inferred action goal of R2 thus triggers a visual search as the most appropriate complementary action.

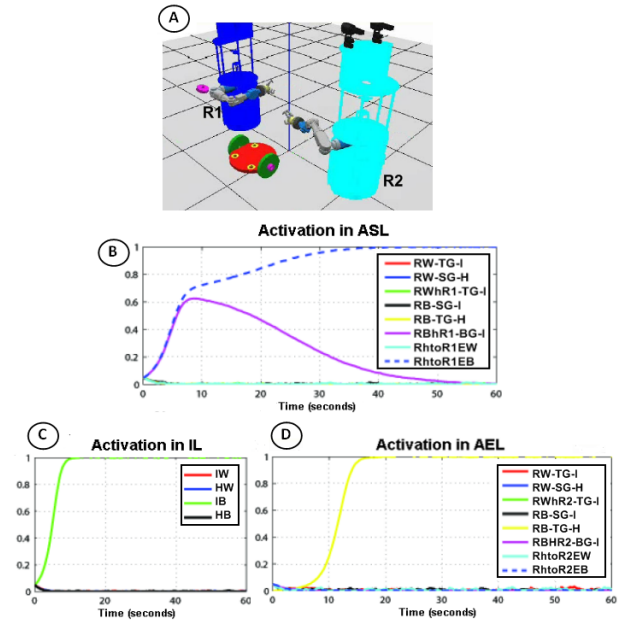


Fig. 8. Panel A shows R2 reaching with its empty hand towards R1. The sub-goal is "insert the right bolt" but R1 cannot see the bolt in its work space form its present position. Panel B, C e D show the time course of the activation of each neuron in layers ASL, IL and AEL, respectively.

Even in the still relatively simple joint construction of the toy vehicle, R2 can make errors or act in an inefficient manner. This is illustrated in the third example. There is a wheel in R1's workspace and in R2's workspace there are a wheel and two bolts. R2 reaches its hand towards R1. R1 interprets again this movement as a "request for a wheel" with the intention to attach it (activation of neuron 'IW' in IL) since an active sub-goal is to insert a wheel on R2's side (Fig. 9). However, the overt behaviour of R2 is considered an error because there is a wheel in its own workspace and thus no need to coordinate a handing over procedure. R1 decides to ignore the request of the partner and grasps instead a wheel with the intention to insert it on its construction side (Fig. 10).

VI. CONCLUSIONS AND FUTURE WORK

We have presented a competitive dynamical system that models the decision process for complementary action

[4]Examples of simulations made in the simulator can be seen in http://mobileanthropomorphicroboticsgroup.blogspot.com.

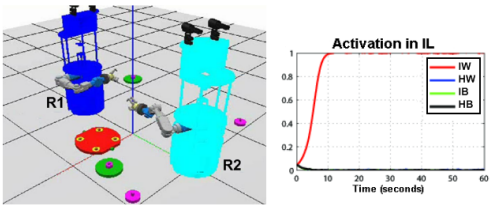


Fig. 9. Simulation experiment illustrating an inefficient action of robot R2. The hand of R2 is reaching towards R1, attaching the two wheels are current sub-goals for the team. R1 infers the intention of R2 (activation of the neuron 'IW' in IL layer). However, there is a wheel inside the workspace of R2.

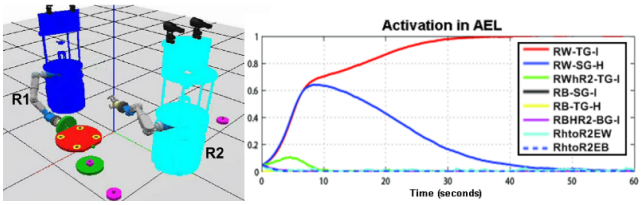


Fig. 10. Illustration of R1 inserting wheel on the platform. The evolution of the activity if the neuron 'RW-TG-I' (reach and grasp wheel with top grip to insert it on the axle of the platform) in AEL representing the decision of R1 is shown.

selection in a cooperative joint action task. The dynamic control architecture is based on a distributed network of neurons, each with specific functionalities. It implements the notion that cognitive processes like decision making or prediction unfold continuously over time under the influence of different external and internal information sources [10]. In order to guarantee an efficient strategy of the team, a robot must be able to understand the actions of the teammate and to infer its goals. The mechanism proposed here is motor simulation. Observed motor acts represented by specific neurons are mapped onto congruent representations in the motor repertoire of the observer. Since the observer is supposed to know the outcomes of its own action sequences it may use a covert motor simulation to make sense of the motor behaviour of its partner in the joint action task [11]. As shown in the examples, the state of the construction and environmental cues like the distribution of objects in the two working areas are additional information sources that play important roles in the inference and decision processes.

To model complementary action selection in joint action we use the mathematical framework of nonlinear dynamical systems that has attracted over the last 15 years a lot of attention in cognitive science and cognitive robotics [11] [12] [13] [14] [15] [16]. This interest results from the fact that the theory of dynamical systems provides the adequate mathematical tools to analyze the dynamics of cognitive processes. Dynamic neural fields (DNFs) formalized as non-linear integro-differential equations have been successfully applied in the past to different robotics problems (e.g. [5] [16] [17] [18] [19], for an overview see [18]). The advantage of using ordinary differential equations is the fact that they are computationally more efficient than DNFs. However the present implementation has the disadvantage that the neural activity cannot become self-stabilized. Self-sustained activity due to recurrent interactions has been exploited in DNF-based control architectures to implement for instance a working memory function (for an over view see [18]). Adding extra layers to the control architecture based on ordinary differential equations that temporally store previously stable activation states may be a manner to overcome this limitation of the presented model .

In the future, we will implement the simplified architecture for joint action in real robots, and extent the competitive dynamic model to more complex cooperative tasks.

VII. ACKNOWLEDGMENTS

Thanks to E. Silva, E. Sousa, J. Ferreira, J. Silva, M. Pinheiro, M. Vaz, N. Hipólito, L. Louro, T. Machado, R. Silva for their contributions. The present research was conducted in the context of the project COOPDYN, "Synthesis of Cooperative Behavior in Multi-robot Systems:a Nonlinear Attractor Dynamics Approach", financed by the FCT and the FEDER (POSI/SRI/38051/2001), and of the EU-project JAST, "Joint-Action Science and Technology", financed by the EC (IST-2-003747-IP).

REFERENCES

[1] L. S. Homem de Mello and A. C. Sanderson, "AND/OR graph representation of assembly plans", *IEEE Trans. Robotics Automation, vol. 6, No. 2*, April, 1990, pp. 188-199.
[2] A. Knoll, "Distributed contract networks of sensor agents with adaptive reconfiguration: modelling, simulation, implementation", *Journal of the Franklin Institute, Elsevier Science, Vol. 338, No. 6*, September, 2001, pp. 669-705.
[3] N. Blaylock and J.Allen, "A collaborative problem solving model of dialogue", in *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue, Lisbon, Portugal*, September, 2005, pp.200-211.
[4] C. Frith and D.Wolpert, "The Neuroscience of Social Interaction", *Oxford University Press*, 2004.
[5] E. Bicho, L. Louro, N. Hipólito and W. Erlhagen, "A dynamic neural field architecture for flexible and fluent human-robot interaction", *in Proceedings of the 2008 International Conference on Cognitive Systems*, University of Karlsrube, Germany, 2008, pp. 179-185.
[6] L. Perko, "Differential Equations and Dynamical Systems", *Texts in Applied Mathematics 7, Springer-Verlag*, 1991.
[7] Y.A. Kuznetsov, "Elementes of Applied Bifurcation Theory,Second Edition", *Springer-Verlag*, 1998.
[8] S.C. Chapra and R.P. Canale, "Numerical Methods for Engineers", *McGRAW-HILL International Editions, Applied Mathematics Series*, 1985, pp. 576-588.
[9] E.Bicho, W. Erlhagen, D5.8 Report on implementing guidelines by WP2 by UMP (T30), *Deliverable 5.8, 19 pages, EU Project JAST*, December, 2006.
[10] R. Beer, "Dynamical approaches to cognitive science", *Trends in Cognitive Science, Vol.4*, March, 2000, pp. 91-98.
[11] A. Clark and R. Grush, "Towards a Cognitive Robotics", *Adaptive Behavior*, Vol.7, 1999, pp. 5-16.
[12] E. Thelen,and L. Smith, "A dynamic systems approach to the development of cognition and action", *The MIT Press*, 1994.
[13] E. Thelen, C. Scheier, G. Schöner and L. Smith, "The dynamics of embodiment: a field theory of infant persevertive reaching", *Behav. Brain Sci, in press*, 2003.
[14] J. Kelso, "Dynamic Patterns: The Self-Organization of Brain and Behavior", *MIT Press*, 1995.
[15] G. Schöner, P. Zanone and J. Kelso, "Learning as change of coordination dynamics: theory and experiment", *Journal of motor behavior, vol. 24*, 1992, pp. 29-48.
[16] G. Schöner, M. Dose and C. Engels, "Dynamics of behavior: Theory and applications for autonomous robot architectures", *Robotics and Autonomous Systems, vol.16*, December, 1995, pp. 213-245.
[17] E. Bicho, P. Mallet and G. Schöner, "Target representation on an autonomous vehicle with low-level sensors", *The International Journal of Robotics Research, vol. 19*, 2000, pp. 424-447.
[18] W. Erlhagen and E. Bicho, "The dynamic neural field approach to cognitive robotics", *Journal of Neural Engineering, vol. 3*, 2006,pp. R36-R54.
[19] C. Faubel and G. Schöner, " Learning to recognize objects on the fly: A neurally based dynamic field approach", *Neural Networks, vol. 21*, 2008, pp. 562-576.