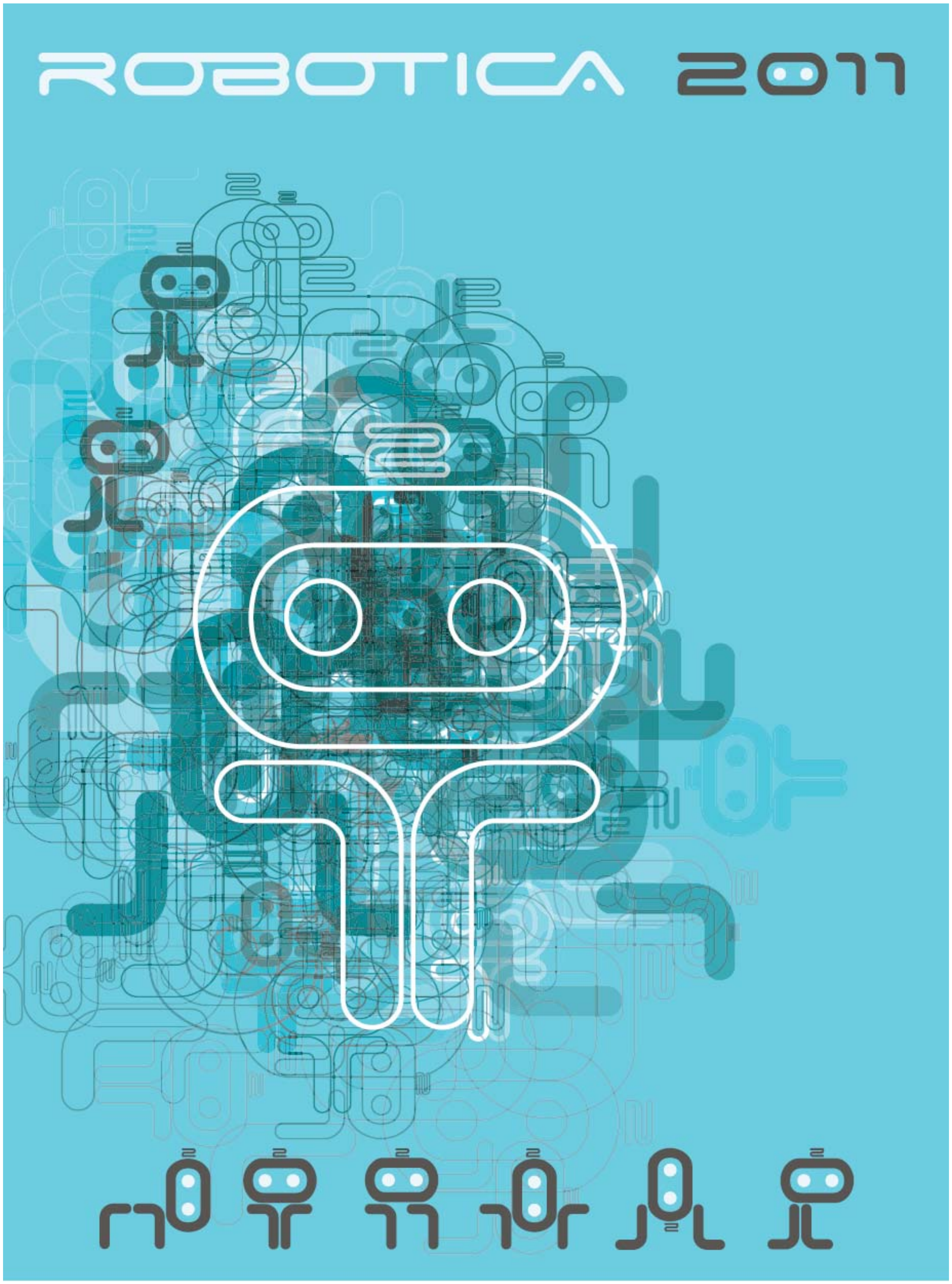


Proceedings of the 11th International Conference on Mobile Robots and Competitions, April 6th, 2011, Instituto Superior Técnico, TU Lisbon, Portugal

Proceedings of the

11th International Conference on Mobile Robots and Competitions

(ROBÓTICA2011)

Pedro Lima, Carlos Cardeira

Instituto Superior Técnico, TU Lisbon

6th April 2011

Proceedings of the11th International Conference on Mobile Robots and Competitions

(ROBÓTICA2011)

Pedro Lima, Carlos Cardeira

Instituto Superior Técnico, TU Lisbon

6th April 2011

Organized by:

Instituto Superior Técnico – Techincal Univerty of Lisbon

Technical Co-Sponsorship:

IEEE Robotics and Automation Society

Portuguese Robotics Society

Sponsored by:

IEEE Robotics and Automation Society

FCT - Portuguese Science Foundation

Portuguese Journal Robotica

Hosted by:

Instituto Superior Técnico – Techincal Univerty of Lisbon

Proceedings Production:

Instituto Superior Técnico – Techincal Univerty of Lisbon

WebSite and cover design:

Ricardo Baeta

Núcleo de Multimédia e e-Learning (NME), Instituto Superior Técnico – Techincal Univerty of Lisbon

WebSite:

http://robotica2011.ist.utl.pt/

Welcome message from the Chairs of the Conference Committee

Welcome to the 11th International Conference on Mobile Robots and Competitions, the scientific meeting of Robotica2011, the 11th edition of the Portuguese Robotics Open (Festival Nacional de Robótica). This year we celebrate the 10th anniversary of the Festival, the annual event of the Portuguese Robotics Society (Sociedade Portuguesa de Robótica - SPR). Once again, 600 Portuguese (and some foreign friends as well) students and teachers from all education levels meet together to share their latest advances in robot competitions and S&T research.

Twenty seven papers were submitted to the scientific meeting, from which the International Program Committee selected sixteen for presentation in the meeting and inclusion in the final Proceedings. Most reviews were thorough and constructive, a sign of respect for the work of all the authors who submitted their work to the meeting*. There will be a Best Paper Award, and the best 5 papers will be published in the Portuguese magazine Robotica.

Though the meeting main goal is to join together the Portuguese researchers in Robotics, it has been enriched in recent years by the co-technical sponsorship of the IEEE Robotics and Automation Society (RAS), the inclusion of world top researchers in the Steering and International Program committees, and the submission of several papers from abroad. This year 8 of the submitted papers were co-authored by a majority of non-Portuguese colleagues. Vijay Kumar has honored us by accepting our invitation for two talks during the competition days, presenting his sound and exciting work on multirobot systems.

We would like to thank several people and institutions who made Robotica2011 possible. IST President, Prof. António Cruz Serra, for his enthusiastic support since the very beginning, other members of IST's Conselho de Gestão who worked with us directly (Palmira Silva, Vítor Leitão), NME, especially Ricardo Baeta, for the fantastic work on the web page, not forgetting the space, infrastructure and personnel provided free of cost, Ciência Viva for its recurrent financial support and dedication to the cause of the promotion of S&T in Portugal, our main sponsors Technical University of Lisbon and INFAIMON, the Portuguese Foundation for Science and Technology, IEEE RAS, CGD, BPI, IPL and, of course SPR and its Specialized Committee for the Festival. The Associated Labs for Energy, Transports and Aeronautics and Institute for Systems and Robotics are our research institutions and supported our participation in the event. Last, but not the least, we thank all the members of the Local Organizing Committee, Volunteers and IST workers who did all the work that made it possible.

We hope you will enjoy Robotica2011!

Pedro Lima, Carlos Cardeira

*The papers co-authored by one of the organizers were reviewed by a set of reviewers selected by the other organizer, whose names are not known to the organizer co-author.

Organizing and Steering Committees

Organizing Committee

Pedro Lima
Carlos Cardeira

Steering Committee

Bruno Siciliano
Toshio Fukuda
Alessandro De Luca

International Programm Committee

Jose Almeida
Luis Almeida
Helder Araujo
Jose Azevedo
Antonio Bandera
Estela Bicho
Carlos Carreto
Alicia Casals
Rui Cortesão
Paulo Costa
Hugo Costelha
Bernardo Cunha
Jorge Dias
Dimos V. Dimarogonas
J. Felippe De Souza
Jorge Ferreira
Paolo Fiorini
Nicolas Garcia-Aracil
Luis Gomes
Paulo Goncalves
Huosheng Hu
François Ingrand
Guy Juanole
Gerhard Kraetzschmar
Jose Lima
Gil Lopes
Philippe Martinet
Alfredo Martins
Aníbal Matos
Paulo Menezes
Antonio P. Moreira
Luis Moreno
Daniele Nardi
Itsuki Noda
Urbano Nunes
Paulo Oliveira
Antonio Pascoal
Federica Pascucci
João Pinto
Luis Paulo Reis
A. Fernando Ribeiro
Isabel Ribeiro
Antonio Ruano
Jose Sa Da Costa
Vitor Santos
Angel Sappa
Luis Seabra Lopes
Joao Sequeira
Bruno Siciliano
Eduardo Silva
Filipe Silva
Florent Teichteil
Jose Tenreiro Machado
Luigi Villani

Venue

Congress Center, Instituto Superior Tecnico

Conference Programm

08:30-08:50 Registration

08:50-09:00 Welcome session

09:00-10:40 Session 1: Navigation & SLAM
Chair:

09:00-09:25 Tiago Nascimento, André Gustavo Scolari Conceição and António Paulo Moreira, *A Modified A* Application to a Highly Dynamic Unstructured Environment*

09:25-09:50 Fernando Carreira, João Calado and Carlos Cardeira, *Mobile Robot Navigation Planning in a Human Populated Environment*

09:50-10:15 Josep Aulinas, Amir Fazlollahi, Joaquim Salvi, Xavier Lladó, Yvan Petillot, Rafael García and Jamil Sawas, *Robust automatic landmark detection for underwater SLAM using side-scan sonar imaging*

10:15-10:40 Claus Lenz, Thorsten Röder, Markus Rickert and Alois Knoll, *Distance-weighted kalman fusion for precise docking* problems

10:40-11:00 Coffee-Break

11:00-12:40 Session 2: Planning and Decision-Making
Chair:

11:00-11:25 Susana Brandao, Manuela Veloso and João P. Costeira, *Active Object Recognition by Offline Solving of POMDPs*

11:25-11:50 Jesus Capitan, Luis Merino and Anibal Ollero, *Multi-robot coordinated decision making under mixed observability through decentralized data fusion*

11:50-12:15 Bruno Lacerda, Pedro U. Lima, Javi Gorostiza and Miguel Salichs, *Petri Net Based Supervisory Control of a Social Robot with LTL Specifications*

12:15-12:40 Ana Rita Mendes, Matthijs Spaan and Pedro U. Lima, *Planning under Uncertainty for Search and Rescue*

12:40-14:00 Lunch

14:00-15:00 Panel 1: Robot Competitions as a Means to Foster I&D and Education in Robotics

15:00-16:15 Session 3: Perception
Chair:

15:00-15:25 Miguel Armando Riem De Oliveira and Vitor Manuel Ferreira Dos Santos, *Autonomous Driving Competition: Perception Approaches used in the ATLAS Project*

15:25-15:50 Nicola Greggio, Alexandre Bernardino and José Santos-Victor, *Robotic Color Image Segmentation by Means of Finite Mixture Models*

15:50-16:15 Arnau Ramisa, David Aldavert, Shrihari Vasudevan, Ricardo Toledo and Ramon Lopez De Mantaras, *The IIIA30 Mobile Robot Object Recognition Dataset*

16:15-16:30 Coffee-Break

16:00-16:50 Session 4: Humanoids
Chair:

16:00-16:25 Luis Rei, Luis Paulo Reis and Nuno Lau, *Optimizing a Humanoid Robot Skill*

16:25-16:50 Nima Shafii, Luis Paulo Reis, Nuno Lau and Lucio Sanchez Passos, *Humanoid Soccer Robot Motion Planning using GraphPlan*

16:50-18:05 Session 5: Learning, Estimation and Applications
Chair:

16:50-17:15 Manfred Hild, Matthias Kubisch and Sebastian Höfer, *Using Quadric-Representing Neurons (QRENs) for Real-Time*

17:15-17:40 Serge Kernbach, *Multi-Modal Local Sensing and Communication for Collective Underwater Systems*

17:40-18:05 Marcelo Petry, Luis Paulo Reis, Antonio Paulo Moreira and Rosaldo Rossetti, Ricardo Toledo and Ramon Lopez De Mantaras, *Intelligent Wheelchair Simulation: Requirements and Architectural Issues*

18:05-18:15 Closing session

20:00-22:00 Conference Dinner
Award Ceremony

Note: 25 mn are allocated per oral presentation, including Q&A.

A Modified A* Application to a Highly Dynamic Unstructured Environment

Tiago P. Nascimento, *Member, IEEE,* André G. S. Conceição, and António Paulo Moreira, *Member, IEEE,*

***Abstract*—This paper presents an application of a modified A* path planning algorithm in a highly unstructured environment. The A* allows the robot to get to the target fast and with few collisions, avoiding obstacles that move as fast as, or even faster than the robot. Two major changes were made, the consideration of an obstacle's safe distance (slack) and an suboptimal value K for gaining in processing time. Some simulations were made using a crowded and highly dynamic environment with twelve randomly moving obstacles. In these first simulations, a middle sized 5DPO robot was used improving the issues involving the robot in following the planned path.**

***Index Terms*—Path planning, mobile robots, obstacle avoidance, dynamic unstructured environment.**

I. Introduction

PATH planning algorithms form a well known area of research in mobile robotics. It's a study that involves from a single robot movement to a group of mobile robots moving in a specific formation. Issues like static or mobile obstacles avoidance, known or unknown worlds and structured or unstructured environments and single or multiple robots' motion are the main study cases in path planning. In this paper it's presented an application of the A* path planning algorithm as a strategy for robot motion planning in the attempt to avoid a crowd of mobile obstacles, sometimes even faster than the robot itself. For instance it's considered a simple target for the robot to reach.

Motion planning algorithms are widely used nowadays. UAV path planning [1], mobile robot outdoor navigation [2], mobile robot indoor navigation [3] and even in video games [4] can be found path planning algorithms to be the solution for many motion planning issues. In this work, it's used the indoor environment for mobile robot path planning aiming a preset target while avoiding mobile obstacles in high velocities. The robots used are the omnidirectional robots used in the Middle-size League (see Fig. 1) from soccer robot championships. A good modeling and control for this platform can be found in [5], [6] and [7] respectively.

Many path planning techniques rose over the years. One among the most famous is the artificial potential field approach. This methodology has been widely used and it states that the collision-free trajectory is generated along the negative gradient of the defined attractive and repulsive potential-field functions. The subsequent studies can be found in [8], [9], and [10]. Nonetheless, the potential-field method is not straightforwardly applicable to mobile vehicles with kinematic constraints since, in the potential-field design, the robot is usually treated as a simple particle. Another major problem is since it's an essentially fastest descent optimization method, it can get trapped into local minima of the potential function other than the goal configuration [11].

T. P. Nascimento, and A. P. Moreira are with the Department of Electrical and Computer Engineering, Faculty of Engineering from University of Porto and INESC-Porto, Porto, 4200-465 Portugal (e-mail: tiagopn@ieee.org and amoreira@fe.up.pt).

A. G. S. Conceição is with Department of Electrical Engineering, Federal University of Bahia, Salvador BA, Brazil. (e-mail: andre.gustavo@ufba.br).

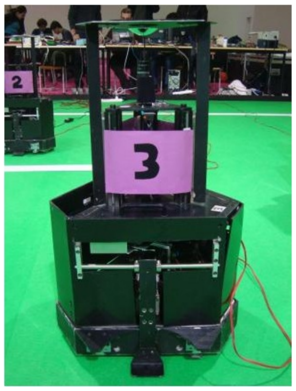


Fig. 1: The 5DPO Middle Size Robot

Over the years solutions for the motion planning problems were also found in artificial intelligence algorithms such as neural networks and fuzzy logic. In early years the use of fuzzy logic was an option for easy controllable systems [12], [13]. Recently, neural networks approaches rose showing considerable results. In [14], the authors propose a neural network based path planner used in multiples nonholonomic mobile robots with moving obstacles. Other authors use neural network approach of non moving obstacles avoidance [15].

Among the most famous is also the Roadmap method. This method can be seen in [16] where a computational geometry data structure was proposed to solve the problem of an optimal path generation between a source and a destination in the presence of simple disjoint polygonal obstacles. In [17] a good application of the Roadmap method is applied where the use of multiple mobile robots in a common environment such as underground mining and warehouse management problem are considered despite no randomly moving obstacles are used. The Roadmap method is well applied for low-dimension configuration spaces and sometimes, depending of the approach, no easy to implement [11].

Finally, the last method among the most classic algorithms for path planning is the Cell Decomposition [11]. In this category are famous and efficient algorithms such as A*, D*, ARA* and AD*. The A* algorithm is the oldest. It's well applied with static [18] and dynamic obstacles [3]. The advantage nowadays of the Cell Decomposition methods is that with the current technology, it is no longer applied to indoor or small spaces. It can be applied from UAV obstacle avoidance [1] to unknown environments [2]. In [19] it was developed an approximate cell-decomposition method in which obstacles, targets, sensor's platform, and FOV (Field of View) are represented as closed and bounded subsets of an Euclidean workspace. A good overview about the advantage and disadvantage of these algorithms cam be seen in [20] and [21].

In this paper it's presented a novel application based in the A* algorithm for highly dynamic and crowded environments. In the next section the problem is formulated describing the issue studied. In section III, the A* path planning algorithm is explained. The results of the experiments and simulations are shown in section IV. Finally, the conclusion is shown in section V.

II. Problem Formulation

As a test bed for the produced algorithm evaluation it was used the robots from robot soccer competition from FEUP, 5DPO - Middle Size league. The middle size league can run up to $2.5ms^{-1}$ in a straight line. The evaluation was done in a simulation environment using a software called SimTwo developed by professor Paulo Costa, PhD, from FEUP with the middle size league robot. In this simulation, the used field has $8x10m$ of size. Also, the grid cell used in A* had $0.05m$ of size, which was chosen to be the bast size by tests of accuracy an precision with different sizes of cell. The A* path planning algorithm was implemented in a software written in free Pascal Lazarus compiler which communicates with SimTwo or the real robot by using UDP protocol.

All simulations were made with the SimTwo software. This software uses an Open Dynamic Engine library [22], [23] which guaranties an exactly realistic simulation from the dynamic of rigid object, resulting in the object's behavior very close to real. The robots, nevertheless, were highly and rigorously previously parametrized in SimTwo [5] and [24], which makes this platform a realistic and ideal simulation environment for tests with path planning algorithms. A screenshot from the program can be seen in the Fig. 2.

Now, let A be a single rigid object - the robot - moving in a Euclidean space W, called workspace, where $W \in \Re^2$. As can be seen in the figure, the simulation sets a 5DPO middle size robot as the rigid object A. In this workspace it's placed in an initial position q_{init} marked with a small circle. In the far side of the workspace W, the target point q_{target} is marked also with a small circle as well. In between the initial and target point there is a bigger circle centered in the middle of the field where it should symbolize any crowded environment such as a shopping mall's hall or a factory corridor, where people have to pass in a random trajectory, or even a near beach air space, where many flaying animals cross a UAV's pass.

Let $B_1,\ldots,B_{12}$ be also, in a first case, fixed rigid objects distributed in W. It is known that the configuration space of A is the space C all configurations of A. Therefore, all

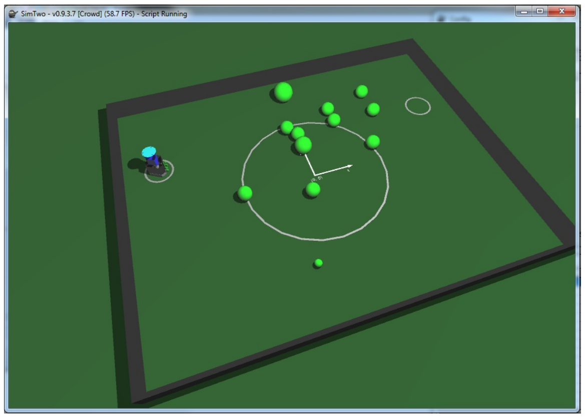


Fig. 2: The SimTwo Simulation Environment

obstacles B_i in the workspace W maps in C to a region $CB_i = \{q \in C/A(q) \cap B_\text{i} \neq 0\}$ which is called **C-obstacle**. The union of all C-obstacles is called **C-obstacle region**, and therefore the set:

$$C_{free} = C \setminus \bigcup_{i=1}^{n} CB_i = \{q \in C/A(q) \cap (\bigcup_{i=1}^{n} CB_i) = 0\} \quad (1)$$

is called the **free space**

Therefore, the problem becomes: Given an initial position and orientation and a target position and orientation of A in W, generate a path τ specifying a continuous sequence of positions and orientations of A avoiding contact with the CB_i's, that minimizes the path τ that starts at the initial position and orientation q_{init}, and terminates at the target position and orientation q_{target}. Report failure if no such path exists [11].

In a second case, let assume now that B_i(t) designates the region of W occupied by the object B_i at time **t** ($\mathbf{t} \geq 0$). Therefore, the dynamic motion planning problem is to plan a collision-free motion of A from the initial configuration q_{init} at time 0 to the target configuration q_{target} at time $\mathbf{T} \geq 0$. The time **T** is called the **arrival time**. The planning problem here requires that a function of time be generated which specifies the configuration of A at every instant in the interval $[0, \mathbf{T}]$. Therefore, the solution of this dynamic path planning problem is to minimize the trajectory of A with respect to the time **t** [11]. This trajectory to be minimized is the equation 4.

$$\mathbf{t} \in [0, \mathbf{T}] \mapsto \mathbf{q}(t) \in C \setminus \bigcup_{i=1}^{n} CB_i(\mathbf{t}) \quad (2)$$

As the C-obstacles are no longer static, in another words they vary their position and orientation with **t**, it can no longer be represented as in the problem before. This issue is solved by simply adding a dimension to C, obtaining therefore, $CT = Cx[0, +\infty)$, which is called **configuration space-time** of A. Thus, every obstacle B_i maps in CT to a stationary region CTB_i, called a **CT-Obstacle**, defined by:

$$CTB_i = \{(q,t)/A(q) \cap B_i(t) \neq 0\} \quad (3)$$

III. The A* Path Planning Algorithm

It's known that most environments are highly dynamic, with obstacles moving randomly and with the dynamic constrains

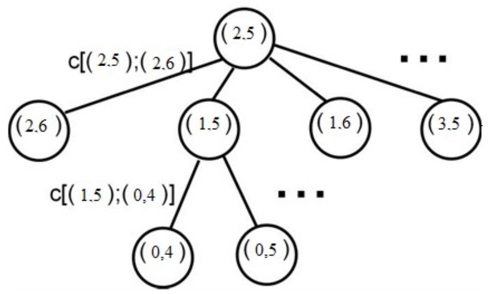
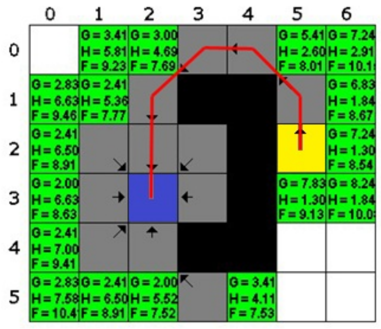


Fig. 3: Resulting Graph from the Cell Division



0 1 2 3 4 5 6
0 1 2 3 4 5
G = 3.41 H = 5.81 F = 9.23
G = 3.00 H = 4.69 F = 7.69
G = 5.41 H = 2.60 F = 8.01
G = 7.24 H = 2.91 F = 10.1
G = 2.83 H = 6.63 F = 9.46
G = 2.41 H = 5.36 F = 7.77
G = 6.83 H = 1.84 F = 8.67
G = 2.41 H = 6.50 F = 8.91
G = 7.24 H = 1.30 F = 8.54
G = 2.00 H = 6.63 F = 8.63
G = 7.83 H = 1.30 F = 9.13
G = 8.24 H = 1.84 F = 10.0
G = 2.41 H = 7.00 F = 9.41
G = 2.83 H = 7.58 F = 10.4
G = 2.41 H = 6.50 F = 8.91
G = 2.00 H = 5.52 F = 7.52
G = 3.41 H = 4.11 F = 7.53

Fig. 4: Map Cell Decomposition

of the robot. The objective then is to find the optimal path between the initial point q_{init} and the target point q_{target}. The A* algorithm is used to do this tasks, even when the environment is highly dynamic. Therefore, one of the concepts that it's needed to highlight is that the optimal path in most of the cases is not the optimal solution, in another words, the best path is not the shortest one, but the fastest one. That is because the robot's velocity is not constant (the robot has limited acceleration) and the robot's controller has difficulty in following trajectories with abrupt changes in direction. To do that, two modifications were made in the A* algorithm to achieve an optimal solution.

A* is a graph search algorithm which calculates the shortest path through a graph between the initial and final node. This algorithm uses a heuristic function

$$F(n) = g(n) + h(n) \tag{4}$$

which estimates the lowest cost of going from the initial to the target point while passing through node n. This sets the order to search for nodes in a way to find the best path as soon as possible. This function is the sum of two other functions.

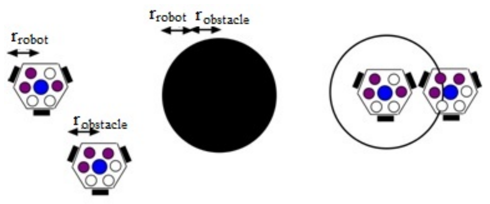


Fig. 5: Obstacle's total radius

1) $g(n)$ = Cost from the origin to node n;
2) $h(n)$ = An heuristic to estimate the cost of the path from node n to the target node.

In this algorithm there are two lists: the O-list and the C-list. The open list, known as the O-list, contains the nodes that are candidates for exploration. The closed list, known as the C-list, contains the already explored nodes. The nodes from C-list 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 "father" node, which is the node used to optimally reach them. This is the node that lies in the shortest path from the origin to current node. The A* algorithm can be seen below.

Algorithm A*

```
1:  Add the initial node to O-list
2:  Do
3:     Choose $n*$ from O-list in which
4:        $F(n*) \leq F(n) \; \forall \; n \in$ O-list
5:     Remove $n*$ from O-list and put into the C-list
6:     For all $n \in$ Star($n*$), which $n \notin$ C-list, do:
7:        if ($n \notin$ O-list) then
8:           add $n$ node to O-list
9:        else if ($g(n*)+c(n*,n) < g(n)$) then
10:          change the father node from $n$ to $n*$
11:       end
12:    end
13: While (O-list $\neq 0$) or ($n*$ = end node)
```

where:

1) Star($n*$) = The set of neighbors to node $n*$
2) c(n_1,n_2) = Cost from going from node n_1 to node n_2
3) $n*$ = Best note in the neighborhood

Once presented the algorithm, it becomes important and necessary to emphasize some considerations. Mainly, it's known that the path τ is optimal if an heuristic function $h(n)$ is admissible. This happens if the function never overestimates the cost to reach the destination, or in another words if

$$h(n) < h_m(n) \forall n \tag{5}$$

where $h_m(n)$ is the lowest cost from n until the destination.

To use the A* algorithm in the calculation of a robot's path, it's necessary to divide the environment map in cells, as stated in the method approximate cell decomposition. Here, 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. 3). 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. In Fig. 4 the black cells represent the obstacles, the yellow cell represents the initial position (node) and the blue cell represents the destination point (node).

Finally, it's considered that the robot A in the workspace W is represented by the initial node and that occupies only a single node, being this last one the geometric center of the objects in a top view. The destination node is the target point q_{target}. All other moving objects are considered C-obstacles (CB_i). As the robot is represented by a single cell, the obstacles have to be bigger in a way to represent both obstacle and robot's body. These obstacles are represented by circles with their radius equal to the sum of the obstacle's radius and the robot's radius. This representation can be seen in the Fig. 5.

IV. The Modifications

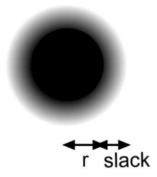


Fig. 6: Obstacle with a slack zone. The black intensity means a higher cost.

The first modification changes the way an obstacle is represented in the cells. A security area is created around the obstacle. This area is built by making the obstacle a little bigger to create enough space to avoid them and compensate the control errors. This security zone should be avoided as if it was the obstacle.

The second change addresses a modified heuristic for the A* search algorithm that reduces the computing time and finds the optimal search effort level, having in mind the computing time and the optimal path costs. The adjustment is done by setting the correct heuristic parameter **k**. Using equation 6 with **k** = 1 there is the guarantee that the final solution is optimal. Using higher value for k the search space is reduced and the solution found can be suboptimal.When performing a path planning with the original A* method with different **k** values it can be noticed that as **k** increases, the region of possible paths decreases. As a result, it is possible to observe that computing time can be controlled possibly paying the price of having a non optimal path where the length of the path found is extended. In fact, **k** affects processing time and path length. While the first increases, the second decreases. Assuming the cost as a weighted sum of both variables, it can be found an optimized **k**. It will depend on the path type and obstacles. In a static environment it can be meaningless, but in a dynamic one no.

$$h(x, y) = \mathbf{k}\sqrt{(x - x_t)^2 + (y - y_t)^2} \tag{6}$$

So, it is desired to optimize the cost function that depends on two factors: the computing time cost (C_{time}) and the result of the search space cost (C_{ss}). The total cost (C_{TOTAL}) can be described in equation 7 where α and β are weighting constants. Initially, α and β are unitary and can be changed to set the cost weight (time and distance). The computing time cost reflects the price of a late solution in real time task.

$$C_{TOTAL} = \alpha C_{time} + \beta C_{ss} \tag{7}$$

Therefore, in this modified A* algorithm, if it's not in the optimized mode, the search space cost (C_{ss}) is the optimum but it requires a lot of computing time, sometimes impossible to reach in a real time scenario. So it is desired to find a compromise between computing time and the result quality (optimizing the cost function). As a result of simulations, the average total cost can be seen in the Fig. 7. It is possible to obtain the minimum cost for a k = 1.2.

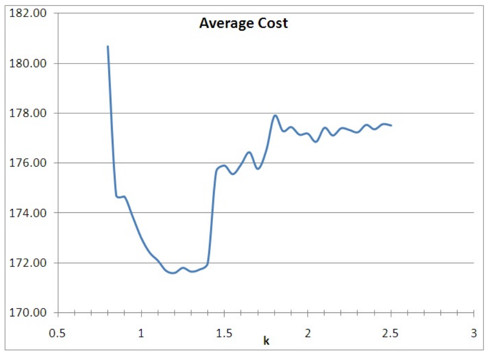


Fig. 7: Average total cost

V. Results

The simulations can be divided in two types of environments: static and dynamic. The first set of simulations has the objective to observe behavior of the A* in a static environment, or in other words, in a structured environment. The second set of simulations has the objective to observe this same behavior in an unstructured highly dynamic environment with mobile obstacles that move "randomly" with speeds that, for some of these obstacles, can be higher that the speed of the robot itself. Finally it's important to mention that all simulations (static and dynamic) started with all objects (robot and obstacles) at the same position.

The Fig. 8 the trajectory calculated by the A* algorithm can be seen. Note that A* builds a path from the robot's initial point to the target point that passes around the obstacles.

For the simulations in the unstructured environment the movement of the obstacles is set to be half random. That's because those movements are set by a simple algorithmic procedure in SimTwo. As it can be seen in the following algorithm, this uploads the values from each sphere and sets a force upon each one, making the movements. With a constant velocity for all the spheres, the movements of each one is practically always the same, except if the robot hits one or more sphere. In Fig. 9, the discrete evolution in A*

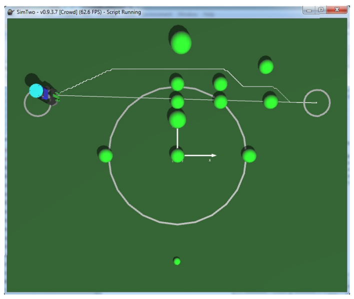


Fig. 8: A* path planed for static environment. The straight line is the distance from the robot to the target and the curved line is the A* generated path.

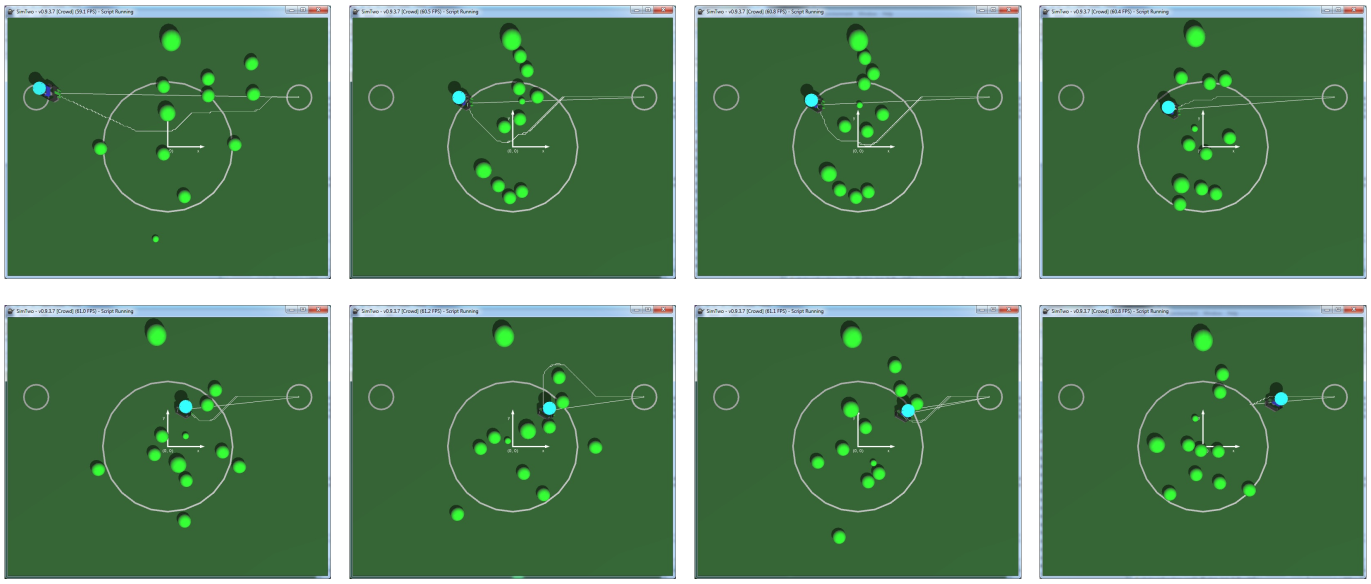


Fig. 9: A* Dynamic Simulation results with discrete instants t=1,2,3,4,5,6,7 and 8

path planner, until reach the target, can be seen. An important observation in the last sub-picture is that the track became static and the robot is choosing not to use the A* algorithm due to the fact that there are no more obstacles around or near it. Therefore, SimTwo prints on screen the last track calculated by the path planner in case.

```
Algorithm Set Sphere Speed
1:  If ∃ Sphere then
2:      $NexPosition_x$=$Initial_x$+Radius·cos(Speed·(t+$\delta$))
3:      $NexPosition_y$=$Initial_y$+Radius·sin(Speed·(t+$\delta$))
4:      Set Force of ith-Sphere with
5:          $V_x$ = 100*($NexPosition_x$ - $Position_x$)
6:          $V_y$ = 100*($NexPosition_y$ - $Position_y$)
7:          $V_z$ = 0
8:      end
9:  end
```

Note now how A* recalculates the path in order to avoid the collision points. As it can be also observed in Table I, the results made by A* are satisfactory, not only in terms of getting to the target point, but on avoiding collisions in a crowded environment that even for humans can be sometimes unavoidable.

TABLE I: Dynamic Obstacles Result

Average Measurements in 30 simulations	A*
Average Time to Reach Target	29.72s
Average Number of Collisions	3.4

The existing collisions are due to the entrapment that occurs to the robot by the moving obstacles. Therefore, the collisions became unavoidable, for the obstacles go towards the robot.

VI. Conclusion

This paper presented a new application of the grid-based path planning algorithm A* using two modifications: slack and sub-optimal K value. The first modification builds a security space around the obstacles making them bigger, while the second allows the A* to calculate a faster path making the processing time smaller. Some simulations were made using a crowded and highly dynamic environment with twelve randomly moving obstacles. The simulations were divided in two types of environments: structured and unstructured. The path planner A* built an entire path around the obstacle. In a static environment the robot passed through the obstacles avoiding them successfully. In a unstructured environment the A* algorithm had to re-plan the path to succeed in reaching the target point while avoiding the obstacles. The path construction made by A* was considerable, not only in terms of getting to the target point, but on avoiding collisions in a crowded environment.

Acknowledgment

The authors would like to thank INESC-Porto for the support in project ROBIS which provided the financial and technical support for this work.

References

[1] D. Alejo, R. Conde, J. Cobano, and A. Ollero, "Multi-UAV collision avoidance with separation assurance under uncertainties," in *2009 IEEE International Conference on Mechatronics*, 2009, pp. 1–6.

[2] X.-c. Lai, S. S. Ge, and A. A. Mamun, "Hierarchical Incremental Path Planning and Motion Planning Considering Accelerations," *IEEE Transaction on Systems, Man, and Cybernetics*, 2007.

[3] P. Costa, A. P. Moreira, and P. Costa, "Real-time path planning using a modified A * algorithm," in *ROBOTICA 2009 - 9th Conference on Mobile Robots and Competitions*, 2009, pp. 141–146.

[4] K. Khantanapoka and K. Chinnasarn, "Pathfinding of 2D & 3D game real-time strategy with depth direction Aâ algorithm for multi-layer," *2009 Eighth International Symposium on Natural Language Processing*, 2009.

[5] T. P. Nascimento, C. C. Paim, and A. L. D. Costa, "Omnidirectional Mobile Robot's Trajectory Tracking Control System: A Multivariable Approach," in *Congresso Brasileiro de Automática*, 2010, pp. 1664–1670.
[6] A. S. Conceição, A. Moreira, and P. Costa, "Practical approach of modeling and parameters estimation for omnidirectional mobile robots," *Mechatronics, IEEE/ASME Transactions on*, pp. 377 –381, 2009.
[7] T. P. Nascimento, A. G. S. Conceição, and A. P. G. M. Moreira, "Omnidirectional Mobile Robot's Multivariable Trajectory Tracking Control: A Robustness Analysis," in *9th Portuguese Conference on Automatic Control*, 2010.
[8] S. Yang, "Real-time torque control of nonholonomic mobile robots with obstacle avoidance," in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*, 2002, pp. 81–86.
[9] K. Pathak and S. K. Agrawal, "An Integrated Path-Planning and Control Approach for Nonholonimic Unycycles Usimg Switched Local Potentials," *IEEE Transactions on Robotics*, 2005.
[10] K. Kurihara, N. Nishiuchi, J. Hasegawa, and K. Masuda, "Mobile Robots Path Planning Method with the Existence of Moving Obstacles," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, 2005, pp. 195–202.
[11] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
[12] P. Tang, "Dynamic obstacle avoidance based on fuzzy inference and transposition principle for soccer robots," in *10th IEEE International Conference on Fuzzy Systems. (Cat. No.01CH37297)*, 2001, pp. 1062–1064.
[13] N. Rahman and A. Jafri, "Two layered behaviour based navigation of a mobile robot in an unstructured environment using fuzzy logic," in *Proceedings of the IEEE Symposium on Emerging Technologies, 2005.*, 2005, pp. 230–235.
[14] H. Li, S. X. Yang, and M. L. Seto, "Neural-Network-Based Path Planning for a Multirobot System With Moving Obstacles," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2009.
[15] H. Qu, S. X. Yang, A. R. Willms, and Z. Yi, "Real-time robot path planning based on a modified pulse-coupled neural network model." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 2009.
[16] P. Bhattacharya and M. Gavrilova, "Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path," *IEEE Robotics & Automation Magazine*, 2008.
[17] M. Peasgood, C. M. Clark, and J. McPhee, "A Complete and Scalable Strategy for Coordinating Multiple Robots Within Roadmaps," *IEEE Transactions on Robotics*, 2008.
[18] J. Yao, C. Lin, X. Xie, A. J. Wang, and C.-C. Hung, "Path Planning for Virtual Human Motion Using Improved A* Star Algorithm," *2010 Seventh International Conference on Information Technology: New Generations*, 2010.
[19] C. Cai and S. Ferrari, "Information-driven sensor path planning by approximate cell decomposition." *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 2009.
[20] D. Ferguson, M. Likhachev, and A. Stentz, "A Guide to Heuristic-based Path Planning," in *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems. International Conference on Automated Planning and Scheduling (ICAPS)*, 2005, pp. 1–10.
[21] J. Bruce and M. Veloso, "Safe Multirobot Navigation Within Dynamics Constraints," *Proceedings of the IEEE*, 2006.
[22] P. J. Costa, "Simtwo," 2010, available from http://paginas.fe.up.pt/ paco/wiki/index.php?n= Main.SimTwo. [Online]. Available: http://paginas.fe.up.pt/~paco/wiki/index.php?n=Main.SimTwo
[23] R. Smith, "Open dynamics engine," 2010, available from http://www.ode.org. [Online]. Available: http://www.ode.org
[24] J. R. Ferreira and A. P. G. M. Moreira, "Non-linear Model Predictive Controller for Trajectory Tracking of an Omni-directional Robot Using a Simplified Model," in *9th Portuguese Conference on Automatic Control*, 2010.

A Mobile Robot Navigation Planning in a Human Populated Environment

Fernando Carreira, J. M. F. Calado and Carlos Cardeira

Abstract— **In recent years, the introduction of mobile robots in populated environments like industry, houses and services, created new challenges for robots, who must consider the emotional reactions shown by humans when faced with unexpected moving robots in their field of view. In particular, it is necessary that path planning algorithms have in consideration the presence of humans and their feelings of safety and comfort. Actually, avoidance of human obstacle should not be based in the same techniques as avoiding classic rigid obstacles. In this paper is described the implementation of a path planner that takes into consideration the localization, orientation and different comfort distances of humans. The robot motion through the generated path planner was simulated in a virtual reality scenario based in CAD and VRML objects. The virtual reality is integrated in the Matlab/Simulink model providing integration of the robot in the environment. This leads to very realistic views of the robot paths allowing a better perception of the motion in the human populated environment.**

I. Introduction

Navigation to a certain goal does not only mean to find and follow the shorter path to there, but essentially to know and decide what is the better way to get there. In fact, finding a path between two places that guarantees safety navigation is not always easy because environments normally have many obstacles that increase the collision risk if the robot does not have the ability to achieve a good-quality path [1].

The introduction of robots in humans daily life [2] creates new challenges that did not exist in industrial environments, where they are usually physically separated from humans [3]. In a populated environment with robots, the accident risk caused by a robot hitting a person is constant and this is one of the most common accidents [4], which must be minimized.

Thus, it is important that robots adopt their behaviour according to the humans activities [5], in order to avoid conflicts, provide courtesy [6], and guarantee a safe, reliable, effective and socially acceptable motion [3].

Manuscript received January 14, 2011. This work was supported by Fundação para a Ciência e a Tecnologia, through IDMEC under LAETA.

F. Carreira is with DEM-Instituto Superior de Engenharia de Lisboa, Polytechnic Institute of Lisbon, R. Conselheiro Emídio Navarro, 1. Lisbon, Portugal. e-mail: fcarreira@ isel.ipl.pt

J.M.F. Calado is with CSI/IDMEC-Instituto Superior de Engenharia de Lisboa, Polytechnic Institute of Lisbon, R. Conselheiro Emídio Navarro, 1. Lisbon, Portugal. e-mail: jcalado@ isel.ipl.pt

C. Cardeira is with CSI/IDMEC-Instituto Superior Técnico, Technical University of Lisbon. Av. Rovisco Pais, 1. Lisbon, Portugal. e-mail: carlos.cardeira@ist.utl.pt

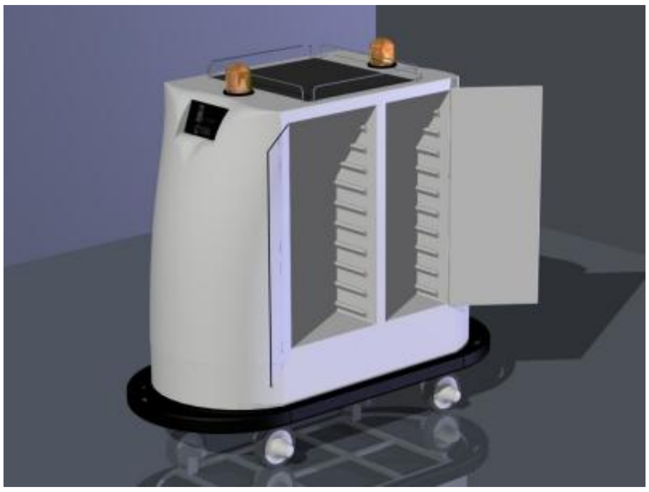


Fig. 1. 3D virtual prototype of i-MERC

For instance, [7] has developed the concept of an automatic vehicle - i-MERC (Fig. 1), in two versions: mobile robot and power-assisted, for the transport of meals, with an omni-directional locomotion system, temperature control and management of food hospitals [8-10] drift.

Considering such a system implemented in populated hospital environments, it is very important that its motion could be friendly, safe and socially acceptable. This problem led the authors to develop a mobile robot planner to a human populated environment combining a potential fields path planning method [11] and the safety issues defined in [3]. The scope of this planner is obtaining a safer and more reliable robot path planning in the presence of humans.

This paper is organized as follows: section two describes the proposed mobile robot planner based on a potential field's method and the details concerned with the human repulsion forces. The third section describes the implemented model and the virtual reality scenario which allows the simulation of the robot movement in a populated virtual environment. The simulation conditions and the different scenarios as well as the results will be described in the fourth section; this section will address several operating conditions that include different initial poses of the robot and humans. The fifth section will present the virtual reality results of the path planning method and the trajectory control simulations when the robot meets humans in different poses. Finally, in the sixth section, some conclusions about the present navigation planning method and a discussion about near future tasks that should be undertaken are presented.

II. Mobile Robot Planning in a Human Populated Environment

Traditionally, path planning with potential fields aims driving the robot through a potential field that gives rise to attractive and repulsive forces created by higher and lower potentials. This path planning methodology represents obstacles as higher potential points that repel the robot being the goal to reach a lower potential location that attracts it.

However, potential fields methods consider that obstacles are all of the same type and the only goal is to avoid the collisions with the obstacles [11]. The path to avoid the obstacle does not take into account different possible reactions that may occur if the obstacle is a human. In our approach, it has been considered that obstacles are humans, which have repulsive forces around them. However, instead common potential fields methods where a repulsive force is uniform around obstacles; the repulsive force in our methodology is based on the human pose, in order to provide a safe and reliable mobile robot path (Fig. 2).

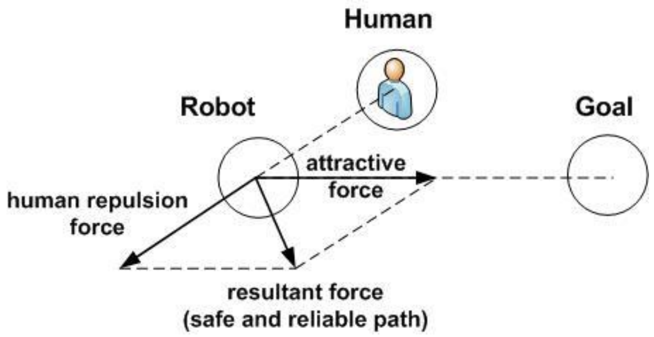


Fig. 2. Potential fields with humans

A. *The Humans Repulsive Forces*

Usually, the path planning based on potential fields considers a repulsive force caused by obstacles, which repel the robot, but the different nature of objects and humans is disregarded.

In the proposed method the repulsion field is not uniform around the humans since they perceive differently their surrounding areas. In [3], the authors identify two important aspects in the definition of a safe and reliable path planning in a robot-human interaction situation: the safety and visibility criterions.

The proposed method borrowed inspiration from these criteria for the definition of a repulsive field around humans. Thus, it has been considered that the human repulsion forces are a function of two repulsive forces, namely “repulsive safety force” and “repulsive visibility force”, which improve the safety and the comfort of the human, respectively.

1) *Repulsive safety force*

Unlike objects, where security distance should be the minimum necessary to prevent accidents, humans need a personal distance to possible harming equipment that assures a feeling of safety. This distance is a function of the personality and culture of each person but, for a public zone, the minimal personal distance will be about 3m [12].

This safety distance should be adapted to humans states: standing up, sitting, etc., since humans normally have less mobility when sitting than when standing up [3].

Thus, a repulsive safety force define by a 3D Gaussian shape around the human which output is a function of xy robot coordinates, as described in the equation 1.

$$SF_R(x,y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma^2}\left((x-h_x)^2+(y-h_y)^2\right)} \tag{1}$$

where h_x and h_y stands for the human position, σ is a parameter given by the following equation, $\sigma = d/3$, being d the human-robot safety distance that should be parameterized according to humans states, culture, age, etc.

In Fig. 3, the repulsive safety field for a personal safety distance of 3m.

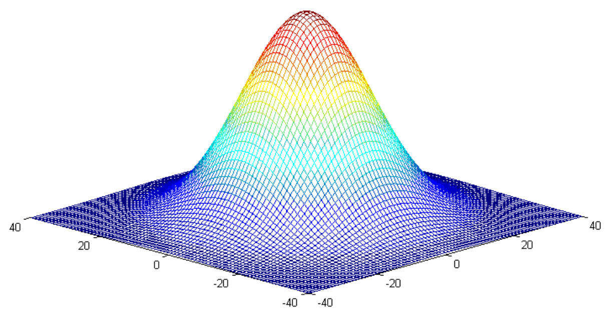


Fig. 3. The repulsive safety field around a human (in cm)

1) *Repulsive visibility force*

However, beyond the safety criterion presented, when humans are in a bustling area with moving machines, normally, they feel more comfortable when the machines are in their field of view. Sisbot *et al.* [3] call this feeling as “mental safety – comfort”. This means that the humans comfort when the robot is in the field of view because they can follow its motion with them own eyes and can react to any endangering movement.

In order to meet this human notion of comfort and safety, a repulsion field based on the human visibility of the robot, has been defined. Once that humans can see (with both eyes) approximately 180 degrees in horizontal [13], following this criterion, the area behind the human is very uncomfortable since humans cannot see the robot and predict its motion. Thus, the proposed methodology consider that the repulsive visibility force is a 3D Gaussian function in the area behind the human and null in the front of him, as it is described in the equation 2,

$$VF_R(x,y) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma^2}\left((x-h_x)^2+(y-h_y)^2\right)}, & \text{if } \theta \in [90;270]^{\text{o}} \\ 0, & \text{if } \theta \notin [90;270]^{\text{o}} \end{cases} \tag{2}$$

where θ is the angle between the direction of the human to the calculation point and the one for which he faces, defining if the point is ahead or behind human:

$$\theta = tan^{-1}\left(\frac{y-h_y}{x-h_x}\right) \quad (3)$$

Like the repulsive safety force, the repulsive visibility force should be adapted to different scenarios, through the safe distance parameter. In Fig. 4 the repulsive visibility field for the same safety distance as in fig 3 (3m), is shown.

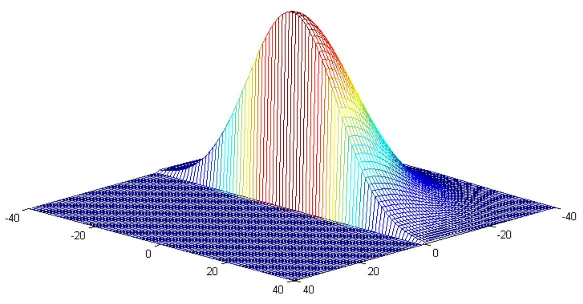


Fig. 4. The repulsive visibility field around a human (in cm)

2) *The human repulsive force*

Once the safety and visibility repulsive forces are available, the human repulsive force is computed as a weighted sum of the both forces [3]. This approach allows, not only to combine the different aspects, but also to assign different relevance to each one, as is represented by equation 4. For example, for different situations or persons, in some cases, it can be given more importance to safety and, in others, to the visibility comfort.

$$F_R(x,y) = w_1 \cdot SF_R(x,y) + w_2 \cdot VF_R(x,y) \quad (4)$$

where *w1* and *w2* are the weighing factors of security and visibility repulsive force.

In Fig. 5, the human repulsive field, considering an equal weighted sum of both repulsive fields, is depicted.

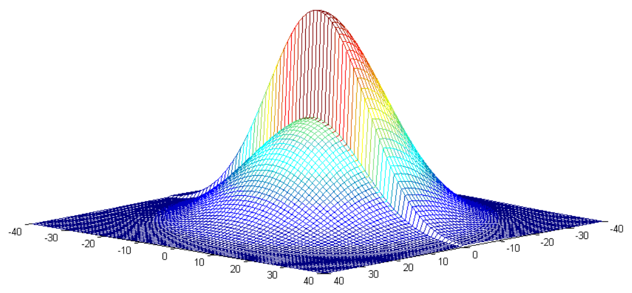


Fig. 5. The human repulsive field (in cm)

3) *The attractive force*

For the attractive field, the method proposed in [11] has been followed, which defines such a field as the inverse of the distance between the robot position and the goal being described by equation 4,

$$F_a(x,y) = \frac{1}{\sqrt{(x-g_x)^2+(y-g_y)^2}} \quad (5)$$

where g_x and g_y are the goal coordinates.

III. Simulations and Results

In order to test the mobile robot navigation planner when the robot passes through humans, different orientations and configuration was performed and analysed. In the simulations, it was expected to see the robot moving towards the goal point while maintaining a safety and comfortable distance from humans when passing near them.

First, it has been conducted tests with the path planner parameterized with a safety distance of 1m without the addition of the visibility criterion. With this configuration, the path planner, with the classical obstacles avoidance (without being human aware), has been simulated. Fig. 6 shows that with two persons near the robot, it has been chosen the shorter path, ignoring the humans comfort distance.

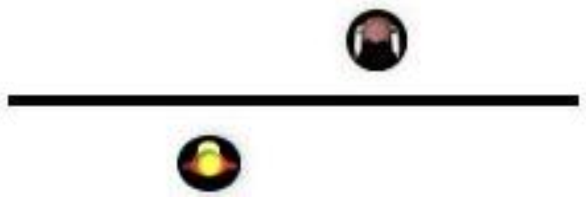


Fig. 6. Robot passes in front of humans (no safety criterion was taken into account)

The second test was concerned with only one person in four different poses: two 0.5m above the start-target line (Fig. 7 and 8) and two 0.5m below it (Fig. 9 and 10).

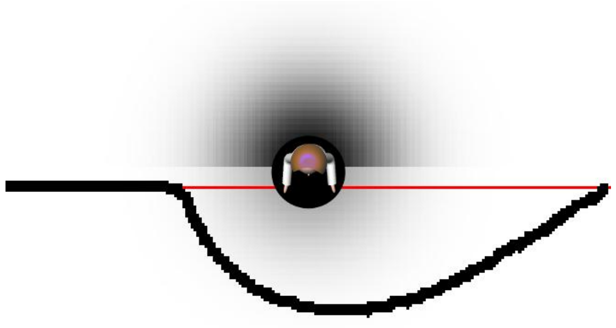


Fig. 7. Robot passes in front of human (short path) when he is above the line (robot find him on its left)

In this test, the robot moves from left to right and the results show that when the robot finds a human on its left avoids him by turning to the right and to the opposite side when it finds him at its right.

However, contrarily to what would happen with the traditional potential fields method, the path is shorter when the human is facing to the robot path (Fig. 7 and 8) than when the robot is folowing a path on the human back (Fig. 8 and 10).

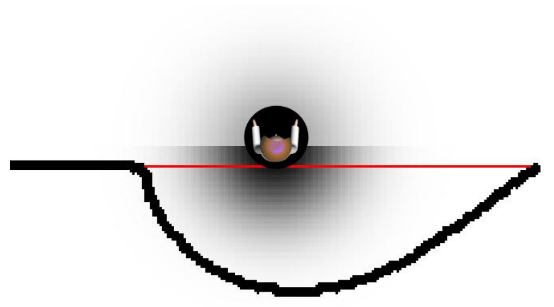


Fig. 8. Robot passes behind of human (longer path) when he is above the line (robot find him on its left)

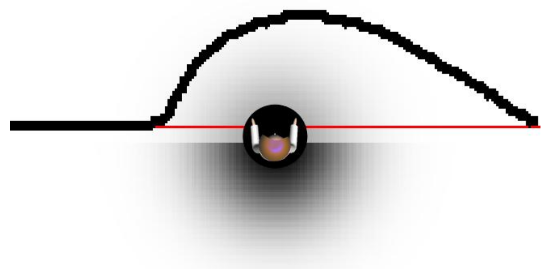


Fig. 9. Robot passes in front of human (short path) when he is below the line (robot find him on its right)

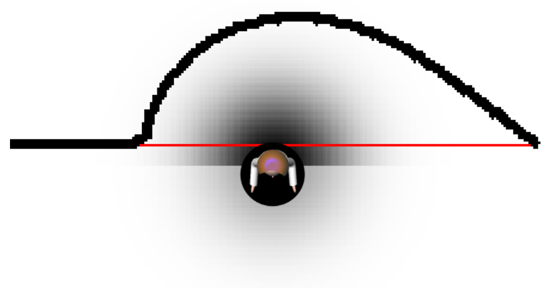


Fig. 10: Robot passes behind of human (long path) when he is below the line (robot find him on its right)

Further tests have been performed considering the path planner with 2 persons facing the start-target line, and different safety parameters (Fig. 11-14).

With 4m of distance and 40% to the safety criterium, the robot avoids both humans leaving more distance when getting close to the second person (Fig. 11).

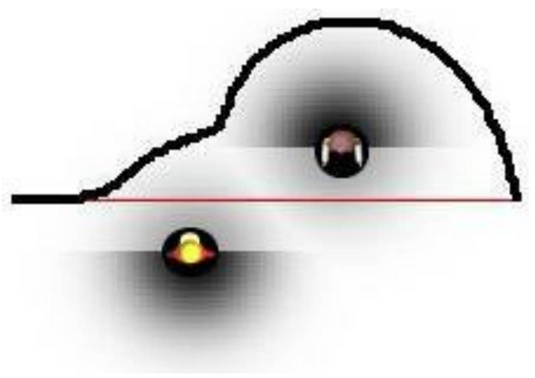


Fig. 11. Robot passes around humans

When the safety distance has been decreased to 3m (Fig. 12) and 2.5m (Fig. 13) the robot began to pass between humans. As expected, the path with 3m of safety distance present a route with a greater bend than with 2.5m.

In the last test, a distance of 4m around humans in the same pose, but without consider the safety criterium, has been considered. As the humans are facing the shorter robot path, the robot follow it, doing the same way than in the fisrt simulation, in other words, doing the same when the traditional obstacle avoidance methods (Fig. 14) are used.

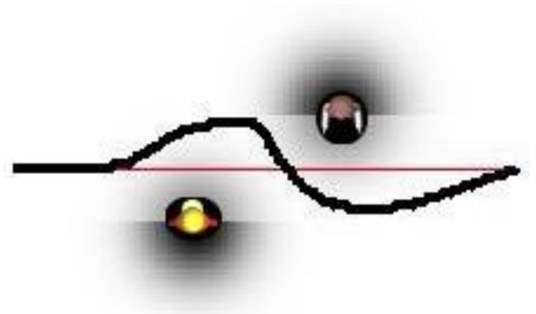


Fig. 12. Robot passes between humans with great bends

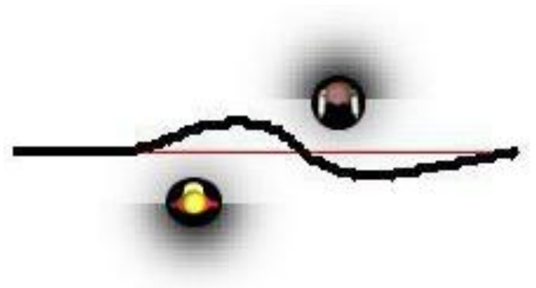


Fig. 13. Robot passes between human with smooth bends

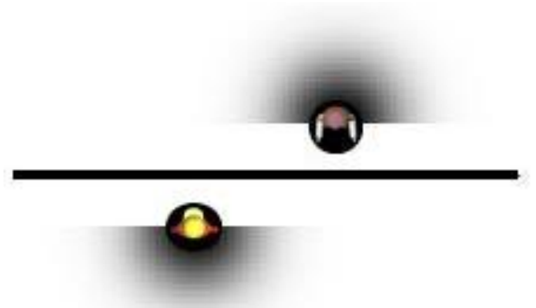


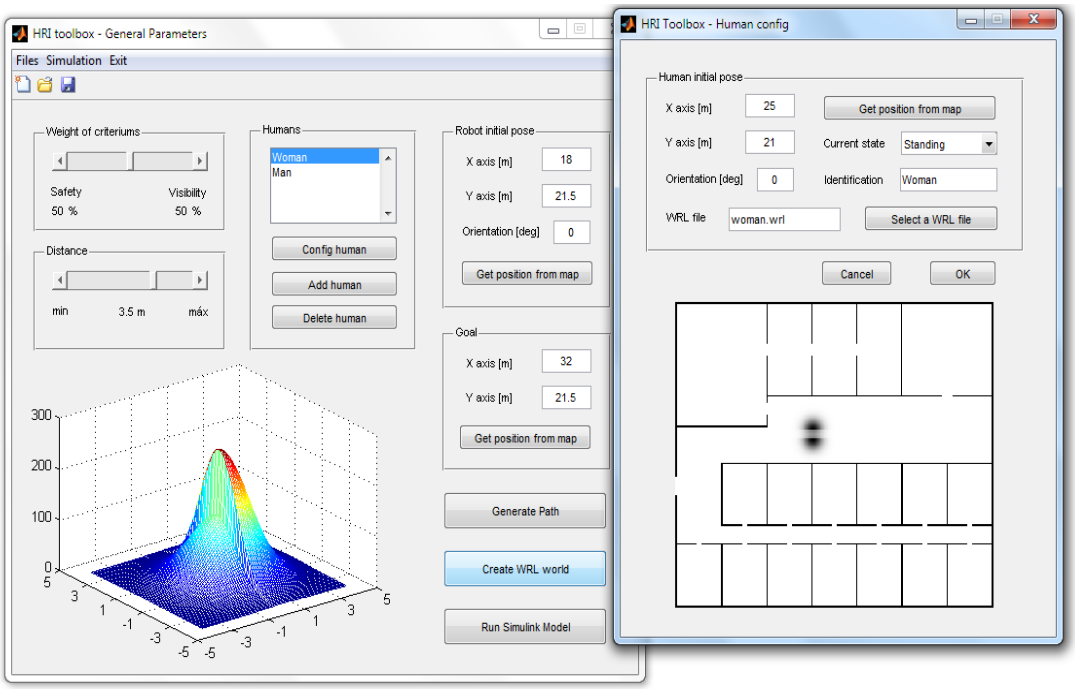
Fig. 14. Robot passes trough he shorter path

IV. Virtual Reality Simulation

To implement the mobile robot navigation planner, we developed a specific tool in MATLAB®/SIMULINK® (Fig.

15) where we can configure the simulation parameters as human and robot pose in map, safety criterions, or analyze the generated path before the simulation.

To increase de reality of the simulation of mobile robot navigation in an environment with humans it has been developed virtual reality scenery where the humans (shape and pose) were included by the developed tool,


HRI toolbox - General Parameters

Weight of criteriums: Safety 50 %, Visibility 50 %

Distance: min 3.5 m máx

Humans: Woman, Man

Robot initial pose: X axis [m] 18; Y axis [m] 21.5; Orientation [deg] 0

Goal: X axis [m] 32; Y axis [m] 21.5

HRI Toolbox - Human config

Human initial pose: X axis [m] 25; Y axis [m] 21; Orientation [deg] 0; Current state Standing; Identification Woman; WRL file woman.wrl

Fig. 15. MATLAB® tool to config the humans

The animation in virtual reality environment aside the simulation were implemented with the blocks of Simulink 3D Animation Toolbox. With this animation, we can have a better perception how i-MERC moves through humans in a hypothetical hospital, but also could be played to show the concept of mobile robot human aware navigation planner's researches or potential customers of robots working in environments with humans.

The robot and the hospital scenery were modelled in Solidworks® (3D CAD software), being the 3D model of the robot, the same that could be used by a design engineers on its concept development or process production. Nevertheless, the human avatars were chosen from the 3D ContentCentral® [14] public dataset. The 3D models were converted to VRML (Virtual Reality Modelling Language) format and several viewpoints were defined in order to assess the global and human level perception of the resulting paths (Fig. 16).

To simulate motion, the coordinates and orientation of robot and humans were parameterized. Thus, the xy translation and the rotation considering z have been used as the inputs to the VRML virtual world.

During simulation, these values are sent from the robot model to the VRML world, through the Simulink 3D Animation Toolbox, thereby producing an animation of the simulated situation.

V. Virtual Reality Results

Simulations were also performed using the Virtual Reality Scenarios providing a better perception of the robot path around humans.

Thus, Fig. 17 shows that in the first (Fig. 6) and last (Fig. 14) simulation, the robot effectively passes very close to humans may causing some discomfort or insecurity. The Fig. 18 shows the simulation when the path planner defines a route around both humans (Fig. 11). The Virtual Reality Scenario shows that this, effectively, is the more comfortable and safe path. However, the robot not always has enough free area to follow this path, requiring a shorter path.

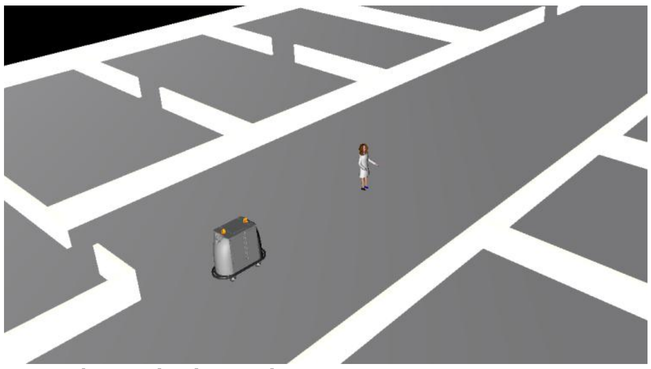


Fig. 16. The virtual reality simulation

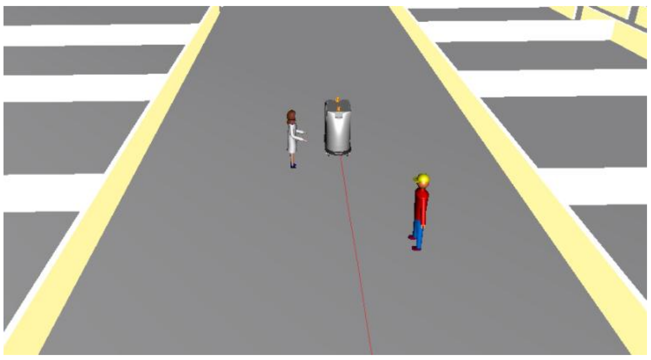


Fig. 17. Robot passes in front of human (Virtual Reality)

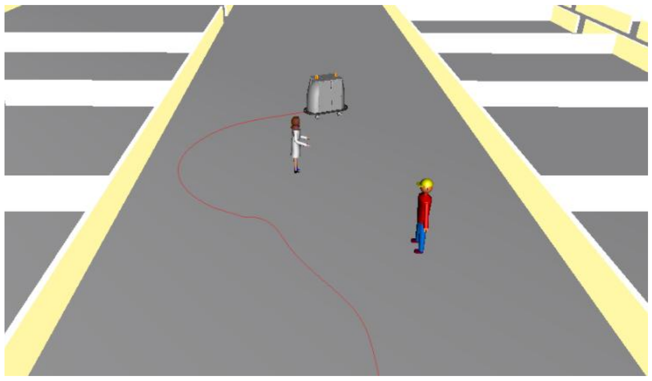


Fig. 18. Robot passes around humans (Virtual Reality)

In fact, when the safety distance has been decreased to 3m the robot followed a shorter path between humans, while keeping a safety distance to them (Fig. 19 and 20).

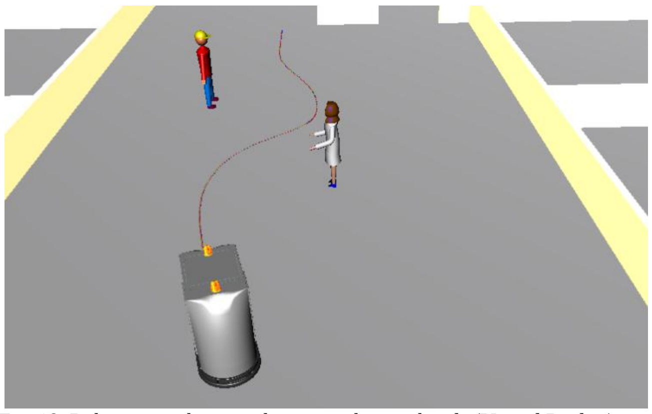


Fig. 19. Robot passes between human with great bends (Virtual Reality)

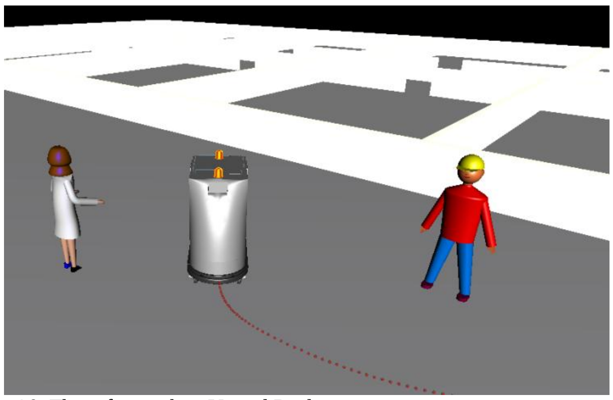


Fig. 20. The safety path in Virtual Reality

A video of virtual reality simulations can be found in http://www.isel.pt/dem/investigacao/moronau/index.html

VI. Conclusions and Future Work

A path planner human aware using the potential fields methods, have been developed. Instead of the traditional potential field's methods, the approach followed in this paper uses a potential field being a function of the human "mental safety" distance and considering the existence of safety and visibility criterion. Thus, the generated path not only depends on the distance to humans but takes into account the humans' orientation.

Furthermore, the developments of a tool enable to configure easily different sceneries and create virtual reality scenarios which allowed a better understanding of the robot movement in a populated environment.

As future work, we will optimize the algorithm trying to eliminate the discontinuities of the function when the robot passes from front to back of a human, but keeping the "mental safety" criterions.

Since the robot has an omni-directional kinematic structure, the corresponding advantages will be explored to increase the comfort of humans and minimize the robots movements along its path.

After these steps, it has been performed simulations considering the motion of humans, creating different and more realistic scenarios. To implement these simulations, an on-line path planner human aware should be implemented becoming the robot sensible for the dynamic of the environment.

References

[1] P. Bhattacharya and M. L. Gavrilova, "Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path," *Robotics & Automation Magazine, IEEE*, vol. 15, pp. 58-66, 2008.

[2] S. Zhao, *et al.*, "Magic cards: a paper tag interface for implicit robot control," 2009, pp. 173-182.

[3] E. A. Sisbot, *et al.*, "A Human Aware Mobile Robot Motion Planner," *Robotics, IEEE Transactions on*, vol. 23, pp. 874-883, 2007.

[4] B. Graf, "Dependability of Mobile Robots in Direct Interaction with Humans," in *Advances in Human-Robot Interaction*. vol. 14, E. Prassler, *et al.*, Eds., Berlin: Springer, 2004, pp. 223-239.

[5] M. Bennewitz, *et al.*, "Learning motion patterns of persons for mobile service robots," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002, pp. 3601-3606 vol.4.

[6] J. Illmann, *et al.*, "Statistical Recognition of Motion Patterns," in *Advances in Human-Robot Interaction*. vol. 14, E. Prassler, *et al.*, Eds., Berlin: Springer, 2004, pp. 69-87.

[7] F. Carreira, "Concepção de Robôs Móveis Aplicados aos Serviços de Saúde," Master, Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa, 2007.

[8] F. Carreira, *et al.*, "i-Merc: A Mobile Robot to Deliver Meals inside Health Services," in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*, 2006, pp. 1-8.

[9] F. Carreira, *et al.*, "i-Merc: um novo conceito para a Segurança e Qualidade da Distribuição de Refeições," *Revista Robótica*, vol. n. 72, 3º trimestre, pp. 4-10, 2008.

[10] F. Carreira, *et al.*, "Veículo Autónomo para Transporte em Segurança de Refeições Hospitalares," Portugal Patent PT 104113, 2008.

[11] P. Vadakkepat, *et al.*, "Evolutionary artificial potential fields and their application in real time robot path planning," 2000, pp. 256-263.

[12] M. L. Walters, *et al.*, "The influence of subjects' personality traits on predicting comfortable human-robot approach distances," Stresa, Italy, 2005, pp. 29-37.

[13] J. Jacobson, *et al.*, "Balance NAVE: a virtual reality facility for research and rehabilitation of balance disorders," 2001, p. 109.

[14] D. ContentCentral. (2008, 29-04-2008). *3D ContentCentral*. Available: http://www.3dcontentcentral.com

Robust automatic landmark detection for underwater SLAM using side-scan sonar imaging

Josep Aulinas, Amir Fazlollahi, Joaquim Salvi, Xavier Lladó, Yvan R. Petillot, Jamil Sawas and Rafael García

***Abstract*— Simultaneous Localization and Mapping (SLAM) consists on building a map of an unknown environment, while simultaneously determining the location of the vehicle within this map. Building a map implies finding a proper representation for its salient features, which are used as landmarks for the localization problem. These landmarks must be very robust in order to be easily detected once reobserved. Associating a new observation with a previously seen landmark provides a proper input for the map and localization update. Instead, wrong associations introduce divergences and inconsistencies in the results. The aim of this paper is to introduce an approach able to detect objects in side-scan sonar images. Side-scan sonar provides high resolution acoustic images, in which an object appears as a bright spot with a dark shadow trail. In order to have a fast and robust object detector, we adapted the framework introduced by Viola and Jones, in which a cascade of classifiers was used to perform a fast face detection with high detection rates. The performance of our detection method is presented, together with the SLAM results obtained after using our robust landmark detector. The results produced high detection rates and small number of false positives, demonstrating the validity of our approach.**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) also known as Concurrent Mapping and Localization (CML) is one of the fundamental challenges of robotics. The goal of SLAM is to build a map of an unknown environment while simultaneously determining the location of the robot within this map [1]. Several methods use features to achieve the SLAM purpose. These features are used as landmarks for the localization problem, and as map elements on the mapping problem. For instance, a point feature based SLAM solution was presented in [2], as shown in Fig. 1. In this specific experiment, a vehicle equipped with a laser-range finder navigates a park full of trees, which are then represented as point features. Other approaches like the one in Fig. 2 propose a line feature based SLAM approach [3]. In this example, an Autonomous Underwater Vehicles (AUV) navigates an abandoned marina, in which line features are the best choice to represent the boundaries between water and land. Finding a proper representation for these features is a key issue to solve a feature based SLAM problem. These features must be very robust, in order to ensure that the same feature will be observed again once revisited. Associating a new observation with a previously seen feature is the key to improve vehicle's localization and the final map. Instead, wrong associations would introduce divergences and inconsistencies in the results, and the consequent loss of the vehicle.

The aim of this paper it to introduce an approach able to

J.Aulinas, J. Salvi, X. Lladó and R. García are with the Computer Vision and Robotics group (ViCoRob) at the University of Girona, 17071, Girona (Spain), e-mails: {jaulinas,qsalvi,llado,rafa}@eia.udg.edu

A. Fazlollahi is with the Le2i - IUT Le Creusot, 71200, Le Creusot (France), e-mail: fazlollahi@gmail.com

Y.R. Petillot and J. Sawas are with the OSL at Heriot Watt University, EH14 4AS, Edinburgh (UK), e-mail: Y.R.Petillot@hw.ac.uk, Jamil.Sawas@hw.ac.uk

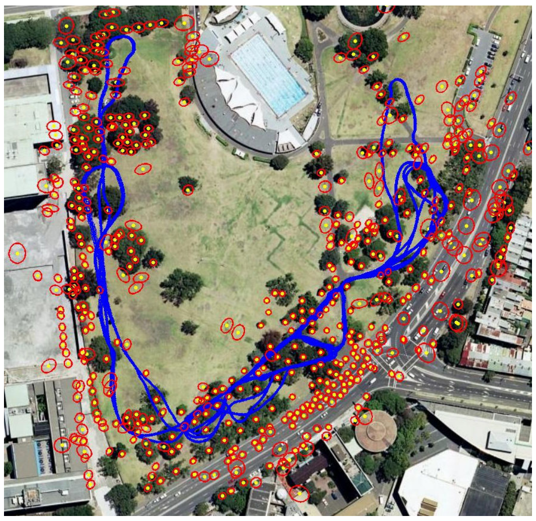


Fig. 1. Example of a point feature based SLAM approach [2]. The image shows the resulting map composed of point features representing trees together with the SLAM trajectory, represented over a satellite image of the scenario.

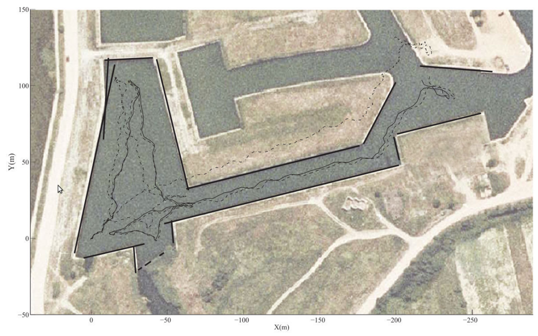


Fig. 2. Example of a line feature based SLAM approach [3]. The image shows the resulting map together with the dead-reckoning (dash-dotted line), GPS (dashed line) and SLAM (solid line) trajectories represented over a satellite image of the scenario.

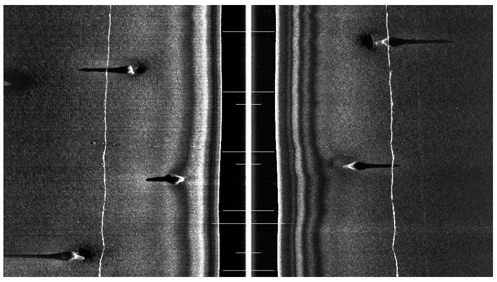


Fig. 3. Side-scan sonar image example, with five objects shown as bright spots and its corresponding shadows.

detect objects in side-scan sonar images on-board of an AUV. Side-scan sonar provides high resolution acoustic images, in which an object appears as a bright spot with a dark shadow trail (see Fig. 3). In order to have a fast and robust object detector, we proposed an approach which follows the Haar cascade framework [4], [5], in which a cascade of boosted classifiers was used to detect objects with high detection rates. Following the same idea to our specific objects, but with a proper feature selection, the detection is significantly improved, providing high detection rates and very low false positives. The performance of our detection method is shown to be a proper input for the SLAM algorithm, providing a consistent map and a correct vehicle localization.

The rest of this paper is organized as follows. Section II briefly summarizes the SLAM approach used. Section III introduces the proposed object detector, describing the steps involved in the process. Section IV summarizes the experimental results obtained. The paper ends with conclusions and future work in Section V.

II. SIMULTANEOUS LOCALIZATION AND MAPPING

The most known consistent Simultaneous Localization and Mapping (SLAM) approach is the Extended Kalman Filter SLAM (EKF-SLAM) [6]. It is based on representing vehicle's pose and the location of a set of environment features in a joint state vector estimated and updated by the Extended Kalman Filter. EKF provides a suboptimal solution due to approximations introduced when linearising the models, which may result in inconsistencies [7], and also due to the assumption that uncertainties associated to the motion and measurement processes are only additional white Gaussian noise. In addition, one of the main drawbacks of EKF implementation is the fact that for long duration missions, the number of landmarks increases and, eventually, computer resources will not suffice to update the map in real-time. This scaling problem arises because each landmark is correlated to all other landmarks, giving a memory complexity of $O(n^2)$ and a time complexity of $O(n^2)$ per step, where n is the total number of features stored in the map. The correlation appears since the observation of a new landmark is obtained with a sensor on-board of the moving vehicle and thus the landmark location is correlated with vehicle's location and other landmarks of the map. This correlation is a key point for the long-term convergence of the algorithm, and needs to be maintained during all the mission.

Using submaps both limitations can be addressed at the same time (i.e. linearisation errors and rise on computational cost). Therefore, dividing the whole scene into submaps, which are then joined in a global map, improves the consistency of the EKF-SLAM [7]. Limiting the size of a submap, by bounding the total number of landmarks or by fixing the maximum distance traveled by a vehicle, maintains the uncertainties of the submap and the linearisation errors small. Another advantage of working with small maps is that the amount of data involved in the EKF-SLAM is small, thus computational cost is reduced. Good examples of submapping strategies are the Local Map Joining [8], the Constant Time SLAM (CTS) [9], the Atlas approach [10], the Divide and Conquer SLAM [11], the Hierarchical SLAM [12] or the Conditional Independent Local Maps (CI) [13]. All of them build submaps first, and then join them following their corresponding strategy.

The approach used in this paper is the Selective Submap Joining SLAM (SSJS). The SSJS was demonstrated to provide good results for terrestrial applications using a well known dataset in [2], and then for underwater applications in [14]. This approach consists of several steps:

- A sequence of local maps is built running a basic EKF SLAM algorithm [8].
- The relative transformation between two consecutive submaps is stored in a relative stochastic global map (similar to the Hierarchical SLAM approach [12]).
- Once a local map reaches its end, the global map is used to generate loop closure hypothesis. Loop hypothesis are created for those submaps that are near the last local map.
- These loop hypothesis are then confirmed by means of data association algorithms, for instance the Joint Compatibility Branch and Bound (JCBB) [15].
- If a loop closure have been accepted, the number of common landmarks between submaps determines whether a fusion and joining steps have to be done.
- Two submaps sharing a high number of features are fused and joined to a single map, producing an update to its local features. Instead, two submaps sharing few features are kept independent.
- Finally, after fusing and joining two submaps, the global level is updated.

As stated in the introduction, a side-scan sonar is used in this work to obtain observations, in a similar way to the implementation presented in [16].

III. OBJECT DETECTION

Side-scan sonar is nowadays widely used in industry and academic research programs to survey the sea floor [17]. Several approaches tackle the issues related to object detection and classification, image segmentation and registration, and feature extraction. For instance, in [19] they proposed a threshold and clustering theory to segment a side-scan sonar image into bright spots, shadows and background. A similar

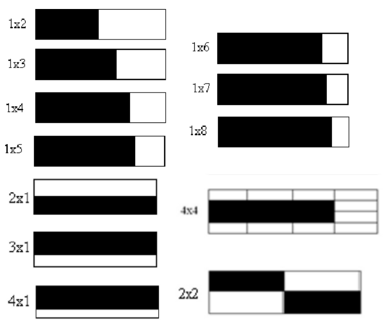


Fig. 4. Haar-like feature examples.

approach was presented in [18]. Another approach uses adaptive threshold techniques to detect and extract geometric features for both bright spots and shadows [20]. In [21] the authors presented an unsupervised model capable of extracting, detecting and classifying shadows automatically. A Markov Random Field (MRF) model was used to segment the image into regions, using the geometric signature of mines in side-scan sonar images. A more recent approach [22], extracts texture features from side-scan images first, and then a region based active contour model is applied to segment objects.

Haar cascade framework has first been presented in [4] for the problem of face detection and provided competitive results. Recently, this framework has been reinvestigated and applied to sonar imagery in [5] and proved to perform very well. This framework can be trained to detect a variety of object classes and it introduces a new algorithm to construct a robust classifier [23]. They use Haar-like features which are really simple (see Fig. 4), fast and cheap to compute using the integral image (see Subsection III-B). Properly selected Haar-like features encode the oriented contrasts between regions in the image and give a quantity for the presence or absence of contrast characteristics at a specific image location.

The following sections describe the principles of our object detector. First, a discussion on the best suited features for our purpose is summarized. Then, basic concepts related to the detection process, such as the integral image and the cascade of classifiers, are introduced. Finally, the training procedure is briefly described.

A. *Feature Selection*

Features are usually more efficient to process than processing the whole intensity image. In side-scan sonar images, objects are simple (see Fig. 5), which means that they can be described only using a few features. Moreover, the performance of the classifier is effected by the number of features. In order to improve this performance, a feature selection is necessary. In this approach, two different types of features have been used: Haar-like features and the distance from the boundary of the object to its centroid. The advantage of such features is their invariance in front of rotations, scale changes, intensity shifts and translations.

In this proposal, rectangular shaped features are used. Although rectangle features are of limited flexibility, they provide a rich image representation and they can be computed using the integral image representation, what makes them extremely computationally efficient. The rectangle is divided into two regions: a dark one in the left, and a bright one in the right. The proportion of each region is different in each feature (see Fig. 5). The value of a two-region feature is the difference between the sum of pixels within the two regions. Fig. 6 represents a set of features, some of them not suited for the problem presented in this work. In contrast, Fig. 7 shows a set of features well suited for our purpose. These plots represent the ability of the system to distinguish between true and false positives. The system used to run these tests is a single weak classifier analysed using a different feature in each case. Each plot is for a specific feature, and shows the value obtained for each training sample after computing the integral image to the corresponding feature. After analysing different feature shapes and proportions, it seems that the most discriminative one is the one with the proportion 8 dark to 1 bright, because the values obtained for the positive set are easily separable from the ones obtained for the negative set.

Another significant feature is the distance from the boundary of the object to its centroid, as in [24]. This feature not only improves a classification stage, but also the object detection.

B. *The Integral Image*

Rectangle features can be computed very rapidly using the integral image (see Fig. 8). The integral image at location x and y contains the sum of pixels above and to left, as shown in the figure. This is a very fast computation, which provides valuable information.

C. *The Cascade Classifier*

A cascade classifier is a sequence of simple classifiers (see Fig. 9 from [23]). The main idea behind a cascade classifier is to detect and reject background information very fast. The initial classifier eliminates a large number of negative examples with very little processing. As the detection goes deeper in the cascade, a higher number of features are used to reject negative objects. Each layer has a strong classifier which is made up of one or more weak classifier. It tries to keep high detection rate in all the layers by decreasing the

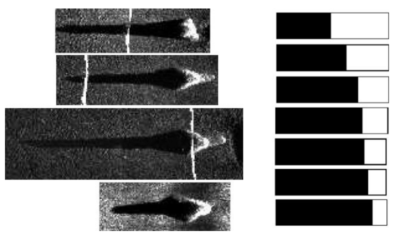


Fig. 5. The left column shows real objects as seen in a side-scan sonar image. The right column shows those Haar-like features best suited to train the sort of real object from the left column.

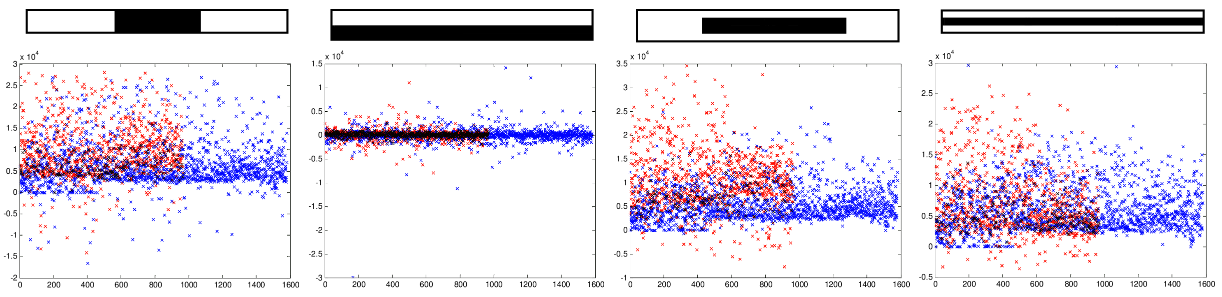


Fig. 6. Comparison of the detection performance using different features not suited for our problem. These features do not distinguish between true and false positives, due to the fact that the orientation they discriminate is not the one from object in side-scan sonar images. In red the integral image results for the positive training sample, and in blue the negative ones.

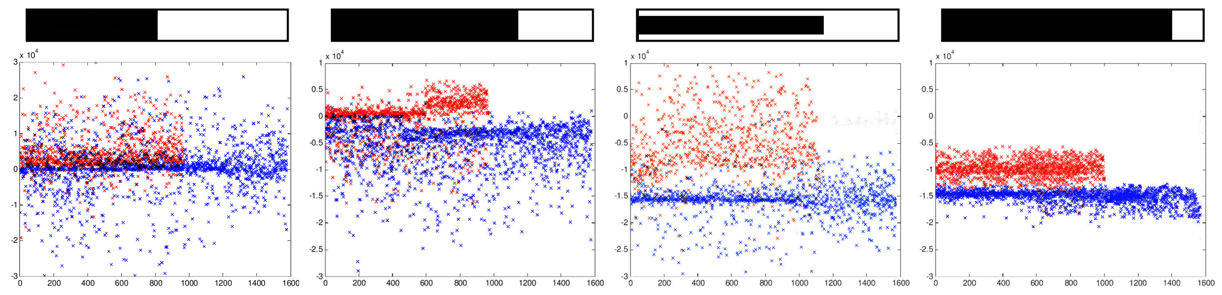


Fig. 7. Comparison of the detection performance using different features well suited for our problem. These features produce proper distinction of true and false positives. They are very similar to real object in side-scan sonar images, but with different proportions.

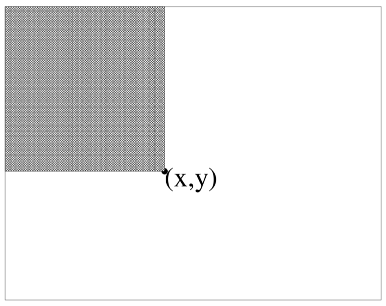


Fig. 8. The value of the integral image is the sum of all the pixels in the white region.

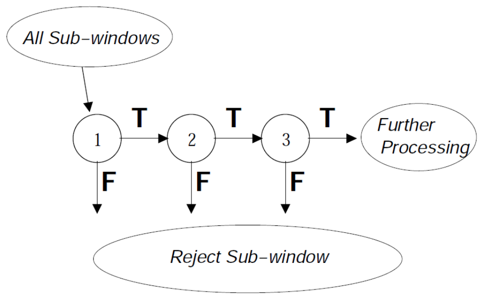


Fig. 9. Schematic depiction of a the detection cascade.

threshold of the strong classifier. Classifiers and their relevant features are selected using AdaBoost [25].

D. Training Stage

At every stage of the cascade detector, more features are added to improve the classification performance. However, using more features means a higher computational cost. For this reason, for the training stage it is necessary to find a trade off between computational cost and detection rates. The optimal solution for a SLAM problem would be to detect all existing objects in the scene, therefore a detection rate of 100% would certainly be optimal. However, forcing the system to detect as many real objects as possible, will introduce false detections. For SLAM algorithm to work properly, this false positive rate must be minimum, because detecting a false object could produce a wrong data association and its consequent inconsistencies in the map and the localization. In addition, on-line SLAM solutions demand for fast detectors, therefore only a few features should be used.

From all these requirements, the main constrains to be met during training process are summarized in Table I. As mentioned in Table I, each layer tries to keep high detection rate while maintaining low false positive rates. In order to achieve this objective, it is necessary to add more features to obtain a stronger classifier. Detection rates are determined by testing the current detector on a validation set. If the overall target false positive rate is not met then another layer is added to the cascade.

IV. EXPERIMENTS AND RESULTS

The experiments were conducted on a real environment dataset. This dataset was acquired with a REMUS-100 AUV (see Fig. 10). The vehicle was equipped with DVL and IMU, providing navigation data relative to the vehicle's reference frame, for instance, velocities, orientations and depth. The AUV was send underwater to perform a recognition mission

TABLE I
TRAINING CONSTRAINTS

Cascade layer	1	2	3	4	...
Detection rate	99.9%	99.7%	99.6%	99.5%	...
False positive rate	50%	20%	10%	5%	...
Number of features	1	2	6	11	...
Cascade layer	...	5	6	7	
Detection rate	...	99.5%	99.3%	99%	
False positive rate	...	1%	0.5%	0.05%	
Number of features	...	20	30	30	

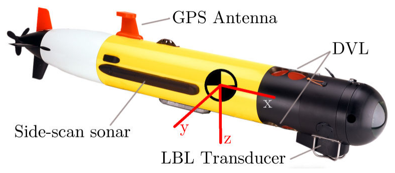


Fig. 10. The REMUS-100 AUV was used to gather the dataset on the experiments.

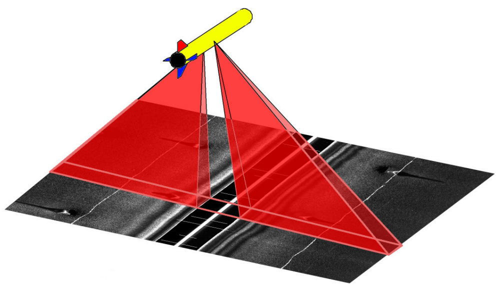


Fig. 12. Side-scan sonar working principle.

(see Fig. 11 for the vehicle trajectory estimated through Long-Baseline sensing (LBL)). During the mission the vehicle navigated a large surface, about 300m x 400m. The whole navigation consisted in a large number of loops, i.e. revisiting the same area several times.

The sea floor was populated with objects, rocks and other salient features. The vehicle was carrying a side-scan sonar pointing both ways, starboard and port (see Fig. 12). This side-scan sonar acquired high resolution acoustic images. The cascade of classifiers trained with the features proposed in this paper was the one in charge of detecting the objects, rocks and other salient features. These features' state is defined as a 3D point in the map. These points are the ones used as inputs for the feature based Selective Submap Joining SLAM, together with DVL and IMU readings (see Fig. 13 for an schematic representation of the whole process).

For the training stage we used a data set of side-scan sonar image patches, some of them containing an object (positive set) and the rest without any object (negative set). According to the behaviour of the classifier, it is recommended to use twice the amount of negative samples than positive ones. In our case, we used 2000 positive objects and 5000 negative objects. Initially the dataset was much smaller, but the performance of the system was not satisfactory. The cascade only improved the false positive rate until a certain layer, there was no further improvement, not even increasing the number of features. For this reason, it was necessary to generate more training images from the original ones, by changing their scale, their intensity and adding noise synthetically. This augmented dataset produced better results. In addition, during the training stage those false object detections, that were significantly affecting the cascade, were removed from the dataset, changing and improving the cascade structure.

Notice that a different set of side-scan sonar images was used to test the performance of the proposed approach, giving the results in Table II. These results show that the obtained cascade is useful at least until its fourth layer, because further layers suffer ans increase in the false negative rate. Until this layer, there is a chance of detecting a false object every hundred observations, which could be a real issue for SLAM. However, the Joint Compatibility Branch and Bound does not take into account spurious observations.

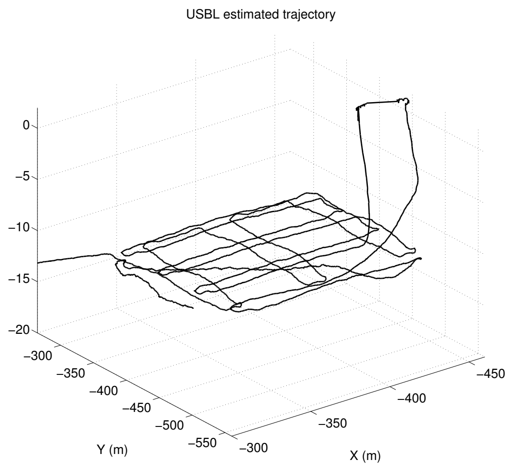


Fig. 11. Estimated trajectory of the vehicle from LBL.

TABLE II
CASCADE PERFORMANCE

Cascade layer	1	2	3	4	...
Detection rate	99.5%	99.3%	99.2%	99%	...
False positive rate	53%	6%	1%	–	...
Number of features	1	2	4	–	...

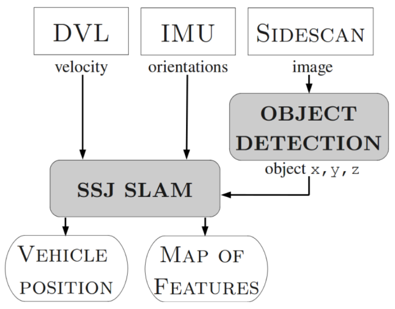


Fig. 13. Block diagram of the system.

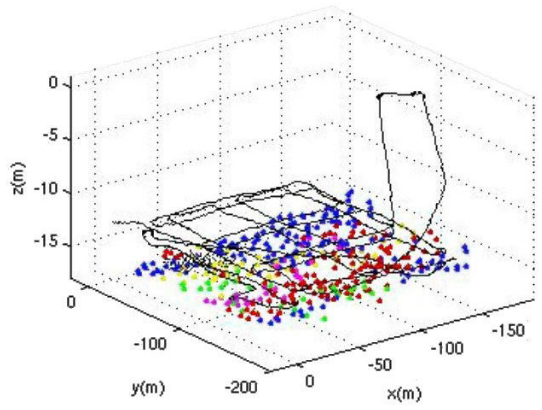


Fig. 14. 3D plot of the map (coloured dots) and the trajectory of the vehicle (black line).

The training stage was very expensive in terms of computational demand, but the detection was very fast. This allowed to check its performance on-line with the SLAM algorithm, providing the qualitative result shown in Fig. 14.

V. DISCUSSION AND FUTURE WORK

In this paper we presented an approach to detect robust features. We conduct a selection of features that improved the detection of objects in side-scan sonar images. The method uses the distance to the centroid and an accurate selection of Haar-like features. This object detection strategy is used to segment objects located on the sea floor observed through side-scan sonar. These objects are then used as observations for a feature based Selective Submap Joining SLAM algorithm. After seeing the results, we can conclude that both the detector and the SLAM algorithm perform satisfactorily. The detector is shown to be fast and provides a high detection rate. After a highly expensive off-line training stage, the detector can perform on-line with the SLAM algorithm, producing large maps consistently and localizing the vehicle within it accurately.

As future work we plan to test the whole system in a real scenario by implementing our approach on the REMUS-100 AUV.

VI. ACKNOWLEDGEMENT

This work was partially funded through the Spanish Ministry of Education and Science (MCINN) under grant CTM2010-15216 and the EU under grant FP7-ICT-2009-248497.

REFERENCES

[1] H. Durrant-Whyte and T. Bailey, ”Simultaneous localization and mapping (SLAM): Part I”, *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp 99–108, 2006.

[2] J. Aulinas, X. LLadó, J. Salvi, Y. Petillot, ”SLAM base Selective Submap Joining for the Victoria Park Dataset”, *7th IFAC Symposium on Intelligent Autonomous Vehicles (IFAC/IAV)*, Lecce(Italy) September 6-8, 2010.

[3] D. Ribas, P. Ridao, J.D. Tards and J. Neira, ”Underwater SLAM in Man Made Structured Environments”, *Journal of Field Robotics*, vol. 25, no. 11–12, pp 898–921, 2008.

[4] P. A. Viola, M. J. Jones, ”Rapid Object Detection using a Boosted Cascade of Simple Features”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 511–518, 2001.

[5] J. Sawas, Y. Petillot, and Y.Pailhas, ”Cascade of boosted classifiers for rapid detection of underwater objects”, *European Conference on Underwater Acoustics*, 2010.

[6] R. Smith, M. Self, and P. Cheeseman, ”A Stochastic map for uncertain spatial relationships”, *Robotics Research, the Fourth Int. Symposium*, O. Faugeras and G.Giralt (Editors), the MIT Press, pp. 467–474, 1988.

[7] J. A. Castellanos, R. Martínez-Cantón, J. Neira, and J. D. Tardós, ”Robocentric map joining: Improving the consistency of EKF-SLAM”, *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 21–29, 2007.

[8] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, ”Robust mapping and localization in indoor environments using sonar data”, *Int. Journal on Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[9] P. M. Newman and J. J. Leonard, ”Consistent, convergent, and constant-time SLAM”, *Int. Joint Conf. on Artificial Intelligence (IJ-CAI)*, pp. 1143–1150, 2003.

[10] M. Bosse, P. M. Newman, J. J. Leonard, and S. Teller, ”SLAM in large scale cyclic environments using the atlas framework”, *Int. Journal in Robotics Research*, vol. 23, no. 12, pp. 1113–1139, 2004.

[11] L. M. Paz, J. D. Tardós, and J. Neira, ”Divide and conquer: EKF SLAM in O(n)”, *IEEE Trans. in Robotics*, vol. 24, no. 5, ISSN 1552–3098, 2008.

[12] C. Estrada, J. Neira, and J. D. Tardós, ”Hierarchical SLAM: real-time accurate mapping of large environments”, *IEEE Trans. Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[13] P. Piniés and J. D. Tardós, ”Large Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision”, *IEEE Transactions on Robotics*, vol. 24, no. 5, October 2008.

[14] J. Aulinas, C.S. Lee, J. Salvi, Y. Petillot, ”Submapping SLAM based on acoustic data from a 6-DOF AUV”, *8th IFAC Conference on Control Applications in Marine Systems (IFAC/CAMS)*, Rostock-Warnemnde (Germany), September 15-17, 2010.

[15] J. Neira and J.D. Tardós, ”Data Association in Stochastic Mapping Using the Joint Compatibility Test”, *IEEE Transaction on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, Dec 2001.

[16] I. Tena-Ruiz, S. Raucourt, Y. Petillot, and D. Lane, Concurrent mapping and localization using side-scan sonar, *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 442–456, 2004.

[17] V. Chandran, S. Elgar and A. Nguyen, ”Detection of Mines in Acoustic Images Using Higher Order Spectral Features”, *IEEE Journals of Oceanic Engineering*, vol. 27, no. 3, pp. 610-618, 2002.

[18] J. Aulinas, X. Lladó, J. Salvi and Y. Petillot, ”Feature based SLAM using Side-Scan salient objects”, *MTS/IEEE Oceans (OCEANS'10)*, Seattle (USA), September 20-23, 2010.

[19] S. G. Johnson and M. A. Deaett, The application of automated recognition techniques to side-scan sonar imagery, *IEEE Journal of Oceanic Engineering*, vol. 19, pp. 138–144, 1994.

[20] C. M. Ciany and J. Huang, Computer aided detection/computer aided classification and data fusion algorithms for automated detection and classification of underwater mine, in *Proc. MTS/IEEE Oceans Conference and Exhibition*, vol. 1, pp. 277–284, 2000.

[21] S. Reed, Y.R. Petillot, J. Bell, ”An Automatic Approach to the Detection and Extraction of Mine Features in Side-scan Sonar”, *IEEE Journal of Oceanic Engineering*, vol. 28, no. 1, pp. 90–105, 2003.

[22] M. Lianantonakis, Y. petillot, ”Side-scan Sonar Segmentation Using Texture Descriptors and Active Contours”, *IEEE Journal of Oceanic Engineering*, vol. 32, pp. 744–752, 2007.

[23] P. Viola and M. Jones, ”Robust Real-Time Face Detection” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[24] L. Atallah , C. Shang , R. Bates, ”Object Detection at Different Resolution in Archaeological Side-scan Sonar Images”, *IEEE Oceans (OCEANS'05)*, Brest (France), June 20-23, 2005.

[25] Y. Freund and R. E. Schapire, ”A decision-theoretic generalization of on-line learning and an application to boosting”, *Computational Learning Theory: Eurocolt 95*, Springer-Verlag, pp. 2337, 1995.

Distance-Weighted Kalman Fusion for Precise Docking Problems

Claus Lenz[1], Thorsten Röder[1], Markus Rickert[2], Alois Knoll[1]

[1]Robotics and Embedded Systems, Department of Informatics, Technische Universität München
[2]Cyber-Physical Systems, fortiss

***Abstract*— This paper proposes a way to solve a highly precise docking problem for a flexible delivery in production environments. The docking problem is seen as one of the fundamental problems to enable more flexible automation using mobile robots. A non-holonomic differential-driven robot with two conveyor belts is used to deliver boxes with goods to two docking slots on an assembly belt and unload them precisely. In order to localize the robot in front of the docking slots, a safety LIDAR and two "minimal invasive" reflecting markers are used that are completely light invariant, thus reaching industrial robustness. This measurement is fused with odometry using a Kalman filter and a distance weighted way to compute the reliability of the data streams.**

I. Introduction

Many industrial facilities already have employed exhaustive methods optimizing productivity in their production labs and construction assemblies [1], [2], [3]. In contrast to traditional optimization methods, the potential of mobile service robotics can be seen as a next revolution with respect to further optimization potentials [4]. This is especially true in conjunction with the increasing demand for flexibility in product and supply management, environmental changes or highly dynamical production flows [5]. As the mobile service robotics can be one solution for further optimizations, the overall performance and robustness is still improvable in real life applications although impressive work was for example done in [6], [7].

A mobile robot requires robust basic capabilities like self-localization, navigation, and closed loop control in order to fulfill given mobility tasks. Besides the overall robustness and reliability of these functionalities, non-functional requirements must also be met for serious industrial usage including smoothness of actuator control, speed, and energy optimizations efficiency, and a safe and predictable behavior of the mobile robot. Fundamental constraints are typically given by choosing a mobile platform of holonomic or non-holonomic type, each with its own advantages and disadvantages.

In this paper, we propose a methodology for solving a highly precise docking problem. The docking problem is one of the fundamental problems such as the handover of goods or the battery charing [8], [9], [10], [11], [12]. Approaches for the docking problem typically use optical, IR [13], or electromagnetic [14], [15] markers. Opposite to the docking in charging positions, the presented application example in this paper requires highly precise approaching. Here, two conveyor belts mounted on a mobile robot are

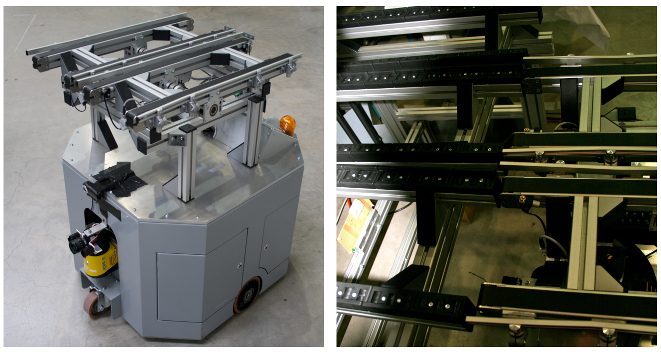

(a) (b)

Fig. 1. (a) The differential-driven platform used in this work. It is equipped with a LIDAR, several cameras and two conveyor belts to load, carry, and unload boxes on assembly line docking station as depicted in (b).

used to deliver boxes with goods to two docking slots on an assembly belt. These boxes need to be precisely pushed from the robot in order to glide into to the docking slot. Using a non-holonomic differential-driven platform, this approach needs to be precomputed exactly and the corresponding path-following needs to be precise as well.

In order to localize the robot in front of the docking slots, we make use of the already installed safety LIDAR and two reflecting markers. This way, almost no invasive markers need to be added and the results are completely light invariant, thus reaching industrial robustness. This measurement is fused with odometry using a Kalman filter [16] and a distance weighted way to compute the reliability of the data streams.

The paper is organized as follows: Sect. II introduces the differential kinematics of the mobile robot, Sect. III describes how a given path can be smoothly followed with such a kinematic, Sect. IV explains the localization and the fusion method in detail and overviews the results.

II. Differential-Driven Mobile Platform

A differential-driven robot—such as the one depicted in Fig. 1—is controlled via two independently actuated wheels on a common axis that have a distance of $2b$ from each other. The robot can move with the linear velocity u and the angular velocity ω along the body axes (Fig. 2). With the assumption that the wheels are perfectly rolling, the kinematic model of such a robot can be expressed as:

$$u = \tfrac{1}{2}(v_r + v_l) \tag{1}$$

$$\omega = \tfrac{1}{2b}(v_r - v_l)\,, \tag{2}$$

where the maximal velocities of the wheels are bound to certain limit values V_m, e.g., due to hardware limits of the motors.

Without any other limitation, such a robot can drive on paths with arbitrary curvatures

$$\kappa = \tfrac{1}{b}\frac{v_r - v_l}{v_r + v_l}\,. \tag{3}$$

The curvature κ of a planar path is related to the radius r_c of a circle that most closely approximates the path at a given point (P):

$$\kappa = \frac{1}{r_{c(P)}}\,. \tag{4}$$

This means, that for counter-rotating motors, the curvature becomes infinite and the robot turns on the spot.

While the robot is driving on the ground plane in a two-dimensional space (x and y), the curvature κ is one-dimensional. Therefore, for arbitrary plane paths that can be expressed parametrically by $(x(t), y(t))$, the signed curvature is given by

$$\kappa = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}}\,. \tag{5}$$

This equation can be reduced to

$$\kappa = \frac{\ddot{y}}{(1 + \dot{y}^2)^{\frac{3}{2}}} \tag{6}$$

$$= \frac{f''(x)}{(1 + f'(x)^2)^{\frac{3}{2}}}\,, \tag{7}$$

under the assumption $\dot{x} = 1$ and $\ddot{x} = 0$.

The curvature [17] is the central parameter of robotic motion when considering maximization of linear velocity and minimization of snatchyness. However, a smooth behavior of a mobile robot without hard brakes is desirable due to multiple issues: From an energy efficiency point of view, hard brakes should be avoided. Additionally, smooth—and therefore predictable—behavior of the robot increases safety, as humans can estimate the trajectory and act accordingly.

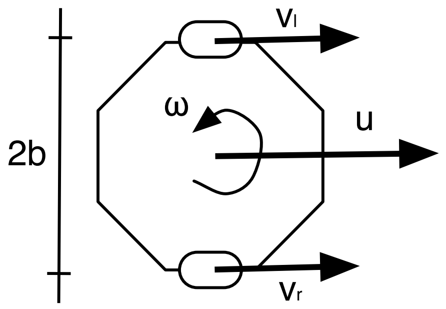


Fig. 2. Differential-driven platforms with a wheelbase of $2b$ are controlled by the turning velocities v_l and v_r of the wheels, resulting in a linear velocity u and an angular velocity ω

Hence, if we want to apply this strategy, the motor velocities should always remain positive

$$v_l, v_r \in [0, V_m]\,, \tag{8}$$

and the curvature of the reference path needs to stay bounded [18] to

$$\kappa = \left[-\frac{1}{b}, \frac{1}{b}\right]. \tag{9}$$

III. Path Following

If we constrain κ, we will get a smoother path trajectory $l \in \mathbb{R}^p$, that can be followed if the robot controller takes special care of a) the distance between the robot and the path and b) the angle between the forward velocity vector and the tangent to the path. Both should be reduced to zero [19] to follow the path.

Following [20], the path following problem can be explained in more detail: Let $P = (x_P, y_P)$ be an arbitrary point on the path l and $Q = (x_Q, y_Q)$ be the center of mass[1] of the differential-driven robot. Along the path, a tangential reference frame $\{F\}$ is attached at every point with a signed curvilinear abscissa denoted with s. This tangential reference system can be referred to as Serret-Frenet frame. Thus, the position of the robot Q can be described in the inertial reference frame $\{I\}$ as

$$\boldsymbol{q}^I = \begin{bmatrix} x_Q & y_Q & 0 \end{bmatrix}^{\mathrm{T}} \tag{10}$$

and in $\{F\}$ as

$$\boldsymbol{q}^F = \boldsymbol{r} \tag{11}$$

$$= \begin{bmatrix} s_{1Q} & y_{1Q} & 0 \end{bmatrix}^{\mathrm{T}}\,. \tag{12}$$

Equivalently, P is given in $\{I\}$ as

$$\boldsymbol{p}^I = \begin{bmatrix} x_P & y_P & 0 \end{bmatrix}^{\mathrm{T}} \tag{13}$$

and in $\{F\}$ always as

$$\boldsymbol{p}^F = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}\,. \tag{14}$$

The rotation from $\{I\}$ to $\{F\}$ is given by $\boldsymbol{R}_{I,F} = \boldsymbol{R}(\theta_c)$, parametrized by the angle θ_c between the inertial frame $\{I\}$ and the curvilinear abscissa s. The reverse rotation is respectively given by $\boldsymbol{R}_{F,I} = \boldsymbol{R}^{-1}(\theta_c)$.

The angular velocity is defined by

$$\omega_c = \dot{\theta}_c \tag{15}$$

$$= \kappa_r(s)\dot{s}\,, \tag{16}$$

with $\kappa_r(s)$ being the curvature of the reference path.

Using these definitions, the velocities of both points Q and P can be easily expressed in both systems.

$$\dot{\boldsymbol{p}}^F = \boldsymbol{R}_{I,F}\,\dot{\boldsymbol{p}}^I \tag{17}$$

$$= \begin{bmatrix} \dot{s} & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{18}$$

The velocity of point Q in $\{I\}$ is given by

$$\dot{\boldsymbol{q}}^I = \begin{bmatrix} \dot{x}_Q & \dot{y}_Q & 0 \end{bmatrix}^{\mathrm{T}} \tag{19}$$

$$= \dot{\boldsymbol{p}}^I + \boldsymbol{R}_{F,I}\,\dot{\boldsymbol{r}} + \boldsymbol{R}_{F,I}\,(\omega_c \times \boldsymbol{r})\,, \tag{20}$$

with $\boldsymbol{r}$ being the vector from P to Q.

[1]for the F5 this is also the center of rotation

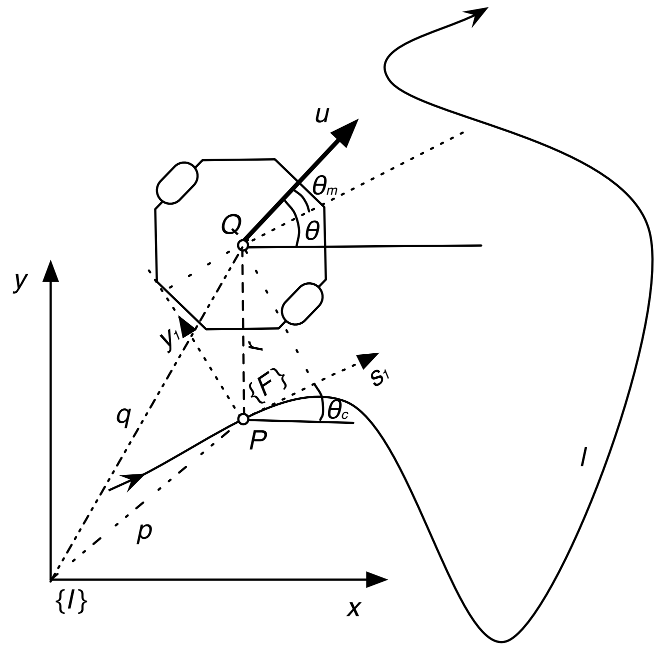


Fig. 3. The mobile robot should follow a given path l. Its position is expressed in an interial frame I and a Frenet frame F rooted in the tangent space of the path. If the angle Θ_m and the distance between robot and path are reduced to zero, the robot moves on the given path

Now, the velocity of Q in $\{I\}$ can be expressed in $\{F\}$ by multiplying both sides with $\boldsymbol{R}_{I,F}$

$$\begin{aligned}\dot{\boldsymbol{q}}^F &= \boldsymbol{R}_{I,F}\,\dot{\boldsymbol{q}}^I \\ &= \dot{\boldsymbol{p}}^F + \dot{\boldsymbol{r}} + (\omega_c \times \boldsymbol{r})\,, \end{aligned} \tag{21}$$

with $\dot{\boldsymbol{r}} = \begin{bmatrix}\dot{s}_{1Q} & \dot{y}_{1Q} & 0\end{bmatrix}^{\mathrm{T}}$.

Considering the relation

$$\omega_c \times \boldsymbol{r} = \begin{bmatrix}0\\0\\\kappa_r(s)\dot{s}\end{bmatrix} \times \begin{bmatrix}s_{1Q}\\y_{1Q}\\0\end{bmatrix} \tag{22}$$

$$= \begin{bmatrix}-\kappa_r(s)\dot{s}\,y_{1Q}\\\kappa_r(s)\dot{s}\,s_{1Q}\\0\end{bmatrix}, \tag{23}$$

equation (21) can be expressed as

$$\dot{\boldsymbol{q}}^F = \begin{bmatrix}\dot{s}\,(1-\kappa_r(s)\,y_{1Q}) + s_{1Q}\\ y_{1Q} + \kappa_r(s)\dot{s}_{1Q}\\0\end{bmatrix} \tag{24}$$

and be solved for

$$\dot{s}_{1Q} = \begin{bmatrix}\cos\theta_c & \sin\theta_c\end{bmatrix}\begin{bmatrix}\dot{x}_Q\\\dot{y}_Q\end{bmatrix} - \dot{s}\,(1-\kappa_r(s)\,y_{1Q}) \tag{25}$$

$$\dot{y}_{1Q} = \begin{bmatrix}-\sin\theta_c & \cos\theta_c\end{bmatrix}\begin{bmatrix}\dot{x}_Q\\\dot{y}_Q\end{bmatrix} - \kappa_r(s)\dot{s}\,s_{1Q}\,. \tag{26}$$

Applying the body-axis speed u (linear velocity), the yaw angle of the vehicle θ_m, and the respective angular velocity $\omega = \omega_m = \dot{\theta}_m$together with the relationship

$$\begin{bmatrix}\dot{x}_Q\\\dot{y}_Q\end{bmatrix} = u\begin{bmatrix}\cos\theta_m\\\sin\theta_m\end{bmatrix} \tag{27}$$

and the mathematical rules for $\theta = \theta_m - \theta_c$

$$\cos\theta = \cos\theta_m\cos\theta_c + \sin\theta_m\sin\theta_c \tag{28}$$

$$\sin\theta = \sin\theta_m\cos\theta_c + \cos\theta_m\sin\theta_c\,, \tag{29}$$

the kinematic model of the unicycle robot can be expressed in $\{F\}$ as

$$\dot{s}_{1Q} = -\dot{s}\,(1-\kappa_r(s)\,y_{1Q}) + u\cos\theta \tag{30}$$

$$\dot{y}_{1Q} = -\kappa_r(s)\dot{s}\,s_{1Q} + u\sin\theta \tag{31}$$

$$\dot{\theta} = \omega_m - \kappa_r(s)\dot{s}\,. \tag{32}$$

Recalling the problem formulation stated previously, a given path is followed exactly if s_{1Q}, y_{1Q}, and θ are zero. On the kinematic level, a locally positive-definite Lyapunov candidate function such as

$$V_1 = \tfrac{1}{2}(s_{1Q}^2 + y_{1Q}^2) + \tfrac{1}{\gamma}(\theta - \delta(y_{1Q},u))^2 \tag{33}$$

can be applied [19] to proof the stability of the system, where it is assumed that

- $\lim_{t\to\infty} u(t) \neq 0$
- $\delta(0,u) = 0$
- $y_{1Q}\,u\,\sin\delta(y_{1Q},u) \leq 0, \forall y\,\forall u\,.$

By choosing

$$\dot{\theta} = \dot{\delta} - \gamma\,y_{1Q}\,u\,\frac{\sin\theta - \sin\delta}{\theta - \delta} - k_2(\theta - \delta) \tag{34}$$

$$\dot{s} = u\cos\theta + k_1\,s_{1Q}\,, \tag{35}$$

with $k_1 > 0$ and $k_2 > 0$ being non-negative constants, the time derivative follows

$$\dot{V}_1 = -k_1\,s_{1Q}^2 - \tfrac{1}{\gamma}(\theta - \delta)^2 + y_{1Q}\,u\,\sin\delta \leq 0 \tag{36}$$

and guarantees that all state variables remain bounded [20]. The function δ has the purpose of shaping the transient convergence of the state to zero and can be chosen according to specific design demands.

Now, (1), (2), (32), and (35) can be merged together, resulting in

$$v_r = u + b\,(\kappa_r\,\dot{s} + \dot{\theta}) \tag{37}$$

$$v_l = u - b\,(\kappa_r\,\dot{s} + \dot{\theta})\,. \tag{38}$$

A further combination of these equations with (3) results in the closed loop curvature of the robot

$$\kappa_{cl} = \frac{\kappa_r\,\dot{s} + \dot{\theta}}{u}\,, \tag{39}$$

for which can be shown [18] that

$$\lim_{t\to\infty}\kappa_{cl} = \kappa_r \tag{40}$$

if the path-following error tends to zero. Please refer to Fig. 4 for an example where the closed loop curvature converges to the reference curvature as the robot follows the path.

Keeping this in mind, the proposed strategy for keeping the robot on a given path becomes

$$u = \begin{cases}\frac{1}{2}V_m & \text{, if } V_1 \geq \epsilon \\ \frac{1}{1+b\,|\kappa_r(s)|}V_m & \text{, if } V_1 < \epsilon\end{cases} \tag{41}$$

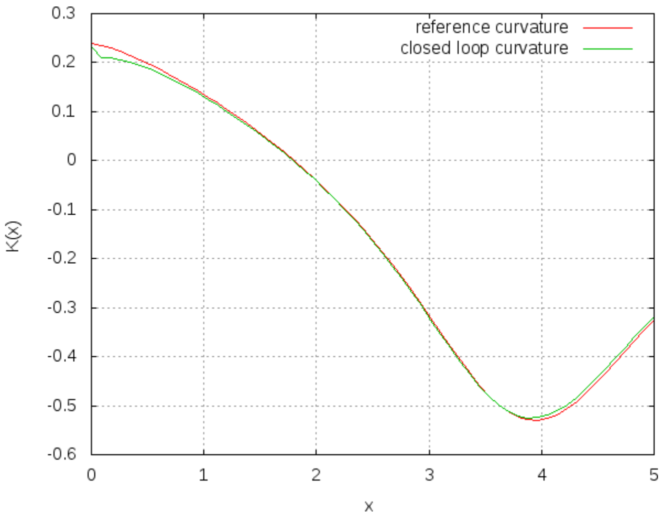


Fig. 4. Example path where the closed loop curvature converges to the reference curvature as the robot follows the path

for the linear velocity with $\epsilon > 0$ being a threshold value regulating the path following error and

$$\omega = \dot{\theta} + \kappa_r(s)\dot{s} \tag{42}$$

for the angular velocity.

IV. Docking Problem

To follow a given path, it is necessary to localize the robot and to keep track of its position to minimize the path-following error. We make use of the safety LIDAR that is already attached to the robot and detect two reflecting markers (left and right) in the polar coordinate frame of the LIDAR: $P_L^{\text{pol}} = (r_L, \theta_L)$, and $P_R^{\text{pol}} = (r_R, \theta_R)$. Please refer to Fig. 5 for a schematic view. These two points are projected to the Cartesian coordinate frame of the robot:

$$P_L = (r_L \cos\theta_L, r_L \sin\theta_L) \tag{43}$$
$$P_R = (r_R \cos\theta_R, r_R \sin\theta_R)\,. \tag{44}$$

With these points, we can compute

$$\beta = \cos^{-1}\left(\frac{|P_M - P_R|^2 + |P_M|^2 - |P_R|^2}{2\,|P_M - P_R|\,|P_M|}\right), \tag{45}$$

which can directly be used to express the angular position of the robot

$$\alpha = \tan^{-1}\left(\frac{P_{L,x} - P_{R,x}}{P_{L,y} - P_{R,y}}\right). \tag{46}$$

The point between these two points in the middle of the docking slot

$$P_M = \tfrac{1}{2}(P_L + P_R) \tag{47}$$

is the center of the coordinate frame in which we want to estimate the robot's position. If we use this as center for a polar coordinate frame, we can interpret β as the angle and $d_m = |P_M|$ as the length. This can also be transformed to a Cartesian coordinate frame which results in

$$P_{\text{robot}} = (d_m \cos\beta, d_m \sin\beta)\,. \tag{48}$$

Hence, the final result of the measurement is given by

$$\hat{p}_l = (P_{\text{robot},x}, P_{\text{robot},y}, \alpha)^{\mathrm{T}} \tag{49}$$
$$= (\hat{x}_l, \hat{y}_l, \hat{\alpha}_l)^{\mathrm{T}}\,. \tag{50}$$

Additionally, the odometry data is initialized with the first measurement of the LIDAR and then propagated ($\hat{p}_o = (\hat{x}_o, \hat{y}_o, \hat{\alpha}_o)^{\mathrm{T}}$) using the wheel-diameter and the kinematic model.

Because the measuring directly returns the position and orientation of the robot with respect to the goal position, we can work in this space and comply with the linearity and Gaussian requirements of a Kalman filter [21]. The robot dynamics and the measurement model are given by

$$\boldsymbol{x}_t = (x, y, \alpha, \dot{x}, \dot{y}, \dot{\alpha})^{\mathrm{T}} \tag{51}$$
$$= \boldsymbol{F}_t \mathbf{x}_{t-1} + \boldsymbol{B}_t w_t \tag{52}$$

and

$$z_t = (\hat{x}_l, \hat{y}_l, \hat{\alpha}_l, \hat{x}_o, \hat{y}_o, \hat{\alpha}_o)^{\mathrm{T}} \tag{53}$$
$$= \boldsymbol{H}_t \boldsymbol{x}_t + \boldsymbol{C}_t \boldsymbol{v}_t\,. \tag{54}$$

w_t and v_t are zero-mean, white Gaussian noise variables, with covariance matrices $\boldsymbol{W}_t, \boldsymbol{V}_t$ respectively. We are using a constant velocity model with white noise acceleration (WNA)

$$\boldsymbol{F}_t = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{I}_{3\times3}\delta t \\ 0 & \boldsymbol{I}_{3\times3} \end{bmatrix}, \quad \boldsymbol{B}_t = \begin{bmatrix} \frac{1}{2}B_0\delta t^2 \\ B_0\delta t \end{bmatrix} \tag{55}$$

and

$$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{I}_{3\times3} \\ \boldsymbol{I}_{3\times3} \end{bmatrix} \tag{56}$$

for the measurement Jacobian.

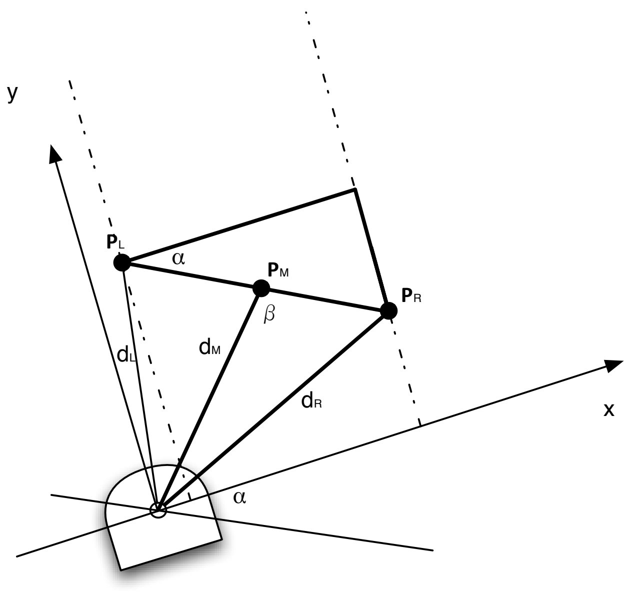


Fig. 5. Schematic view to geometrically solve the position and angle of the mobile robot using a LIDAR and two markers

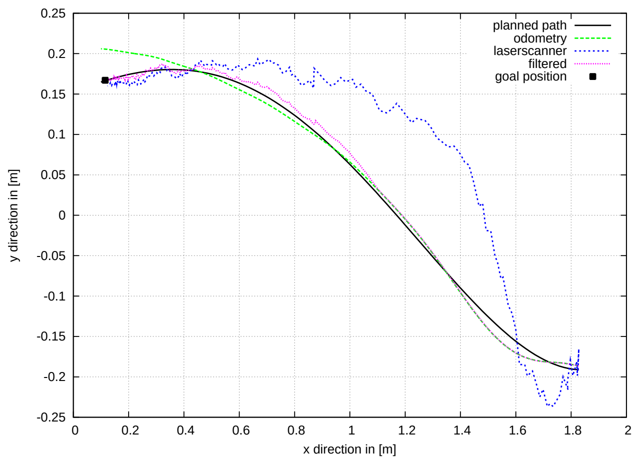


Fig. 6. Docking of the robot to the right assembly line position. The purple line is the fusion of the odometry data (green) and the LIDAR measurements (blue)

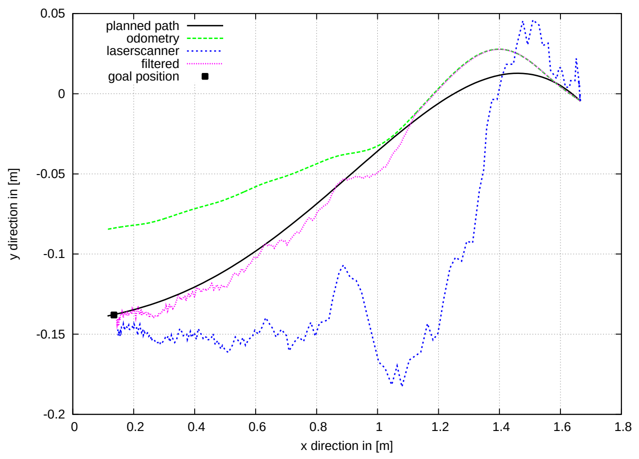


Fig. 7. Docking of the robot to the left assembly line position. The purple line is the fusion of the odometry data (green) and the LIDAR measurements (blue)

The overall noise covariance matrices

$$\boldsymbol{Q}_t = \boldsymbol{B}_t \boldsymbol{W}_t \boldsymbol{B}_t^{\mathrm{T}} \tag{57}$$
$$\boldsymbol{R}_t = \boldsymbol{C}_t \boldsymbol{V}_t \boldsymbol{C}_t^{\mathrm{T}} , \tag{58}$$

with $\boldsymbol{R}$ being a diagonal matrix, realize the optimal Bayesian predictor-corrector scheme by

$$\boldsymbol{x}_t^- = \boldsymbol{F}_t \mathbf{x}_{t-1} \tag{59}$$
$$\boldsymbol{P}_t^- = \boldsymbol{F}_t \boldsymbol{P}_{t-1} \boldsymbol{F}_t^{\mathrm{T}} + \boldsymbol{Q}_t \tag{60}$$

and

$$\boldsymbol{x}_t = \boldsymbol{x}_t^- + \boldsymbol{K}_t \left(z_t - \boldsymbol{H}_t \boldsymbol{x}_t^- \right) \tag{61}$$
$$\boldsymbol{P}_t = \left(\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t \right) \boldsymbol{P}_t^- , \tag{62}$$

with the Kalman gain

$$\boldsymbol{K}_t = \boldsymbol{P}_t^- \boldsymbol{H}_t^{\mathrm{T}} \left(\boldsymbol{H}_t \boldsymbol{P}_t^- \boldsymbol{H}_t^{\mathrm{T}} + \boldsymbol{R}_t \right)^{-1} . \tag{63}$$

The two modalities used (LIDAR and odometry) show different behaviors with respect to the distance to the goal: the odometry delivers smooth results, but if the initialization or the initial localization was not done properly, the error will be accumulated with the inherent drift in odometry data leading to an incorrect behavior. Opposite to that, the LIDAR localization method measures with a lot of noise, which reduces as the robot approaches the goal. Therefore, we use the noise covariance matrix to control the trust in the two signals by integrating the distance related variable d_t in order to update the matrix

$$\boldsymbol{R}_t(d_t) = \mathrm{diag} \begin{bmatrix} \frac{1}{((1-a)+a\,(d_{\max}-\frac{d_t}{d_{\max}}))\,r_l} \\ \frac{1}{((1-a)+a\,(d_{\max}-\frac{d_t}{d_{\max}}))\,r_l} \\ \frac{1}{((1-a)+a\,(d_{\max}-\frac{d_t}{d_{\max}}))\,r_l} \\ \frac{1}{((1-b)+b\,\frac{d_t}{d_{\max}})\,r_o} \\ \frac{1}{((1-b)+b\,\frac{d_t}{d_{\max}})\,r_o} \\ \frac{1}{((1-b)+b\,\frac{d_t}{d_{\max}})\,r_o} \end{bmatrix} , \text{ if } d_t < d_{\max} \tag{64}$$

in each time step, with r_l and r_o being the initial inverse covariance values, a and b being splitting factor to guarantee a constant base value, and d_{max} representing the threshold when to apply the distance weighting of the matrix. If $d_t > d_{max}$, the initial values r_l and r_o are applied.

As it can be seen in Fig. 6 and Fig. 7, the proposed method was was successfully applied in order to perform a docking maneuver. From an initial position in about $1\,6\mathrm{m}$ from the actual goal position, the robot was approaching to the two delivery positions of the docking station via a computed fourth-order polynomial trajectory. The robot is moving from right to left. With a predominance on the smooth odometry data in the beginning, the more and more accurate LIDAR measurement gets integrated. Throughout the approach, the planned path is being followed by the robot. Screenshots from the docking maneuver are depicted in Fig. 8.

V. Conclusions

In this paper, we have shown a way to precisely dock to an assembly line with a non-holonomic differential-driven mobile platform. The platform was equipped with two conveyor belts in order to carry boxes and unload them to a docking slot on an assembly line. To drive smoothly, the motion space was constraint to only positive motor velocities. The path to approach the goal position was computed using a fourth-order polynomial equation and followed using Frenet frame representations. To localize the robot, the safety LIDAR was used to detect two reflecting markers. This measurement was fused with a Kalman filter to determine the current position of the robot, that was used to control the robot on the path. The two modalities from odometry and the LIDAR were fused using a distance weighted way in order to compute the overall measurement noise covariance matrix that was used to compute the Kalman gain. The results in Sect. IV showed that a precise docking was possible and the loading and unloading could be performed successfully.

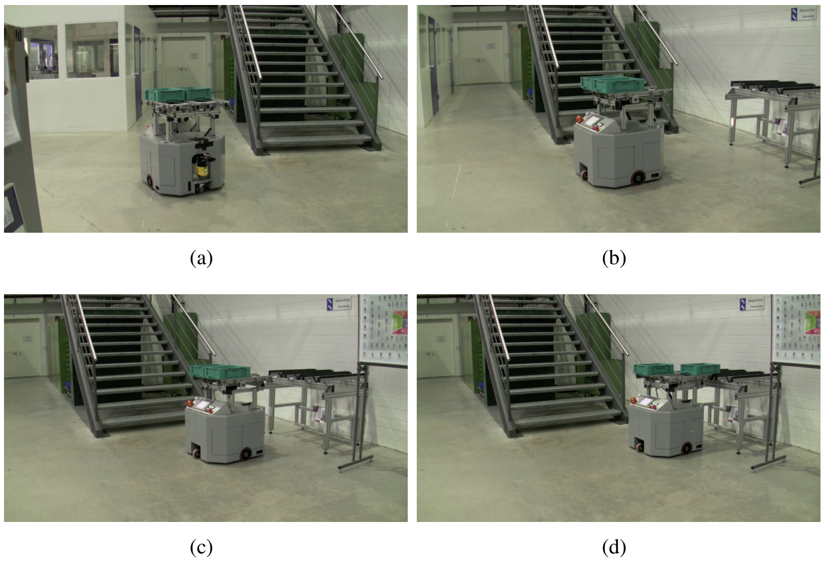


Fig. 8. Application scenario – The mobile platform F5 is approaching a docking station and delivers two boxes with high precision

VI. Acknowledgments

This work was partially supported by the DFG excellence initiative research cluster *CoTeSys (Cognition for Technical Systems)*[2].

References

[1] R. Arkin, R. Murphy, M. Pearson, and D. Vaughn, "Mobile robot docking operations in a manufacturing environment: Progress in visual perceptual strategies," in *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 1989, pp. 147–154.

[2] H. Hu and D. Gu, "Generalised predictive control of an industrial mobile robot," in *Proceedings of the IASTED International Conference on Intelligent Systems and Control*, 1999, pp. 235–240.

[3] G. Alenya, J. Escoda, A. Martinez, and C. Torras, "Using laser and vision to locate a robot in an industrial environment: A practical experience," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 3528–3533.

[4] A. Stopp, T. Baldauf, S. Horstmann, and S. Kristensen, "Dynamic work space surveillance for mobile robot assistants," in *Proceedings of the IEEE International Workshop on Robot and Human interactive Communication*, 2003, pp. 25–30.

[5] M. Zäh, M. Beetz, K. Shea, G. Reinhart, K. Bender, C. Lau, M. Ostgathe, W. Vogl, M. Wiesbeck, M. Engelhard, C. Ertelt, T. Rühr, and M. Friedrich, *Changeable and Reconfigurable Manufacturing Systems*. Springer, 2009, ch. The Cognitive Factory, pp. 355–371.

[6] Y. Yang and O. Brock, "Elastic roadmaps–motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[7] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, J. Hsu, R. B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. P. Gerkey, and E. Berger, "Autonomous door opening and plugging in with a personal robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010, pp. 729–736.

[8] M. Silverman, B. Nies, D.and Jung, and G. Sukhatme, "Staying alive: A docking station for autonomous robot recharging," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 2002, pp. 1050–1055.

[9] R. Cassinis, F. Tampalini, P. Bartolini, and R. Fedrigotti, "Docking and charging system for autonomous mobile robots," University of Brescia, Tech. Rep. R.T.2005-02-4, 2005.

[10] R. Luo, C. Liao, K. Su, and K. Lin, "Automatic docking and recharging system for autonomous security robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2953–2958.

[11] R. Luo, C. Liao, and S. Lin, "Multi-sensor fusion for reduced uncertainty in autonomous mobile robot docking and recharging," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2203–2208.

[12] C. McCarthy, N. Barnes, and R. Mahony, "A robust docking strategy for a mobile robot using flow field divergence," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 832–842, 2008.

[13] S. gon Roh, J. H. Park, Y. K. Song, K. Yang, M. Choi, H.-S. Kim, H. Lee, and H. R. Choi, "Flexible docking mechanism using combination of magnetic force with error-compensation capability," in *Proceedings of the IEEE International Conference on Automation Science and Engineering*, 2008, pp. 697–702.

[14] Y.-C. Wu, M.-C. Teng, and Y.-J. Tsai;, "Robot docking station for automatic battery exchanging and charging," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2009, pp. 1043–1046.

[15] S. gon Roh, J. H. Park, Y. H. Lee, Y. K. Song, K. W. Yang, M. Choi, H.-S. Kim, H. Lee, and H. R. Choi, "Flexible docking mechanism with error-compensation capability for auto recharging system of mobile robot," *International Journal of Control, Automation, and Systems*, vol. 6, no. 5, pp. 731–739, 2008.

[16] S. Shoval, I. Zeitoun, and E. Lenz, "Implementation of a Kalman filter in positioning for autonomous vehicles, and its sensitivity to the process parameters," *The International Journal of Advanced Manufacturing Technology*, vol. 13, no. 10, pp. 738–746, 1997.

[17] A. Balluchi, P. Souères, and A. Bicchi, "Hybrid feedback control for path tracking by a bounded-curvature vehicle," in *Proceedings of the International Workshop on Hybrid Systems: Computation and Control*, 2001, pp. 133–146.

[18] G. Indiveri, A. Nüchter, and K. Lingemann, "High speed differential drive mobile robot path following control with bounded wheel speed commands," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 2202–2207.

[19] A. Micaelli and C. Samson, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," INRIA, Research Report RR-2097, 1993.

[20] D. Soetanto, L. Lapierre, and A. Pascoal, "Adaptive, non-singular path-following control of dynamic wheeled robots," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 2, 2003, pp. 1765–1770.

[21] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[2] `http://www.cotesys.org/`

Active Object Recognition by Offline Solving of POMDPs

Susana Brandão [#,+1], Manuela Veloso [*2], João Paulo Costeira [+3]

[#]*ECE,* [*]*CS Carnegie Mellon University*
5000 Forbes Avenue, Pittsburgh USA,
[+] *ISR, IST-UTL,*
Av. Rovisco Pais 1, 1049-001 Lisboa Portugal
[1]sbrandao@andrew.cmu.edu
[2]mmv@cs.cmu.edu
[3]jpc@isr.ist.utl.pt

Abstract— **In this paper, we address the problem of recognizing multiple known objects under partial views and occlusion. We consider the situation in which the view of the camera can be controlled in the sense of an active perception planning problem. One common approach consists of formulating such active object recognition in terms of information theory, namely to select actions that maximize the expected value of the observation in terms of the recognition belief. Instead in our work we formulate the active perception planning as a Partially Observable Markov Decision Process (POMDP) with reward solely associated with minimization of the recognition time. The returned policy should be the same as the one obtained using the information value. By recognizing observations as a time consuming process and imposing constrains on time, we minimize the number of observations and consequently maximize the value of each one for the recognition task. Separating the reward from the belief in the POMDP enables solving the planning problem offline and the recognition process itself becomes less computational intensive. In a focused simulation example we illustrate that the policy is optimal in the sense that it performs the minimum number of actions and observation required to achieve recognition.**

I. Introduction

Object recognition is still an open problem. From the choice of features to the actual classification problem, we are still far from a global recipe that would allow for a complete discriminative approach to recognition. The large majority of object recognition community is focused on offline, database driven tasks. State of the art is measured with respect to performance in datasets gathered from web images such as the Pascal challenge datasets. Two problems arise from the use of such datasets. The first is the large variability of images. The second is the incapacity to look at the scenes in images from different poses, which would provide not only different, and probably more discriminative, views of objects as would help to segment objects from the background.

In the context of a robot moving in a constrained environment, the object variability is no longer present. The chairs in an office building are all very similar to each other and will be the same for long periods of time. For a robot moving in such a building, the model for a chair can be much simpler and efficient than a model built from web datasets. So, in this project, we assume that recognition can be feasible in such an environment.

However, in spite of having highly accurate models of each object in a room, the robot may not be able to completely distinguish between two different objects. Both self-occlusion and occlusion caused by other objects may cover the distinctive parts of an object, making the robot inable to distinguish between two object classes: the object classes are ambiguous given the occlusion. This ambiguity appears, for example, between a computer screen and a card box. Although they may look the same when the robot is directly in front of them, if the robot looks to the side of the screen it should be able to correctly differentiate the screen from the card box. Since the robot will never have access to all the views of the object at a given time instant, the type of ambiguity described arises even when the robot performs 3D object recognition. The robot only has access to partial information on the object until it decides to move with relation to the object.

We assume that most of the ambiguity in object recognition can be removed by having the robot looking to objects through different angles. I.e., we assume that, in spite the ambiguity between object A and object B, there is always an angle in A or B from which the objects can be disambiguated.

There is a vast literature on active perception and the reader may find a detailed overview of the field with special focus on multi-view object recognition at Chen et. al. [1]. In recent years, the main contributions to the field concern the action decision algorithm. In the early 2000', approaches (e.g. [2], [3]) focused on information theory arguments to make decisions. The next viewpoint in a task was selected in order to minimize an entropy function, i.e., to minimize the uncertainty in the state. The cost of the whole plan in terms of time and energy is neglected. In a recent work of R. Eidenberger and J. Scharinger, [4], an action control cost is added to the value of information reward. In the present work, we consider the problem solely as the minimization of time to recognition. Since time is spent in both image processing and movement actions, by minimizing time we guarantee that the viewpoints selected for image processing are the most informative.

To minimize the number of movement actions and the time spent in image processing we formalize our problem as a Partially Observable Markov Decision Process, POMDP. The partial observability arises from the incapacity of the robot to see the whole object from the same viewpoint.

The formalization of active object recognition as a POMDP is also present in some recent works. In particular we refer to [4]. However, in their work, POMDP rewards are linked to the expected minimization of information entropy from the next observation. In practice, the reward of future actions needs to be computed online, since it depends on the entropy of the current state. The dependency of rewards on the current state means that the robot has to solve a POMDP after each observation, which is a very costly process. In our approach, we assign negative rewards to all time consuming actions and all the rewards are defined a priori. This enable us to solve the POMDP problem off-line and use the policy online.

Another important feature of the current work is that we only try to recognize one object at a time. In our formulation of the POMDP problem, states are linked to object orientation with respect to the robot. By separating the object recognition problem into individual objects, we ensure that all possible states are known a priori, which is essential for solving the POMDP offline. In their work, [4], the authors aim at having the robot identifying several objects at the same time. This means that the state space is not initially known and hence the POMDP solution cannot be determined beforehand.

Contrarily to what can be found in the literature and without loss of generality, we define our state as the orientation between the robot and the object and neglect the relative distance between the two. It is assumed that the robot is able to control its distance to the object or that this distance will not pose a problem to recognition. The assumption is valid since we could use distance invariant features for object recognition, or we could just expand the number of states in our problem to accommodate relative distances between the robot and the object.

The main contribution of this work is how we represent the active object recognition as a POMDP problem. Our objective is to minimize the overall time spent by the robot in the object recognition task. As such, we want to minimize not only the time spent on movement and image processing, but also the time spent on planning. Solving a POMDP is still computationally expensive and we do not wish to solve it online. By defining all the problem offline we only need to solve the POMDP once and thus gain access to a policy which can be used online in a time efficient fashion.

This paper is organized as follows: In Section II, we present the approach overview. In Section III we formalize our problem as a POMDP, in Section IV we present our experiments and results and in Section V we draw conclusions and present future work.

II. Approach Overview

In our object model, instead of associating one object to each state and constructing a 3D model for the object, we consider all the possible orientations of the object with relation to the robot. Each orientation for each object is a state in our plan. For each state we can retrieve an observation. For example, let $\mathcal{N} = \{n_1, ..., n_N\}$ be a set of N objects, each with M possible orientations. The total number of states related to objects will be $\mathcal{S} = \{s_{1,1}, s_{1,2}, ..., s_{1,M}, ..., s_{N,1}, ..., s_{N,M}\}$. To these states we may have observations $\mathcal{O} = \{o_1, o_2, ..., o_O\}$, where $O < N \times M$. Due to ambiguity in states with relation to observations, there is not a direct relation between observations and states, i.e., observations provide us with only partial information with respect to the object and its orientation. The 3D structure of the object is coded by the state transitions when the robot moves. If the robot decides to rotate left from state $s_{n_i,m}$ it will end up in state $s_{n_i,m+1}$. To each of these states there is an associated observation $o_{n_i,m}$ and $o_{n_i,m+1}$ which may be the same. The object structure is thus represented by the fact that for the object n_i we can have access to observation $o_{n_i,m+1}$ if we rotate left after observing $o_{n_i,m}$. One example is provided in Figure 1. In this figure we have one object, a cube, and we are only considering 4 possible orientations, which give rise to 4 possible states. From each of the states we there is a single possible image which can be retrieved. However the same image can correspond to more than one orientation. The 3D shape of the object constrains the order of images that the robot can obtain when it rotates.

The correct identification of the object is then mapped to the identification of at least one of its possible orientations. The robot is able to identify the object $n_i \in \mathcal{N}$ if it is able to do one of the following: (i) identify one of the object M states $s_{n_i,m}$; (ii) have uncertainty over a set of states, all belonging to the same object. In other words, the robot will do a correct identification of the object n_i if and only if its belief distribution respects $b_{j,k} = 0, \forall j \neq n_i$.

III. POMDP Formulation

We formulate our POMDP as a tuple ($\mathcal{S}$,$\mathcal{A}$, $\mathcal{O}$, $\mathcal{T}$, Ω, $\mathcal{R}$, b_0), where:

$\mathcal{S}$ is the set of states;
$\mathcal{A}$ is the set of actions;
$\mathcal{O}$ is the set of observations;
$\mathcal{T}$ is the set of conditional transition probabilities;
Ω is the set of conditional observations probabilities;
$\mathcal{R}$ is the set of rewards;
b_0 is the initial belief.

States

States are the object orientations with respect to the robot. If we consider that those orientations are spaced with angles of $\Delta\theta$, we have $M = 2\pi/\Delta\theta$ per object. For a set of N objects, the total number of states would be $N \times M$. Furthermore, we have an extra state, the *Sink*, where the robot enters after an attempt to identify the object. In the case of our example in 1 our object is a cube and thus we need $\Delta\theta = \pi/2$ which lead

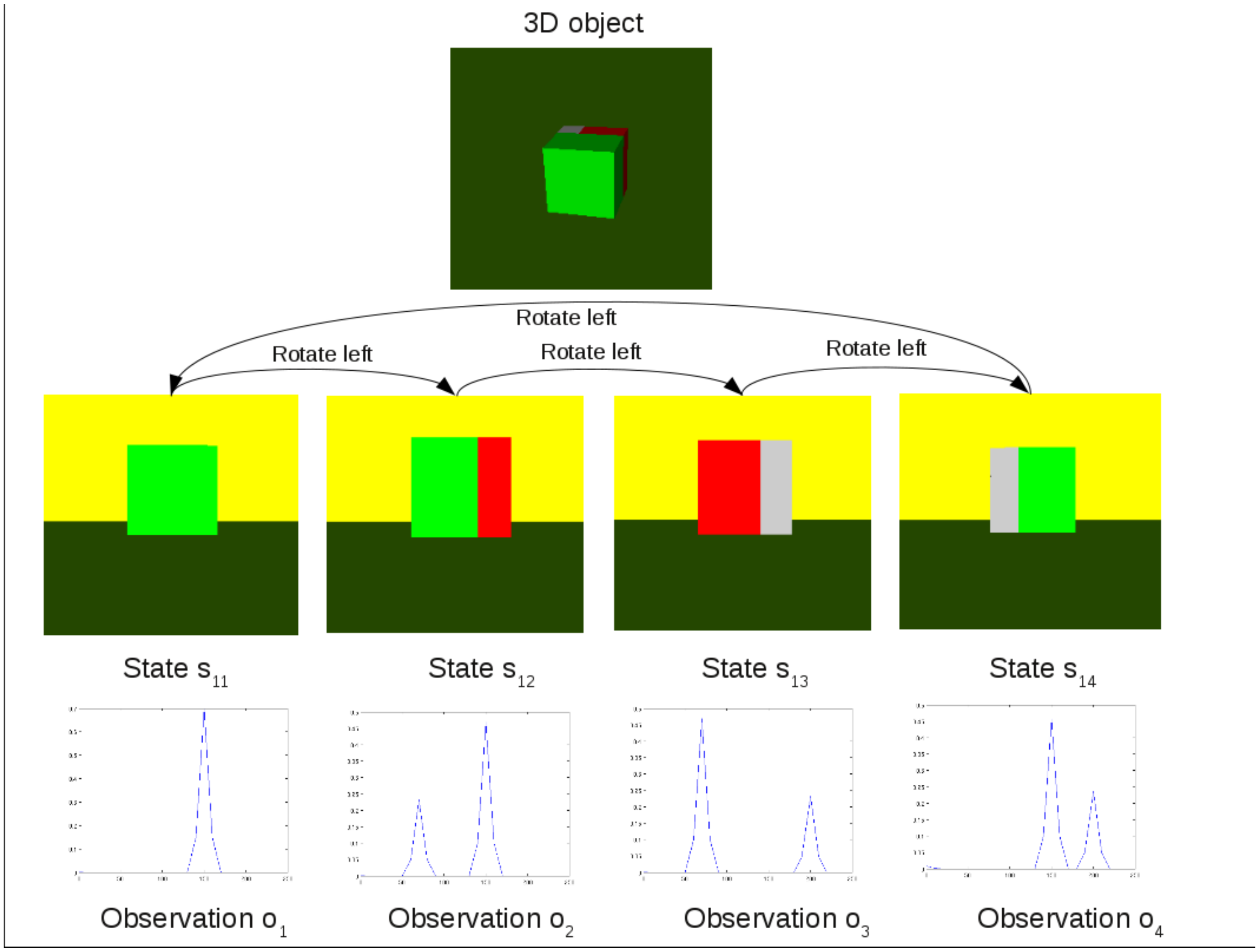


Fig. 1. Example of how one object is defined. A state corresponds to an orientation of the object with relation to the robot. We move states by applying movement actions such as rotate left. At any given state the robot may choose to do an observation. In the example, an observation corresponds to the construction of a color histogram.

to 4 states per object. For two cubes we have $2 \times 4 + 1 = 9$ states: $\mathbf{S} = \{s_{11}, s_{12}, s_{13}, s_{14}, s_{21}, s_{22}, s_{23}, s_{24}, Sink\}$ where, e.g, s_{11} corresponds to an orientation of $\theta = 0$ of the object 1 with relation to the robot and s_{24} corresponds to an orientation of $3\pi/2$ of object 2 with relation to the robot.

The advantage of this representation is that it enable us to make a direct connection between states, orientations, and movement actions.

Actions

Actions can be divided in three groups: *Movement*, *Observation* and *Identification*. We assume that the robot is moving at a constant distance from the object and the only *Movement* actions we need are rotate left or rotate right. Both actions correspond to a rotation of Δ/θ, one clockwise and the other counter clockwise. The *Observation* action involves processing a 2d image of the object taken by the robot at its current orientation. *Identification* actions correspond to the act of attempting to identify the object. The identification of an object is equivalent to the identification of one of the states corresponding to the object. In the previous example of two cubes, the correct identification of the object 1 corresponds to the identification of at least one of the states $s_{11}, s_{12}, s_{13}, s_{14}$.

The set of actions is thus defined as:

$$\begin{aligned}\mathcal{A} = \{&rotateLeft;\\ &rotateRight;\\ &observe;\\ &identify_1;\\ &identify_2;\\ &...;\\ &identify_N\},\end{aligned}$$

where N is the total number of objects being considered.

Observations

Observations are the result of processing one image from the object at the current orientation. They form a discrete set, since we are only observing the object from a finite set of orientations. We assume that the observations from each orientation are all known a priori. In this context, processing one image refers to features retrieval and image matching and the *observation* action becomes a classification process, where the features of the new image are compared with the a priori expected features for each state.

If the features chosen were good enough to completely define the object, all states would be connected to an unique observation and our problem would be reduced to a Markov Decision Process. However, this is rarely the case and commonly the features and the matching algorithm are not discriminative enough. If we cannot discriminate two or more states at all, we assign them the same observation. If we just do not trust enough in the classification, we consider the observations as different, but assign them lower probabilities in the observation table.

For the POMDP formulation, the observations are a set $\mathcal{O} = \{ o_1, o_2 , \dots , o_O\}$, where $O < N \times M$. The type of features and matching algorithms used are not relevant from the POMDP point. What matters is the classification output and all the computer vision algorithm can be treated as a black box. In Section IV-C, we will show how, for the specific example of this paper, we process the image from acquisition to an observation probability.

Transitions

Movement Actions:
The action *rotateLeft* corresponds to a rotation of $\Delta\theta$ and as such shifts between states of the cube in an ascending order: if we start with an orientation of $\theta = 0$ and rotate left we end up with an orientation of $\theta = \Delta\theta$. In terms of states for the object 1, this is equivalent to move from the state s_{11} to the state s_{12}. Formally, we can write: $T(s_{i',j'}|rotateLeft, s_{i,j}) = \delta_{i,i'}\delta_{j,(j'+1)\%M}$ for all states $s_{i,j}$ except $Sink$ and where $\delta_{i,j}$ is the Kronecker's delta, M is the number of possible orientations and $\%M$ is the operator modulo of M. There is no movement action which directs the robot to the sink state and thus: $T(Sink|rotateLeft, s_{i,j}) = 0$.

Following the previous example, we can also write: $T(s_{i',j'}|rotateRight, s_{i,j}) = \delta_{i,i}\delta_{j,(j'-1)\%M}$ and for the *Sink*: $T(Sink|rotateRight, s_{i,j}) = 0$

All movement action in the $Sink$ do not change state. Formally: $T(s|rotateLeft, Sink) = \delta_{s,Sink}$ and $T(s|rotateLeft, Sink) = \delta_{s,Sink}$.

Observe Action:
Captures and processes an image. It affects the belief, but the state remains the same. $T(s|observe, s_{i,j}) = \delta_{s,s_{i,j}}$

Identification Actions:
The identification action corresponds to an announcement of the object identity. There is an identification action per object class and all lead the robot to the $Sink$ state. $T(s'|identify_{i'}, s) = \delta_{s,Sink}$, $\forall s \in \mathcal{S}$.

Observations Probabilities

For each state there is only one possible observation, but the same observation can be retrieved from more than one state. Formally we can write: $\Omega(o_k|observation, s_{i,j}) = 1$ if the observation k corresponds to the state $s_{i,j}$ and $\Omega(o_k|observation, s_{i,j}) = 0$ if not.

Reward

The robot receives reward when it identifies an object correctly. The identification of the object corresponds to the identification of at least one of its corresponding states. This is encoded in the rewards the robot receives. In the example of two cubes with 4 orientations per object and 1 identify action per object, we have the rewards: $reward(identify_i, s_{i',j'}) = 300 \times \delta_{i,i'} - 500(1 - \delta_{i,i'})$. If the robot chooses the action $identify_i$ at any state corresponding to the object i, it will receive the reward 300. If the action is chosen in any other state, it will receive a negative reward of 500.

Furthermore, we want to minimize the number of moves and observations that the robot does, so per each of these actions we will also add a negative reward. $reward(rotateLeft) = reward(rotateRight) = -10$. The observations will be a little less expensive: $reward(observe) = -2$. The difference in reward is explained by the extra energy consumption.

Solving POMDP's

Policies were learned using Perseus algorithm [5]. This algorithm is a variation of point based methods and is freely available at the authors website.

IV. Experiments

Our experiments are performed using simulations in Matlab. In the following we describe those simulations, the type of objects and the classification performed during the *observe* action. At the end of the section we present our results and highlight the fact that algorithm always chooses the observations which enable state desambiguation.

A. Simulation

The world simulation is done using the Matlab Simulink Virtual Reality toolbox. Object orientation and image acquisition and processing are controlled by a Matlab script.

There is a perfect match between states, observations and actions between the simulator and the POMDP formulation. The same actions in the same starting states in both worlds lead to the same final states. Also, the relation between states and observations follows the same probability distribution.

B. Objects

Objects are represented by 3D cubes. With the cubes, we can represent objects variability by the colors of the faces. Different object perspectives, are represented by faces with different colors. Similar perspectives that cannot be correctly distinguished are represented by faces with the same color.

The representation of objects as cubes, albeit simple, illustrates the main characteristics of an active vision system. If we assume the robot can only move by factors of $\pi/2$, we do not need a more complicated object. All the objects when looked from these directions only present 4 different observations. The number of possible angles from which the robot can look at the object is linearly connected to the number of faces of the object model we need to consider and consequently to the number of states per object and corresponding observations. If we want to look at the object in intervals of $\Delta\theta = \pi/3$, we would need $2\pi/\Delta\theta = 6$ different observations and consequently we would need an object with hexagonal symmetry with relation to the rotation axis. However, this would increase the number of states by 1.5 times. The main consequence of the change would be the increase in the policy computation time.

The cube faces were chosen to highlight the fact that policies obtained minimize the number of observations and the number of movements. In particular, they show that the policy resulting from the POMDP forces to robot to move directly to those sides of the cube which are more informative, in the sense that observations perfomed from those sides allow to disambiguate between states. To illustrate the first situation we use cube 1 and cube 2 from Figure 2 and for the second case we use cube 1 and cube 3. The policies were constructed using just pairs of cubes.

C. Observations

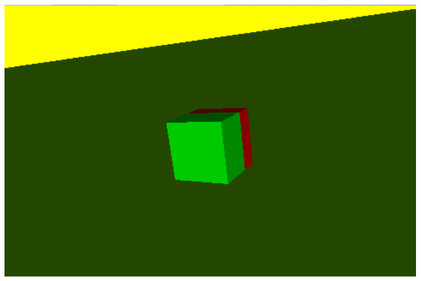
Observations correspond to the aquisition of a new image from the current orientation and its classification as one of the a priori expected observations.

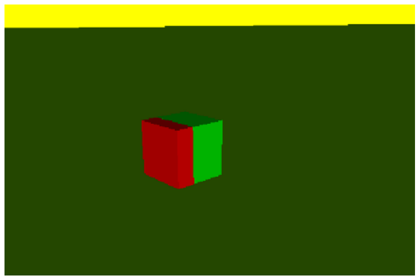
In the simulated world, we are dealing with controlled and colored images and the classification process can be greatly simplified. In these experiments, we used a nearest neighbourgh classifier based on color histograms. Examples of such histograms can be found in Figure 1. The histograms are computed in gray scale. To represent the distance between 2 histograms, we measure the cosine of the angle formed by those histograms. Two histograms from the same observation $o_k \in \mathcal{O}$ will have high cosine values (~ 1) and 2 histograms from different observations will have low cosine values.

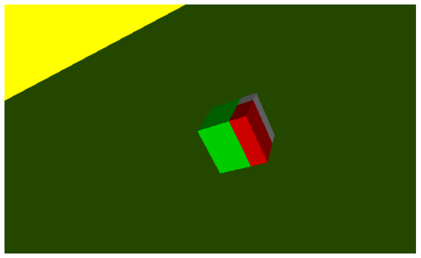
D. Results

Experiments were performed using the cubes in Figure 2. In the first experiment we used cube 1 and cube 2. The two cubes yield different observations from orientation 1 and 2, but in orientations 3 and 4 the observations are the same. The robot should be able to identify correctly the cubes after performing an *observe* action in states s_{11}, s_{12}, s_{21} and s_{22}. However, when it is facing the object from orientation 3 or 4, the robot is not able to identify directly the object. Furthermore, the robot should have different behaviors in both orientations. While at orientation 3, the shortest way to identify the objects is to go to orientation 2 where objects can be desambiguated, in orientation 4 it should chose to go to orientation 1. The actions that the robot needs to perform in order to minimize the cost of identification are different. In the first case it should chose to rotate right (moving from states $s_{*,3}$ to $s_{*,2}$) and in the second case it should chose to rotate left (moving from states $s_{*,4}$ to $s_{*,1}$).

In table I, we show the policy chosen by the robot, when facing each of the orientations of cube 2. Note that the policies described match what was just described.


(a) View 1 from object 1


(b) View 2 from object 1


(c) View 1 from object 2

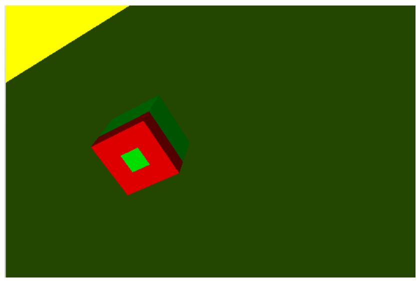

(d) View 1 from object 3

Fig. 2. Object views in our experiment. The objects have in total 6 different faces. The first object has 2 identical faces, o_1, plus 2 different ones, o_2, o_3. The second object shares two faces with object 1: o_1 and o_3 and has 2 new faces o_4 and o_5. The third one is identical to the first, but only one face changed, o_6.

In our second experiment, which is exemplified in table II, we used the first and the third cube. The two cubes differ only in one face, but have two identical faces from the point of view of the observations, i.e., the color histograms retrieved from orientations 2 and 4 are exactly the same. From the point of view of recognition, there is one single orientation, $s_{*,1}$, which allows the robot to differentiate the two objects. From all the other states, the robot will have to rotate with relation to the object in order to arive at that specific state. All the observations that it may do in any other state will not help him in the recognition task. In the example in table II this is reflected in the policies presented for the initial states $s_{*,1}$, $s_{*,2}$ and $s_{*,3}$. We also note that, due to the ambiguity in observations from state $s_{*,2}$ and $s_{*,4}$ in both cubes if the robot starts in one of these states the policy will not be optimal in the sense that it produces more movements than those stricly required to disambiguate between objects. After the first observation, the belief state is the same for both states $s_{*,2}$ and $s_{*,4}$ and thus the policy will dictate the same action in both cases. While in one of the cases, this action may lead the robot directly to the state where it desambiguate the objects, $s_{*,1}$, in the second state, the same action will take him to state $s_{*,3}$.

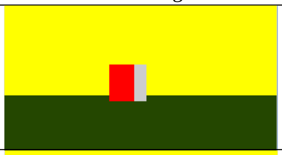
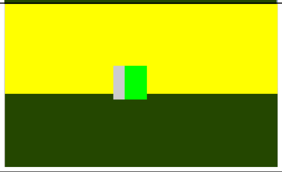
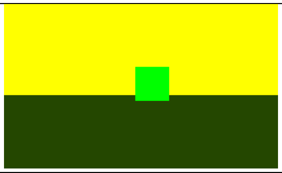
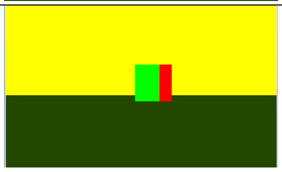
Initial State	initial Image	Policy
$s_{2,1}$		observe $identify_2$
$s_{2,2}$		observe $identify_2$
$s_{2,3}$		observe rotateRight observe $identify_2$
$s_{2,4}$		observe rotateLeft observe $identify_2$

TABLE I
POLICY FOR EXPERIMENT 1

V. Conclusions and Future Work

In this work we showed how to formalize an active object recognition task as solving an offline POMDP and provided evidence, through simulation, that the policies obtained performed observations only from the viewpoints which provided direct disambuguation between states.

The main relevance of our result is that we did not formulate the problem in terms of the commonly used information theory. Instead, we formulate the problem solely in terms of control costs. The policies, obtained by solving our POMDP problem, still ensure that the robot always chooses the observations that provide most information. The robot only performs observations which actually contribute to a decrease on the uncertainty in the current state.

By formulating the problem uniquely in terms of control costs, we can provide the robot with an a priori policy. There is no need to re-solve the POMDP during run-time and the robot does not incur in the heavy time penality caused by the extra computational effort.

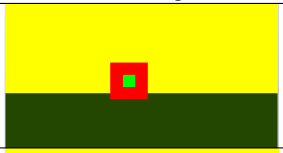
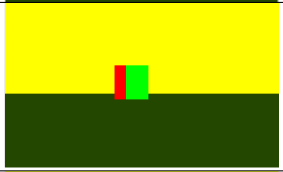
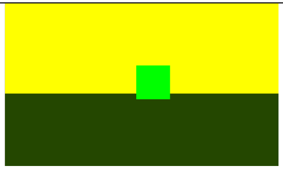
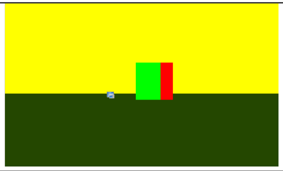
Initial State	initial Image	Policy
$s_{2,1}$		observe $identify_2$
$s_{2,2}$		observe rotateRight observe $identify_2$
$s_{2,3}$		observe rotateRight rotateRight observe $identify_2$
$s_{2,4}$		observe rotateRight observe rotateRight rotateRight $identify_2$

TABLE II
POLICY FOR EXPERIMENT 2

As future work, it is important to study the impact of adding multiple objects in the POMDP formulation. Adding more objects leads to occlusions other than self occlusion which are more difficult to model offline.

Acknowledgment

This work was partially supported by the Carnegie Mellon/Portugal Program managed by ICTI from Fundação para a Ciência e Tecnologia. Susana Brandão holds a fellowship from the Carnegie Mellon/Portugal Program, and she is with the Institute for Systems and Robotics (ISR), Instituto Superior Técnico (IST), Lisbon, Portugal, and with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA. The views and conclusions of this work are those of the authors only.

References

[1] S. Chen, Y. F. Li, J. Zhang, and W. Wang, *Active Sensor Planning for Multiview Vision Tasks*, 1st ed. Springer Publishing Company, Incorporated, 2008.
[2] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz, "Appearance-based active object recognition," *Image and Vision Computing*, vol. 18, 2000.
[3] J. Denzler and C. M. Brown, "Information theoretic sensor data selection for active object recognition and state estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 145–157, February 2002.
[4] R. Eidenberger and J. Scharinger, "Active perception and scene modeling by planning with probabilistic 6d object poses," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1036 –1043.
[5] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

Multi-Robot Coordinated Decision Making under Mixed Observability through Decentralized Data Fusion

J. Capitán, L. Merino and A. Ollero

***Abstract*— Partially Observable Markov Decision Processes (POMDPs) provide a sound mathematical framework to deal with robotic planning when tasks outcomes and perception are uncertain, but their main problem is scalability. This paper deals with planning in multi-robot teams, where this problem is even more evident. The paper presents a novel combination of two methods to cope with this complexity. *Mixed observability* can lead to simpler representations of the problem. The basic idea is to assume that some of the components in the state are fully observable, and solve the POMDP only on the partially observable part. For multi-robot teams, the complexity can be also reduced by using a *decentralized* approach in which robots have no knowledge about others' actions. In this sense, an implicit coordination will be derived from the sharing of the beliefs among the robots. The paper illustrates the advantages of the methods proposed in a multi-robot tracking application. Simulations and actual experiments are shown.**

I. Introduction

Techniques for planning under uncertainty are being applied more and more to robotics (see for example [1], [2]). In all cases, the underlying idea is that the state is only partially observable, and thus, the planning objective is to find a policy that indicates which action a robot should take given the information available (the history of actions and observations gathered by the robot so far, called the information space) in order to reach a goal. This paper is concerned with decision making in applications that consider teams of networked robots.

Considering probabilistic models and a Markovian environment, a belief state can be used to represent the information space; in this case, POMDP techniques provide an elegant way to model the interaction of a robot with its environment. Based on prior knowledge of the sensor's model and the environment dynamics, policies which indicate the robot how to act given the belief can be computed. These policies can be extracted by optimizing iteratively a certain value function over the belief space. However, the main problem of POMDPs is scalability. The optimal plan has to be searched on the belief space, which can be very large. This drawback is even more evident in multi-robot teams, as the space of actions and observations increases exponentially with the number of robots.

Approximate point-based methods to obtain POMDP policies for large state spaces have been studied ([3], [4], [5]). They restrict the optimization procedure to a bounded set of feasible sampled beliefs. Particularly, [4] proposes a point-based solver called Perseus, where no computation is needed for all the belief points at every iteration, hence improving performance. In [6], an extension of Perseus is presented. It is called *Symbolic* Perseus and uses Algebraic Decision Diagrams (ADDs) [7] in order to optimize the operations for factored POMDPs. Besides, [8] proposes SARSOP, which maintains a tree-shaped set of reachable beliefs that is expanded at every iteration using the best policy so far.

Other authors try to cope with the high dimensionality of the belief space by exploiting the idea that many robotic systems often have mixed observability, i.e. although the state is not fully observable, some components might be. Thus, [9] propose MOMDPs (*Mixed Observability Markov Decision Processes*), which separate the fully and partially observable components, leading to a lower-dimensional representation of the belief space.

Nevertheless, these techniques face ultimately an scalability problem with the number of robots in a team. A decentralized (Dec-)POMDP [10] is a suitable model for multi-agent planning considering decentralized execution. Basically, in a Dec-POMDP each agent has only access to local information, and communication is not available. Thus, the agents reason on all the potential action-observation histories of the other agents (which cannot be observed) in order to compute an optimal policy. Despite their decentralized execution, Dec-POMDP policies are computed in a centralized manner, which is a NEXP-complete problem [10].

The idea presented in this paper is that communication and data fusion between the robots can induce a common Markovian belief signal, which can be used to coordinate the robots' local plans, alleviating the mentioned complexity and scaling with the number of robots. Hence, MOMDPs are considered for local planning, whereas Decentralized data fusion (DDF) is used to induce a common belief signal among the robots and coordinate the plan execution.

The idea is similar to that proposed in [11], [12], where a distinction between cooperative and coordinated information-theoretic approaches is made, proposing the latter to control fleets of robots. In a coordinated approach, the members of the team have no knowledge about the others' models or control actions, but they exchange some information that may

This work was partially supported by the European Commission under the Cooperating Objects NETwork of Excellence CONET (contract FP7-ICT-224053) and the Spanish project ROBAIR (DPI2008-03847)

L. Merino is with Pablo de Olavide University, Carretera Utrera km1, 41013, Sevilla (Spain). lmercab@upo.es

J. Capitán and A. Ollero are with the University of Seville, Escuela Superior de Ingenieros, Camino de los Descubrimientos s/n, 41092 Sevilla (Spain). {jescap,aollero}@cartuja.us.es

A. Ollero is also with the Center for Advanced Aerospace Technologies (CATEC), Parque Tecnológico y Aeronáutico de Andalucía, C. Wilbur y Orville Wright 17-19-21, 41309, La Rinconada (Spain)

influence implicitly other members' subsequent decisions. Hence, by sharing fused perception information or the impact of others' control actions over a certain objective function, and acting locally, coordinated behavior can be obtained. However, these approaches consider mainly greedy control algorithms that try to obtain the next suitable action and do not reason on long-term goals.

The paper is structured as follows: Sections II and III give some theoretical background about POMDPs and MOMDPs; Section IV describes the communication and data fusion process to coordinate the robots; Section V details a coordinated tracking application for multiple robots to illustrate the ideas; and Sections VI and VII explain the experimental results and conclusions respectively.

II. POMDP Model

Formally, a POMDP is defined by the tuple $\langle S, A, Z, T, O, R, h, \gamma \rangle$. The *state space* is the finite set of possible states $s \in S$; the *action space*, the finite set of possible actions $a \in A$; and the *observation space* consists of the finite set of possible observations $z \in Z$. At every step, an action is taken, an observation made and a reward given. Thus, after performing an action a, the state transition is modeled by the conditional probability function $T(s', a, s) = p(s'|a, s)$, and the posterior observation by the conditional probability function $O(z, a, s') = p(z|a, s')$. The reward obtained at each step is $R(s, a)$, and the objective is to maximize the total expected reward earned during h time steps. To ensure that this sum is finite when $h \to \infty$, rewards are weighted by a discount factor $\gamma \in [0, 1)$.

Given that it is not directly observable, the actual state cannot be known by the system. Instead, a probability density function $b(s)$ over the state space is maintained. This is called the *belief state* and, due to the Markov assumption, it can be updated with a Bayesian filter for every action-observation pair:

$$b'(s') = \eta O(z, a, s') \sum_{s \in S} T(s', a, s) b(s) \tag{1}$$

where η acts as a normalizing constant such that b remains a probability distribution.

The objective of a POMDP is to find a policy that maps beliefs to actions in the form $\pi(b) \longrightarrow a$, so that the total expected reward is maximized. This expected reward gathered by following π starting from belief b is called the value function:

$$V^{\pi}(b) = E\left[\sum_{t=0}^{h} \gamma^t r(b_t, \pi(b_t)) | b_0 = b\right] \tag{2}$$

where $r(b_t, \pi(b_t)) = \sum_{s \in S} R(s, \pi(b_t)) b_t(s)$. Therefore, the optimal policy π^* is the one that maximizes that value function: $\pi^*(b) = \arg\max_{\pi} \; V^{\pi}(b)$.

III. MOMDP Model

Even for a finite set of $|S|$ states, π is defined over a $(|S| - 1)$-dimensional continuous belief space. The key in MOMDPs is to gain computational efficiency by solving a number of lower-dimensional POMDPs instead of the original one. Thus, all operations work on lower-dimensional belief spaces, which can lead to a relevant improvement in the performance.

The MOMDP [9] is represented as a factored POMDP where the state vector is composed of two different parts. Component x is the fully observable part of the original state s and y is another vector representing the partially observable part. Thus, the state is specified by $s = (x, y)$, and the state space is $S = X \times Y$, where X is the set with all possible values for x and Y all possible values for y.

Formally, the MOMDP is defined by the tuple $\langle X, Y, A, Z, T_x, T_y, O, R, h, \gamma \rangle$. The components are the same as in the POMDP case, but the transition function T is now decomposed into T_x and T_y. $T_x(x', a, x, y) = p(x'|a, x, y)$ gives the probability that fully observable state component has value x' if the robot takes action a in state (x, y). $T_y(y', x', a, x, y) = p(y'|x', a, x, y)$ gives the probability that the partially observable state component has value y' if the robot takes action a in state (x, y) and the fully observable state component has value x'.

In a MOMDP, since it can be observed, there is no need to maintain a belief over x. Therefore, it can be excluded in order to just maintain a belief $b_y(y)$, which is a probability distribution over y. Any belief $b \in B$ on the complete system state $s = (x, y)$ is then represented as (x, b_y), where $b_y \in B_y$. Furthermore, for each value x of the fully observable state component, a belief space $B_y(x) = \{(x, b_y) | b_y \in B_y\}$ is associated. Here, every $B_y(x)$ is a subspace in B, and B is the union of these subspaces $B = \bigcup_{x \in X} B_y(x)$.

Note that while B has $|X||Y|$ dimensions, each $B_y(x)$ has only $|Y|$ dimensions. Therefore, the objective of representing a high-dimensional space as a union of lower-dimensional subspaces is achieved. Moreover, when $|Y|$ is small, a remarkable computational improvement can be reached, since operations to solve the MOMDP are performed over the subspaces $B_y(x)$.

Now, since every belief is represented by (x, b_y), [9] prove that the value function can be rewritten as a piece-wise, linear and convex combination of vectors (α-vectors):

$$V(x, b_y) = \max_{\alpha \in \Gamma_y(x)} \alpha \cdot b_y \tag{3}$$

where for each x, $\Gamma_y(x)$ is a set of α-vectors defined over $B_y(x)$. Therefore, the value function is now a collection of sets of $|Y|$-dimensional vectors, and the value of the observable component x determines which $\Gamma_y(x)$ is selected. Then, the maximum over $\Gamma_y(x)$ is calculated. Now, since the vectors have $|Y|$ dimensions instead of $|X||Y|$, the execution of the policy is also faster for a MOMDP.

A. Point-Based MOMDP Solvers

Once MOMDPs have been introduced, the question is how to solve them. Fortunately, with some modifications, the same point-based algorithms for POMDPs are valid now. Since the value function consists of a set of α-vectors for every $x \in X$, the idea is to run an independent value iteration for each of them separately. In [9], for instance, the point-based solver SARSOP for POMDPs is modified in order to cope with MOMDPs. Here, Symbolic Perseus [6] has been also extended to cope with mixed observability policies. The resulting algorithm has been named MO-Symbolic Perseus and it allows a useful comparison with SARSOP for MOMDP users. Moreover, recalling that Symbolic Perseus exploits the factored representation of POMDPs and the ADD structures in order to optimize the original Perseus, MO-Symbolic Perseus can be of great interest in certain domains where the variables are not very coupled.

The two main steps to adapt point-based POMDP solvers to deal with MOMDPs are the belief update and the backup operation. In a POMDP, the belief update was determined by (1). However, since $b_y(y) = b(s)$ when $s = (x, y)$ in a MOMDP, the equation can be transformed:

$$\begin{aligned} b'_y(y') =& \eta O(z, a, x', y') \\ & \sum_{y \in Y} T_y(y', x', a, x, y) T_x(x', a, x, y) b_y(y) \end{aligned} \tag{4}$$

When the belief is updated, the values of the observable states before (x) and after (x') taking an action are known. Hence, (4) is particularized for specific values of those variables.

The other major modification in MOMDP solvers is the backup operation. Based on all the considerations made for MOMDPs, [9] show how the backup operation included in the original Perseus can be rewritten for MOMDPs:

$$\texttt{backup}(b_y) = \arg\max_{a \in A} \; b_y \cdot \alpha_a, \text{ where} \tag{5}$$

$$\begin{aligned} \alpha_a(y) =& R^x_a + \gamma \sum_{z \in Z} \sum_{x' \in X} \sum_{y' \in Y} (T_x(x', a, x, y) \\ & T_y(y', x', a, x, y) O(z, a, x', y') \alpha_{a,x',z}(y')) \end{aligned} \tag{6}$$

Finally, note that:

$$\alpha_{a,x',z}(y') = \arg\max_{\alpha \in \Gamma_{y,n-1}(x')} \; \alpha(y') \cdot b^z_{a,y}(y') \tag{7}$$

In this case, unlike (4), all the possible x' must be taken into account. Even though x' and z are concrete values, they cannot be known beforehand like x, so all their values must be considered and weighted by their probabilities.

Apart from the steps mentioned above, MO-Symbolic Perseus also needs some modifications when initializing. First, a different set of beliefs $B_y(x)$ over the component y must be sampled for every value of x. In case y is probabilistically independent of x, the same set of beliefs over y could be used for every x. Moreover, all the value functions $\Gamma_y(x)$ are initialized separately with the same values as in the original Symbolic Perseus.

IV. Coordination through Decentralized Data Fusion

Given a multi-robot planning problem, the first option is to solve the above MOMDP for the whole team, considering all the potential joint actions and observations, but this approach ultimately does not scale with the number of robots. Hence, for a team of N robots, a decentralized scheme is proposed, where each robot i solves its own MOMDP without considering the other robots actions. However, if the robots solve independent MOMDPs, and use only their local information (action and observations), no coordination or cooperation is achieved. On the other hand, if communication is allowed among the robots and the same belief state can be recovered locally, this would represent a coordination signal that summarizes all the information gathered by the fleet. This way, the execution of the policies of the different robots will be coordinated. The solution should be of lower quality than a fully cooperative centralized solution, but it can represent a tradeoff between quality and complexity.

Then, once the policies have been calculated, the coordination is achieved during the execution phase by sharing a common belief state $b_{cen}(y)$ that considers information from all the robots. If $a^J = \langle a^1, \cdots, a^N \rangle$ is the joint action and $z^J = \langle z^1, \cdots, z^N \rangle$ the joint measurement, a centralized node with access to all the information would update the belief according to (4):

$$b'_{cen}(y') = \eta p(z^J | a^J, x', y') \sum_{y \in Y} p(x', y' | a^J, x, y) b_{cen}(y) \tag{8}$$

The question is how to recover this centralized belief in a decentralized manner. Assuming that the data gathered by the different robots at any time instant are *conditionally independent* given the state at that instant s', i.e. $p(z^J|a^J, x', y') = \prod_i p(z^i|a^i, x', y')$, and the prediction does not depend on the robot actions (or the robot actions are known when predicting), it is possible to combine locally the received belief from other robots with the local one of robot i, $b'_i(y')$, to recover the centralized belief [13], [14]:

$$b'_{cen}(y') \propto b'_i(y') \prod_{j \neq i} \frac{b'_j(y')}{b'_{ij}(y')} \tag{9}$$

where $b'_{ij}(y')$ represents the common information between the robots i and j (i.e. information previously exchanged between the robots). This common information can be maintained by a separate filter called channel filter [15]. If there are loops in the information channels, the problem of double-counting should be taken into account as well. The authors have shown previously [14] that it is possible, by including delayed states in the belief, to obtain locally the same belief as in a centralized node with access to all the information available.

V. MOMDP for Target Tracking

In order to illustrate the proposed approach, an application for tracking a target by means of multiple robots is considered here. In this problem there is a moving target and a team of N robots which are the pursuers. Each of these robots carries a sensor which determines whether the target is visible or not within its field of view (FOV). Then, the objective is to find the target in the environment and localize it as well as possible.

The state is composed of the position of the target and the position and heading (*north*, *west*, *south* or *east*) of the pursuer robots. The state space is discretized into a cell grid, and a map of the scenario is assumed to be known. At each time step, each robot can choose among four possible actions: *stay*, *turn right*, *turn left* or *go forward*. *stay* means doing nothing; when *turning*, the robot changes its heading 90^o degrees; and when *going forward*, it moves to the cell ahead. Nonetheless, noisy transition functions for the states of the robots are considered. Besides, the target is assumed to move randomly. Therefore, the transition function for its position indicates that, from one time step to the next, the target can move to any of its 8-connected cells with the same probability (only non-obstacle cells are considered in order to calculate that probability).

In addition, every sensor provides a boolean measurement: *detected* or *non-detected*. These sensors proceed as it follows, if the target is out of its FOV, the sensor produces a *non-detected* measurement. However, when the target is within its FOV, it can be *detected* with a probability p_D. The robots are heterogeneous in the sense that the sensor's FOV and p_D could vary from one robot to another.

Finally, the design of the reward function so that the target is tracked by the team of robots is crucial. Since some cooperation is desirable within the heterogeneous team, a different behavior is assigned to each robot. Thus, the reward function also depends on the specific robot: $\{R^1(x,y,a), R^2(x,y,a), \cdots, R^N(x,y,a)\}$. For all the members of the team, no cost is assigned to the action *stay*, whereas a cost of 1 is associated to the other actions.

This application is a fair example of how MOMDPs can help to reduce the belief space dimensionality. Here, even though robots and target locations are considered within the state, the one involving a greater uncertainty is the latter. Here, the locations of the robots are assumed to be observable (they could be obtained accurately enough by means of the on-board sensors and the available map, at least for the cell resolution), remaining as non-observable the target's position. Thus, the non-observable part of the state consists of the target location, $y = t_l$, whereas the fully observable part consists of all the robots locations and headings, $x = (r_l^1, r_h^1, \cdots, r_l^N, r_h^N)$. In this case, for a 10x10-cell grid, there would be 400 possible states for each pursuer and 100 for the target, which means, for a single robot, to reduce a POMDP with a 40,000-dimensional belief space to a union of 400 disjoint 100-dimensional subspaces.

Then, the proposed coordinated approach will be applied. That means that each robot i solves its own MOMDP without considering the other robots. That MOMDP has states $x^i = (r_l^i, r_h^i)$ and $y^i = t_l$, and reward $R^i(x,y,a)$, being possible to specify a different strategy for each robot.

With this decentralized approach based on MOMDPs, coordination arises implicitly due to the fused belief, and there is no need to solve a MOMDP for the whole team nor a POMDP, whose complexities grow exponentially with N. Of course, the policy is not optimal, as the robots do not reason about the other robot actions, but, as it will be seen, through proper design of the rewards and the communication between robots, it can be obtained a helpful coordinated behavior for many applications.

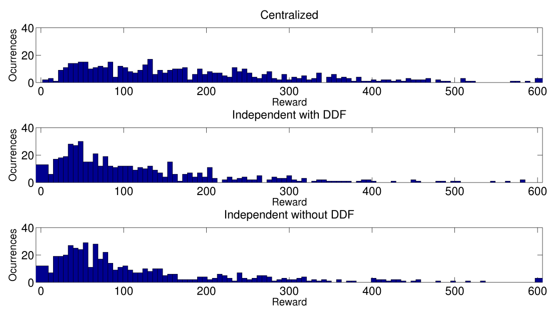


Fig. 1: Histograms of the average rewards obtained during 500 simulations for the different approaches. At each simulation the expected discounted reward for the whole team is considered.

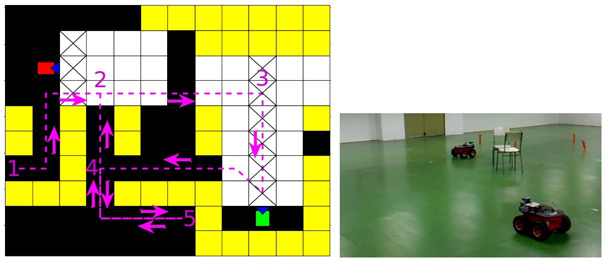


Fig. 2: a) Simulated environment and field of view (white cells) for each robot. Cells representing obstacles are in yellow. If the target is in one of the cells with crosses, a high reward is obtained. The path of the target is also shown. b) Image of the experiments in the real CONET testbed.

VI. Experiments

A. Simple scenario

A simple target tracking scenario was simulated in order to highlight the differences between the proposed approach and a fully cooperative centralized solution. The model is the one explained above particularized for a map with 11 available cells (a 3x4 grid with an obstacle in the middle). Then, two pursuer robots are considered with equal FOVs, composed by a single cell in front of them and $p_D = 0.9$. Each robot gets a high reward ($+100$) when the target is in the cell of its FOV, otherwise the reward is zero. The discount factor is $\gamma = 0.9$.

A MOMDP for a single robot without considering the other and a fully cooperative MOMDP considering both robots were solved. The reward for the latter model was simply the sum of the independent rewards for each robot. Moreover, both policies were calculated with SARSOP in a computer with an Intel Core 2 Duo processor @2.47GHz and 2.9GB, being the former computed for 0.2 seconds and the latter for 4677. Then, three different approaches were tested: (i) the fully cooperative policy; (ii) independent policies for each robot but using DDF; (iii) independent policies for

	Coordinated	Not coordinated
MO-Symbolic Perseus	246.13 ± 3.33	209.38 ± 3.17
SARSOP	279.64 ± 4.67	197.76 ± 3.10

TABLE I: Rewards (with 95% confidence interval) obtained with and without coordination (DDF) during the simulations. The value at each simulation is obtained by computing the expected discounted reward for the whole team ($\gamma = 0.9$).

each robot without sharing any information. Fig. 1 depicts histograms for the average rewards obtained during 500 simulations of 150 steps each one.

In this case, with much less computation time, the quality of the proposed coordinated approach was quite close to the fully cooperative. Actually, computing the single-robot policy for 20 seconds the proposed approach outperformed the centralized. In general, the reward is usually higher when sharing information compared to no data fusion. Nonetheless, due to the small size of the map, differences were not very substantial in this example.

B. Simulations

Additional simulations for more complex environments were conducted in order to show the advantages of introducing *mixed observability* and coordination through DDF. A simulation environment was created from the original testbed of the Cooperating Objects Network of Excellence (CONET)[1]. The map of the real testbed was discretized into 2x2-meter cells and resulted in the occupancy grid of 12x10 dimensions shown in Fig. 2, where cells representing obstacles are in yellow.

Again, a team with two robots is considered. Nevertheless, their perception capabilities are different. The first robot carries a more accurate sensor ($p_D = 0.9$) but its FOV is smaller, whereas the second has a wider FOV but is less accurate ($p_D = 0.8$) (see Fig. 2). That is reasonable, because many vision-based detectors work more accurately when the FOV of the scene is more restricted. Therefore, the role of the robot with the wider FOV would be to survey a big area from a distant position whereas the robot with the more accurate sensor would try to get closer to confirm the target detection. Hence, the first robot gets a high reward (+100) when the target is in one of the closest cells of its FOV, but the second robot gets a high reward (+100) when the target is in one of the central cells of its FOV (see Fig. 2); otherwise the reward is zero.

First, the decentralized approach with independent MOMDPs proposed in Section V was tested for this scenario. Independent policies were calculated for each robot with SARSOP and MO-Symbolic Perseus. Then, 500 simulations of 150 steps each one were performed with the robots starting at random positions and the target following the fixed path depicted in Fig. 2. In order to include some uncertainty in its behavior, at every time step, the target could (with equal probability) either stay in the same cell or follow the path.

[1]http://www.cooperating-objects.org

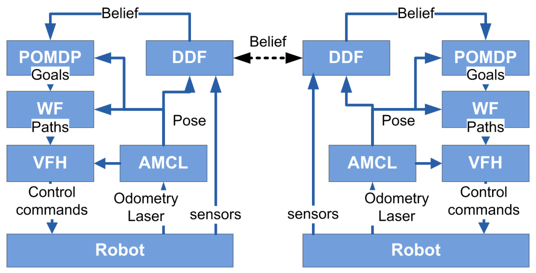


Fig. 3: Software architecture of the real robots.

Map size	Model	Time (s)	Precision	Reward
12 cells	MOMDP	0.18	21.97	125.87 ± 6.19
	POMDP	1517	21.99	88.74 ± 8.90
80 cells	MOMDP	1054	54.89	64.12 ± 8.61
	POMDP	3169	54.80	44.49 ± 8.70
120 cells	MOMDP	3071	57.42	42.58 ± 5.84
	POMDP	3081	47.30	40.24 ± 7.18

TABLE II: Evaluation of POMDP and MOMDP policies for a single robot (rewards with their 95% confidence interval) as the state space is increased. Expected discounted rewards ($\gamma = 0.9$) are considered to compute the average.

Table I shows the average rewards obtained with and without coordination (DDF). It can be seen how, for both solvers, the use of DDF produces an increase of the average reward. It is also crucial to remark that a comparison with the fully coordinated MOMDP is not included here due to the complexity of the domain. Both solvers run out of memory (with the same computer mentioned in the previous section) trying to compute a fully coordinated policy.

Finally, to show the advantage of introducing the *mixed observability* in the model, some simulations (100 runs of 150 steps) with a single pursuer robot were carried out varying the size of the scenario and comparing with a standard POMDP solution. This robot has the same sensor configuration as the red robot in Fig. 2. Three different scenarios are considered: a 3x4 grid without occupied cells; the scenario in Fig. 2 (80 cells); and a 12x10 grid without occupied cells. The results of the average rewards for both, POMDP and MOMDP policies, are summarized in Table II. All the policies were calculated with the same solver (SARSOP) and the same computer (the one mentioned above), and their precision measured, this meaning the distance between the upper and lower bounds of the value function [8]. For the smallest and medium scenarios, it can be seen that the MOMDP policy can achieve better average rewards with much less computation time. The biggest scenario tries to show how the results are better for the MOMDP after similar computation time.

C. Real experiments

In order to show the applicability of the approach, some real experiments were conducted. In these experiments, two robots were used to follow a target represented by a third robot (see Fig. 2). The models and scenario considered were the same as in the simulation described above.

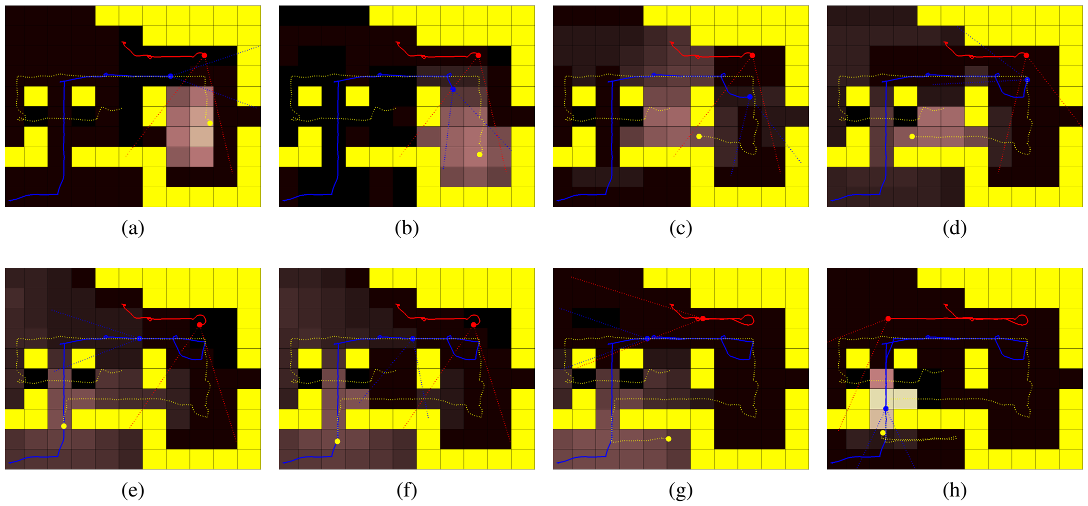


Fig. 4: Screenshots of a real experiment using coordinated robots. The color scale of the cells represents the (fused) belief from the blue robot. In yellow the target. The FOVs of the robots are also represented. Yellow cells cannot be reached.

Figure 3 shows the software modules used by the robots. Player [16] was used to control the robots, which were able to localize themselves within the map. Besides, a path planning algorithm was used to obtain the path to the high level goals provided by the MOMDP controller (next cell to move), whereas a local navigation algorithm was used to safely navigate the given path. Each robot had an estimation filter implementing the DDF scheme of Section IV and a MOMDP controller that executed the obtained policies.

Results of the experiment with coordinated execution[2] are summarized in Fig. 4. The localization of the target is improved, as it can be seen in screenshot 4a, where the belief state of the blue robot is narrow even though the target is not in its FOV. Moreover, a certain coordination arises between the robots. For instance, in screenshots 4c, 4d, 4e, the target escapes and the blue robot goes through its lowest cost path while the red robot keeps observing the place, as there is a certain probability of the target coming back and it has a larger FOV. When most of the probability mass is in the lower left room (4f), both robots eventually move to that direction (4g), and the blue robot moves inside the room while the red awaits observing from afar (4h). Fig. 5 compares the trajectories of the robots and target in two different experiments, with and without coordination.

VII. Conclusions

The scalability of POMDP models is a concern for their application to multi-robot planning. This paper addresses decentralized data fusion as a manner to coordinate independent plans computed by local MOMDPs. Since each robot does not reason about the others' actions, this approach is scalable. The main drawback is that sub-optimal policies are obtained, and no intentional cooperation is considered. However, a proper design of the local rewards allow to cope with different applications with a coordinated team. Thus, the paper mainly contributes showing through simulations the advantages of introducing MOMDPs and a coordinated decentralized solution versus a fully cooperative solution. Real experiments are used to demonstrate the applicability of the approach. Moreover, a modification of the algorithm Symbolic Perseus to deal with mixed observability is proposed and compared.

[2]See http://grvc.us.es/staff/jescap/expCONET3.mp4

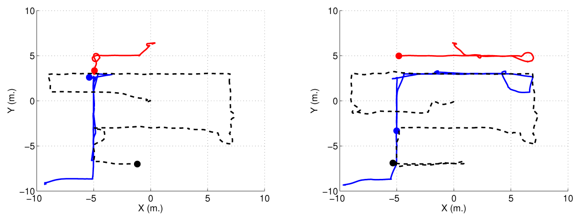


Fig. 5: Paths of the target (black) and robots. Left: no coordination. Right: coordination. In the first case the robots lost the target in part of the scenario.

Future work includes demonstration with fleets of more robots. A guideline to design the reward functions to achieve a desired coordination would be of interest as well. Certainly, better plans could be obtained if the actions of several robots were considered during the planning phase. Thus, another direction is to include other robots' actions in the planning phase, but trying to maintain locality (that is, to consider only the actions of potential neighbor robots, no the whole fleet). This should lead to better policies, but maintaining the scalability of the approach.

References

[1] N. Vlassis, G. Gordon, and J. Pineau, "Planning under uncertainty in robotics," *Robotics and Autonomous Systems*, vol. 54, no. 11, 2006, special issue.

[2] S. Prentice and N. Roy, "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1448–1465, November/December 2009.

[3] J. Pineau, G. Gordon, and S. Thrun, “Point-based value iteration: An anytime algorithm for POMDPs,” in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003, pp. 1025–1032.

[4] M. T. J. Spaan and N. Vlassis, “Perseus: Randomized point-based value iteration for POMDPs,” *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[5] T. Smith and R. Simmons, “Point-based POMDP algorithms: improved analysis and implementation,” in *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence*, 2005, pp. 542–547.

[6] P. Poupart, “Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes,” Ph.D. dissertation, University of Toronto, 2005.

[7] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier, “SPUDD: Stochastic planning using decision diagrams,” in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[8] H. Kurniawati, D. Hsu, and W. Lee, “SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces,” in *Proc. Robotics: Science and Systems*, 2008.

[9] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee, “POMDPs for Robotic Tasks with Mixed Observability,” in *Robotics: Science and Systems Conference*, 2009.

[10] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, “Optimal and approximate q-value functions for decentralized POMDPs,” *Journal of Artificial Intelligence Research*, 2008.

[11] B. Grocholsky, A. Makarenko, T. Kaupp, and H. F. Durrant-Whyte, *Lecture notes in Computer Science*. Springer, 2003, vol. 2634, ch. Scalable Control of Decentralised Sensor Platforms.

[12] G. Mathews, H. Durrant-Whyte, and M. Prokopenko, *Advances in Applied Self-Organizing Systems*. Springer-Verlag, 2008, ch. Decentralized Decision Making for Multiagent Systems.

[13] E. Nettleton, H. Durrant-Whyte, and S. Sukkarieh, “A robust architecture for decentralised data fusion,” in *Proc. of the International Conference on Advanced Robotics (ICAR)*, 2003.

[14] J. Capitan, L. Merino, F. Caballero, and A. Ollero, “Delayed-State Information Filter for Cooperative Decentralized Tracking,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[15] F. Bourgault and H. Durrant-Whyte, “Communication in general decentralized filters and the coordinated search strategy,” in *Proc. of The 7th Int. Conf. on Information Fusion*, 2004, pp. 723–730.

[16] B. Gerkey, R. Vaughan, and A. Howard, “The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems,” in *Proceedings of the 11th International Conference on Advanced Robotics, ICAR*, 2003, pp. 317–323.

Petri Net Based Supervisory Control of a Social Robot with LTL Specifications

Bruno Lacerda, Pedro U. Lima, Javi Gorostiza and Miguel A. Salichs

Abstract— **We describe the implementation of a method to control a social robot based on discrete event system supervisory control theory. The sensors and actuators of the robot are modelled as Petri nets, and the target behaviour is given as a set of rules written as linear temporal logic (LTL) formulas. The Petri net models and LTL rules are then used to build a Petri net realization of a supervisor that is guaranteed by construction to restrict the robot's behaviour such that the rules are fulfilled. This approach provides a close-to-natural-language description of the target behaviour, and can be the basis for a human-robot speech interaction device, where the rules for the robot to fulfil are described by the human in natural language.**

I. INTRODUCTION

Social robotics is an emerging field where a special attention is paid to the interaction between the robot and the human user(s). To be appropriate for this kind of interaction, a robot must fulfil several requirements, such as providing different forms of interacting (e.g., speech, touch), possessing a high level of autonomy, and being able to learn and to adapt to different situations.

In this work, we present the application of a method that implements linear temporal logic (LTL) [7] specifications in Maggie [14], a social robot developed at the RoboticsLab of Universidad Carlos III de Madrid with the main purpose of interacting with children. We will model a subset of Maggie's sensors and actuators as Petri nets (PN) [13], and define rules that restrict Maggie's behaviour in LTL. The PN and the LTL formulas are then used to build a PN realization of a supervisor [4] that forces the system to fulfil the formulas. The use of LTL provides a close-to-natural-language framework to define the rules for the robot, which can allow the development of a speech interaction module, where the user states the rules using a pre-defined subset of natural language, and the robot then shows the behaviour originated by those rules. This kind of interaction can be used as a game for children, where they can see the impact of the rules they state in the behaviour of the robot.

The work presented here can be seen as a first step to develop an alternative approach to the work presented in [9], where a method to teach action sequences by means of speech interaction is defined. This method relies on speech interaction with the user, who utters orders in a pre-defined subset of natural language. The orders are translated to a sequential function chart (SFC) [5], a graphical programming language based on binary PNs, used for programmable logic controllers. SFCs feature elements such as conditional choices, parallel execution of sequences and loops. In this approach, the user starts from an "empty plan", and builds a behaviour for the robot by fully stating all the actions, and in which order and in which conditions they are to be performed. Conversely, in our work, the user starts with all the possible actions for the robot being able to be executed randomly and restricts them to the desired behaviour by stating rules that the robot must fulfil. Hence, [9] can be seen as a direct approach to building sequences for the robot to execute where the whole behaviour must be specified step-by-step, while we propose a more abstract approach, where the desired behaviour is obtained from a set of general rules. This might lead to the appearance of "surprises" in the behaviour of the robot, in the sense that it may exhibit properties that were not thought of by the user when stating the rules.

Another technique used for humans teaching behaviours to robot systems, which has presented good results, is programming by demonstration, where the user shows the robot how to perform a given task, being imitated by the robot afterwards. Two examples of this approach are [1] and [3]. Due to its suitability to model concurrent systems and the wide range on analysis methods available, PNs are a widely used tool for the modelling of robot systems [15], [6]. Also, LTL has been successfully applied as a language to specify and synthesize correct by design admissible behaviours [12], [10]. Furthermore, a translation between a restricted subset of English and LTL to deal with motion planning problems for mobile robots is defined in [11].

The paper is outlined as follows: in Section II we provide a brief description of Maggie, the platform where we implemented the method. In Section III we show how to model Maggie's sensors and actuators as Petri nets where some places correspond to the truth value of binary variables used to describe the state of the system, followed by an explanation on how to write the LTL specifications, in Section IV. Section V describes the method for the construction of the supervisor and, in Section VI, a discussion about the approach and future developments is provided.

B. Lacerda and P. Lima are with the Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, Portugal {blacerda, pal}@isr.ist.utl.pt. The work of B. Lacerda was supported by the portuguese Fundação para a Ciência e Tecnologia through grant SFRH/BD/45046/2008. Most of this work was accomplished while they were visiting Universidad Carlos III de Madrid, in P. Lima's case under a Cátedra de Excelencia awarded by Banco Santader.

J. Gorostiza and M. Salichs are with the Robotics Lab, Universidad Carlos III de Madrid, Madrid, Spain {jgorosti, salichs}@ing.uc3m.es

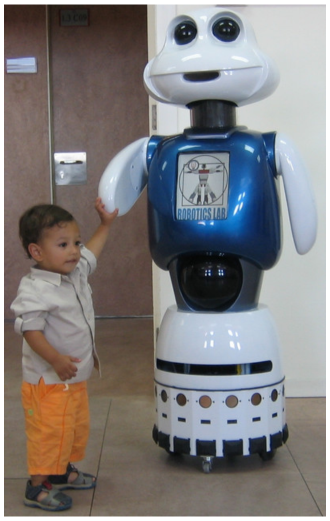


Fig. 1. Maggie, the social robot, interacting with a child.

II. Maggie, The Social Robot

Maggie, depicted in Figure 1, is a social robot developed at the RoboticsLab of Universidad Carlos III de Madrid to interact mainly with children. In this Section, we provide a brief description of its capabilities, further details can be found in [14].

Maggie is a 1.35m tall girl-like doll. Her base is equipped with two differentially driven wheels and a caster wheel on each side. The arms and the eyelids have 1 DOF: up/down, while the neck has 2 DOF: up/down and left/right. She also possesses tactile sensors, including on the shoulders and on top of her head. These are the sensors and actuators that we took into account in our implementation. Maggie possesses many other capabilities, such as infrared and ultrasound sensors used for navigation, a color camera for people tracking and a mouth shape with invisible web-cam and coloured lights synchronized with the speech. These capabilities result in a platform well suited to study human-robot interaction and robot learning by training and teaching.

The control architecture is based on the automatic-deliberative (AD) control architecture proposed in [2]. In this architecture, the robot skills are divided in two levels. In the automatic level, we find the skills related with robot sensors and actuators. In the deliberative level, we find higher-level skills, such as a planner or our implementation of the feedback-loop of supervisory control. The skill we developed subscribes to events representing changes in sensor readings, for which an event handler is implemented. Also, it can order the execution of skills related to performing simple actions (e.g., raising an arm or start spinning).

III. Petri Net Modelling and Execution

We will use PNs to model Maggie behaving freely in the environment. This model is a building block of the supervisor, being used in conjunction with the LTL formula specifying the target behaviour to build it. In the PN models used here both places and transitions have a specific interpretation:

- *Places* represent the value of binary variables that are used to define the state of the system. For each variable, there is one place representing that is is *true* and one place representing that is is *false*. For all reachable markings of the Petri net, there is one token in one of these places and zero in the other. This means that, for a given marking, one can unambiguously extract the corresponding state. A state is the set of variables for which the places meaning that they are true have one token;
- *Transitions* represent orders to execute actions or changes in sensor readings, i.e., the firing of a transition represents a communication between our implementation and the automatic level of the architecture.

In Figure 2, the PN model for Maggie is depicted. The model is composed of several modules, one for each actuator and one of each sensor. We represent the interpretation of each place and transition as $\langle . \rangle$. For example, a token in place p_1 means that $moving\_forward$ is true and a token in place p_2 means that $moving\_forward$ is false. Furthermore, the firing of transition t_1 represents executing the $move\_forward$ action and the firing of transition t_{19} means that someone started touching the tactile sensor on the left shoulder.

To illustrate, the initial state for Maggie, given by the depicted marking, is the set:

$$\{base\_idle,\ head\_center,\ head\_down,\ left\_arm\_down,\ right\_arm\_down,\ left\_eyelid\_down,\ right\_eyelid\_down\}$$

We note that, for the base, when a *start* action is issued, the robot starts performing that action until a *stop* action is issued, that is, the action continues until it is explicitly stopped. For the other models, we assume that the actuator moves for a fixed amount and then stops, hence no *stop* action is required.

We implemented a *Petri net executor* in C++, so that the robot is able to run its system Petri net. We start by dividing the transitions into transitions corresponding to actions - t_1 to t_{18} - and transitions corresponding to changes in sensor readings - t_{19} to t_{24}. The implementation is a loop that, in each step, randomly selects one of the active transitions corresponding to actions and fires it, ordering the execution of the corresponding action and updating the marking. Also, when one of the sensors changes state, the handler interrupts the loop, and the transition corresponding to that sensor change is immediately fired and the marking is updated. By running this PN without supervising it, Maggie simply executes random actions, displaying an unrestricted behaviour.

IV. Writing the LTL Specification

We will write LTL formulas that restrict Maggie's behaviour. These formulas are written over the set Π, defined by the union of the set E of events (actions plus sensor readings) and the set D of variables that describe the states. Formulas

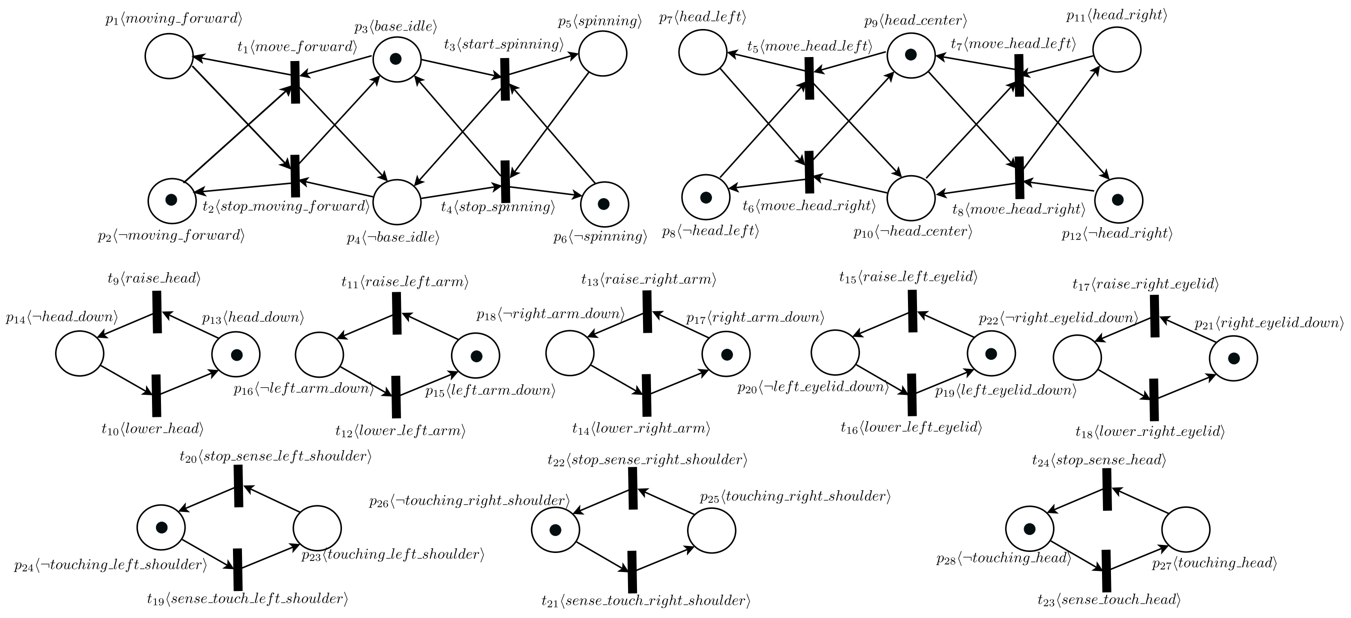

$p_1\langle moving\_forward\rangle$
$t_1\langle move\_forward\rangle$
$p_3\langle base\_idle\rangle$
$t_3\langle start\_spinning\rangle$
$p_5\langle spinning\rangle$
$p_7\langle head\_left\rangle$
$t_5\langle move\_head\_left\rangle$
$p_9\langle head\_center\rangle$
$t_7\langle move\_head\_left\rangle$
$p_{11}\langle head\_right\rangle$
$t_2\langle stop\_moving\_forward\rangle$
$t_4\langle stop\_spinning\rangle$
$p_2\langle\neg moving\_forward\rangle$
$p_4\langle\neg base\_idle\rangle$
$p_6\langle\neg spinning\rangle$
$t_6\langle move\_head\_right\rangle$
$t_8\langle move\_head\_right\rangle$
$p_8\langle\neg head\_left\rangle$
$p_{10}\langle\neg head\_center\rangle$
$p_{12}\langle\neg head\_right\rangle$
$t_9\langle raise\_head\rangle$
$t_{11}\langle raise\_left\_arm\rangle$
$t_{13}\langle raise\_right\_arm\rangle$
$t_{15}\langle raise\_left\_eyelid\rangle$
$t_{17}\langle raise\_right\_eyelid\rangle$
$p_{14}\langle\neg head\_down\rangle$
$p_{13}\langle head\_down\rangle$
$p_{18}\langle\neg right\_arm\_down\rangle$
$p_{17}\langle right\_arm\_down\rangle$
$p_{22}\langle\neg right\_eyelid\_down\rangle$
$p_{21}\langle right\_eyelid\_down\rangle$
$p_{16}\langle\neg left\_arm\_down\rangle$
$p_{15}\langle left\_arm\_down\rangle$
$p_{20}\langle\neg left\_eyelid\_down\rangle$
$p_{19}\langle left\_eyelid\_down\rangle$
$t_{10}\langle lower\_head\rangle$
$t_{12}\langle lower\_left\_arm\rangle$
$t_{14}\langle lower\_right\_arm\rangle$
$t_{16}\langle lower\_left\_eyelid\rangle$
$t_{18}\langle lower\_right\_eyelid\rangle$
$t_{20}\langle stop\_sense\_left\_shoulder\rangle$
$t_{22}\langle stop\_sense\_right\_shoulder\rangle$
$t_{24}\langle stop\_sense\_head\rangle$
$p_{26}\langle\neg touching\_right\_shoulder\rangle$
$p_{25}\langle touching\_right\_shoulder\rangle$
$p_{24}\langle\neg touching\_left\_shoulder\rangle$
$p_{23}\langle touching\_left\_shoulder\rangle$
$p_{28}\langle\neg touching\_head\rangle$
$p_{27}\langle touching\_head\rangle$
$t_{19}\langle sense\_touch\_left\_shoulder\rangle$
$t_{21}\langle sense\_touch\_right\_shoulder\rangle$
$t_{23}\langle sense\_touch\_head\rangle$

Fig. 2. The Petri net model for Maggie's different actuators and sensors.

LTL are written using the usual propositional connectives plus a set of temporal connectives, including the next (X), always (G) and until (U) connectives. LTL formulas are evaluated over infinite sequences of sets of propositional symbols $\sigma : \mathbb{N} \to 2^{\Pi}$. Intuitively, for $i \in \mathbb{N}$:

- A state $\sigma(i)$ satisfies $X\varphi$ when state $\sigma(i+1)$ satisfies φ;
- A state $\sigma(i)$ satisfies $G\varphi$ if all states $\sigma(j)$ with $j \geq i$ satisfy φ;
- A state $\sigma(i)$ satisfies $\varphi U \psi$ if φ is satisfied in all states, until a state appears where ψ is satisfied.

With these operators, we can specify a wide array of different rules for Maggie to fulfil. We will illustrate this by providing the LTL specifications for three different target behaviours. The first behaviour is a simple sequence triggered by touching the head of the robot. The sequence can be stated in natural language as "*When the head is touched raise the left arm, then raise and lower the right arm. When the head stops being touched, lower the left arm*". This sequence can be translated into the following simple rules:

- The left arm must only be raised when the head is being touched:

$$G(touching\_head \Leftrightarrow (X(\neg left\_arm\_down))) \tag{1}$$

- The right arm must only be raised after the left arm is raised:

$$G(raise\_left\_arm \Leftrightarrow (X raise\_right\_arm)) \tag{2}$$

- After raising the right arm, it must be immediately lowered again:

$$G(raise\_right\_arm \Rightarrow (X lower\_right\_arm)) \tag{3}$$

The second behaviour shows how to specify different reactions to different sensor readings. We can state it has "*If the left shoulder is touched, wait until the right shoulder is touched and then raise your right arm. If the right shoulder is touched, wait until the left shoulder is touched and raise your left arm. After raising an arm, lower it again and return to the initial state*". To keep track of which shoulder is touched, we use the state of the eyelids. If both eyelids are down, then no shoulder has been touched yet. If one of the eyelids is up, then the corresponding shoulder was touched and Maggie is waiting for the other shoulder to be touched. Hence, the behaviour can be implemented by the following rules:

- If the left shoulder is touched and both eyelids are down (i.e., no shoulder was touched yet), raise the left eyelid:

$$G((sense\_left\_shoulder \wedge right\_eyelid\_down \wedge left\_eyelid\_down) \Leftrightarrow (X raise\_left\_eyelid)) \tag{4}$$

- If the right shoulder is touched and the left eyelid is up (i.e., the left shoulder was previously touched), raise the right arm:

$$G((sense\_right\_shoulder \wedge \neg left\_eyelid\_down) \Leftrightarrow (X raise\_right\_arm)) \tag{5}$$

- After raising the right arm, return to the initial state by lowering the left eyelid and the right arm:

$$G(raise\_right\_arm \Leftrightarrow (X lower\_left\_eyelid)) \tag{6}$$

$$G(lower\_left\_eyelid \Leftrightarrow (X lower\_right\_arm)) \tag{7}$$

- If the right shoulder is touched and both eyelids are down (i.e., no shoulder was touched yet), raise the right eyelid:

$$G((sense\_right\_shoulder \wedge right\_eyelid\_down \wedge \\ left\_eyelid\_down) \Leftrightarrow (Xraise\_right\_eyelid)) \quad (8)$$

- If the left shoulder is touched and the right eyelid is up (i.e., the right shoulder was previously touched), raise the left arm:

$$G((sense\_left\_shoulder \wedge \neg right\_eyelid\_down) \Leftrightarrow \\ (Xraise\_left\_arm)) \quad (9)$$

- After raising the left arm, return to the initial state by lowering the right eyelid and the left arm:

$$G(raise\_left\_arm \Leftrightarrow (Xlower\_right\_eyelid)) \quad (10)$$

$$G(lower\_right\_eyelid \Leftrightarrow (Xlower\_left\_arm)) \quad (11)$$

Note that in this behaviour, touching the same shoulder more than one time in a row does influence the arm to be raised. After touching one of the shoulders, Maggie ignores all other sensor readings until the other shoulder is touched. When that happens, she raises the corresponding arm.

The third behaviour is also triggered by touching the head, but we allow the robot to perform random actions during the behaviour. It can be stated as "*Move arms randomly. When the head is touched, start spinning until both arms are up*". This can be translated into the following rules:

- Only start spinning when the head is touched, you are not spinning yet and both arms are not already up:

$$G((sense\_head \wedge (\neg spinning) \wedge \\ \neg(\neg left\_arm\_down \wedge \neg right\_arm\_down)) \Leftrightarrow \\ (Xstart\_spinning)) \quad (12)$$

- After starting to spin, continue spinning until both arms are up:

$$G(start\_spinning \Rightarrow (X(\\ spinning U (\neg left\_arm\_down \wedge \neg right\_arm\_down)))) \quad (13)$$

- When both arms are up, stop spinning (or continue stopped if you are not spinning):

$$G(\neg left\_arm\_down \wedge \neg right\_arm\_down) \Rightarrow \\ (X \neg spinning)) \quad (14)$$

All of these examples also include an additional formula which avoids the execution of the actions that are not referred to in the specification, i.e., in each example, a formula of the form $G\left(\bigwedge_{e \in E_o} \neg e\right)$, where E_o is the set of actions that is not mentioned in that example is also added.

V. Constructing the Supervisor

To build the supervisor, we need to define a way to compose the LTL formulas with the PN model. This is done by translating the formula φ to a (non-deterministic) Büchi automaton (BA) B_φ that accepts exactly the infinite sequences that satisfy φ. There are several methods for the construction of such automaton. In the implementation of the method we present here, we use one of the most efficient translation algorithms, LTL2BA, described in [8]. In Figure 3, we show the BA obtained for formula (14).

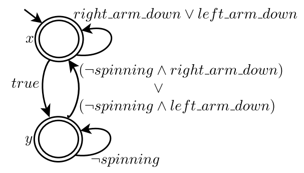

x
y
$right\_arm\_down \vee left\_arm\_down$
$true$
$(\neg spinning \wedge right\_arm\_down)$
$\vee$
$(\neg spinning \wedge left\_arm\_down)$
$\neg spinning$

Fig. 3. The Büchi automaton for LTL formula (14)

The alphabet set of this automaton is $2^{E \cup D}$, but propositional logic formulas in the disjunctive normal form (DNF) are used to describe the transition labels in a more compact way. A formula in the DNF is the disjunction ($\vee$) of a set of conjunctive clauses. A conjunctive clause is a conjunction ($\wedge$) of literals. A literal is a propositional symbol or its negation. For example, the transition label from sate y to state x means that any element of $2^{E \cup D}$ that does not contain $spinning$ and contains $right\_arm\_down$ or that does not contain $spinning$ and contains $left\_arm\_down$ is a label for the transition. We also note that the automata outputted by LTL2BA are *trimmed*, i.e., all their states can be reached and there is a path between each state and at least one accepting state.

The PN that restricts Maggie's behaviour to the one specified by an LTL formula is a PN that simulates a run in parallel of the PN model of Maggie and the observer[1] of the BA obtained for that formula, where a transition t can only fire in parallel with a transition of the observer of the BA when we are ensured that the marking to which the PN evolves satisfies the formula labelling the BA transition. This is done by looking at each transition of the PN and checking in which conditions it can fire while satisfying the transition labels of the BA. To illustrate, we will show how to compose the transitions from state x in Figure 3 with transition t_3 of Figure 2. We start by stating the facts that are guaranteed to happen after t_3 fires:

- Action $start\_spinning$ has just occurred;
- All other actions and sensor readings in E did not just occur;
- State description variable $spinning$ is $true$ - this fact is guaranteed because place p_5, the place corresponding to $spinning$ being $true$, is an output place of t_3, hence after t_3 fires it will always have a token - and $base\_idle$ is $false$ - by the same reasoning as with $spinning$.

Hence, we can define a *partial* valuation that represents all the information about events and state description variables that is guaranteed to happen after the firing of t_3:

$$v_{t_3}(\pi) = \begin{cases} 1 & \text{if } \pi \in \{start\_spinning, spinning\} \\ 0 & \text{if } \pi \in (E \setminus \{start\_spinning\}) \cup \\ & \quad \{base\_idle\} \\ \downarrow & \text{if } \pi \in D \setminus \{spinning, base\_idle\} \end{cases}$$

[1]The observer of a non-deterministic automaton G is its deterministic version, built using the known power-set construction [4].

The valuation is undefined for all state description variables that are not directly related to the firing of t_3, in the sense that none of the places representing its value receives a token with the firing of t_3. This valuation can be applied to the transition labels of the BA, which we will denote as $[\![.]\!]_{v_{t_3}}$. The result of such evaluation is:

- $true$ or $false$ if v_{t_3} provides enough information to evaluate the truth value of the label;
- The formula composed of the elements of the label for which v_{t_3} should be defined for one to unambiguously be able to evaluate its truth value.

In the case of the transitions of the BA in Figure 3:

$$[\![right\_arm\_down \vee left\_arm\_down]\!]_{v_{t_3}} = right\_arm\_down \vee left\_arm\_down \tag{15}$$

$$[\![true]\!]_{v_{t_3}} = true \tag{16}$$

$$\left[\!\!\left[\begin{array}{c}(\neg spinning \wedge right\_arm\_down)\\ \vee \\ (\neg spinning \wedge left\_arm\_down)\end{array}\right]\!\!\right]_{v_{t_3}} = \left[\!\!\left[\begin{array}{c}(false \wedge right\_arm\_down)\\ \vee \\ (false \wedge left\_arm\_down)\end{array}\right]\!\!\right]_{v_{t_3}} = false \tag{17}$$

$$[\![\neg spinning]\!]_{v_{t_3}} = false \tag{18}$$

Due to the non-determinism of the BA, we will have to analyse 3 different cases for state x:

- *Can we keep the BA is state x after firing t_3?* To guarantee that we stay in state x, we need to check in which conditions can t_3 fire while satisfying formula (15) and not satisfying formula (16), i.e., satisfying the label of the transition that goes to x and not satisfying the label of the transition that goes to y:

  $$(right\_arm\_down \vee left\_arm\_down) \wedge \neg true = false$$

  This means that this situation can never happen. Thus, we do not add transitions to the supervisor;
- *Can we take the BA from state x to state y after firing t_3?* In this case we need to check in which conditions can t_3 fire while not satisfying formula (15) and satisfying (16), i.e.:

  $$\neg(right\_arm\_down \vee left\_arm\_down) \wedge true = \neg right\_arm\_down \wedge \neg left\_arm\_down$$

  Hence we add two transitions to the supervisor (one for each conjunctive clause), with arcs equal to t_3 plus place $\{x\}$ as an input place and place $\{y\}$ as an output place (thus evolving the "observer" of the BA from the state $\{x\}$ to state $\{y\}$) and, for each of the added transitions, a reflexive-arc[2] to the place representing the corresponding literal, for example, for the transition corresponding to $\neg right\_arm\_down$, a reflexive-arc to place p_{18};

[2]A reflexive arc between a transition t and place p is a pair of arcs, one from p to t and another from t to p. Hence, t only fires when there is a token in p but it does not change the amount of tokens in p.

- *Can we take the BA from state x to both states x and y after firing t_3?* In this case we need to check in which conditions can t_3 fire while satisfying both formulas (15) and (16) , i.e., satisfying:

  $$(right\_arm\_down \vee left\_arm\_down) \wedge true = right\_arm\_down \vee left\_arm\_down$$

  Hence, we add one transition to the supervisor with arcs equal to t_3 plus place $\{x\}$ as an input place and place $\{x, y\}$ as an output place and reflexive-arcs to the places representing the literals in the obtained conjunctive clause.

Figure 4 depicts the fragment of the supervisor obtained from the analysis above, showing the three transitions created. The algorithm starts by analysing the transitions from the initial state of the Büchi automaton and adds newly reached states of the observer to a queue to be analysed next. Note that the fact that both formulas (17) and (18) are false means that when we analyse t_3 in the observer BA state $\{y\}$, no transition will be created. This is aligned with what one would expect, since we are building a PN that satisfies the natural language rule "*When both arms are up, stop spinning (or continue stopped if you are not spinning)*", hence the action *start_spinning* must be disabled whenever both arms are up, which, simplifying, is the meaning of BA state y.

The PNs obtained using this composition are then used in the feedback loop of modular supervisory control [4]. Informally, they run in parallel with the PN model of Maggie, executing the same events, and outputting the set of enabled events, given by the intersection of the labels of the active transitions for each PN supervisor. At each step, all events that are not in the set of enabled events cannot be executed by Maggie. This feedback loop was implemented on top of the Petri net executor.

In the video available at `http://bit.ly/dQqVQK`, we show both the uncontrolled behaviour of Maggie and its behaviour when being supervised by the Petri nets obtained from the specifications described here.

VI. Conclusion and Further Work

We described the implementation on Maggie, the social robot, of a method to perform PN-based supervision of robotic tasks, where the admissible behaviours are given in LTL. The similarities between natural language and temporal logic, and the fact that we can write formulas over both the events (actions plus sensor readings) of the system and a set of variables used to describe the state of the robot, allows the writing of rules for a wide array of different goal behaviours. This approach can be the basis for a method based on speech interaction, that allows the user to state rules for the robot to fulfil in natural language. Specifically for Maggie, this can be made into a game where children can see the impact of their rules in the robot's behaviour. Future work includes creating such a method, where the allowed natural language utterances are translated into LTL

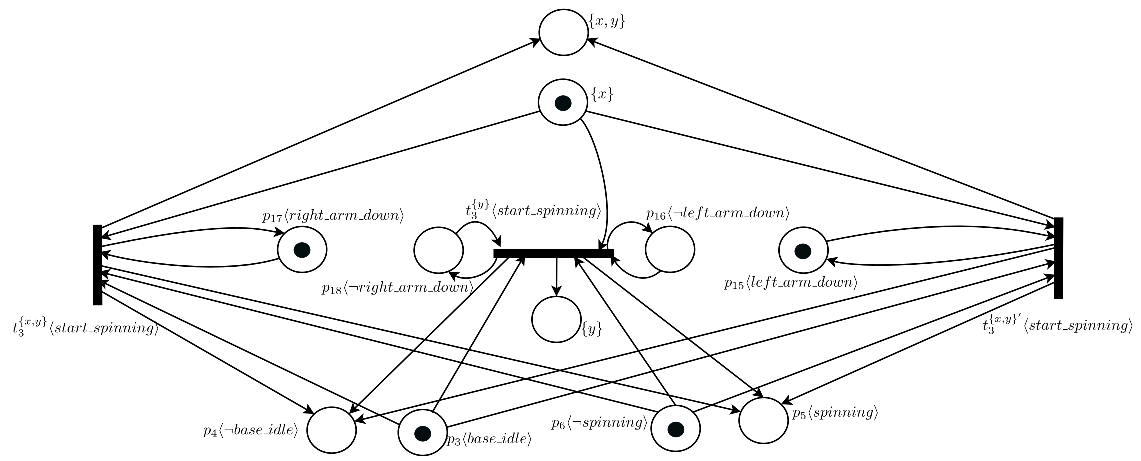

$\{x, y\}$
$\{x\}$
$p_{17}\langle right\_arm\_down\rangle$
$t_3^{\{y\}}\langle start\_spinning\rangle$
$p_{16}\langle \neg left\_arm\_down\rangle$
$p_{18}\langle \neg right\_arm\_down\rangle$
$p_{15}\langle left\_arm\_down\rangle$
$\{y\}$
$t_3^{\{x,y\}}\langle start\_spinning\rangle$
$t_3^{\{x,y\}'}\langle start\_spinning\rangle$
$p_4\langle \neg base\_idle\rangle$
$p_3\langle base\_idle\rangle$
$p_6\langle \neg spinning\rangle$
$p_5\langle spinning\rangle$

Fig. 4. A fragment of the obtained supervisor

and the method presented here is then applied. This can probably be achieved by adapting the structured English defined in [11] to Maggie's domain. Maggie already has a speech recognition skill, hence this appears to be the main issue in the development of the method. Afterwards, we intend to compare this approach with the work presented in [9], where sequences of actions are taught to Maggie by means of speech interaction, where the user exhaustively describes every possible action to be executed. We feel that, by providing a more abstract approach where a set of rules results in a certain behaviour, this approach might be more appealing to older children that already have the capabilities of inferring what might be the consequences of stating a given rule for the robot to fulfil.

REFERENCES

[1] Aris Alissandrakis, Chrystopher L. Nehaniv, Kerstin Dautenhahn, and Joe Saunders. Evaluation of robot imitation attempts: comparison of the system's and the human's perspectives. In *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 134–141, Salt Lake City, Utah, USA, 2006. ACM.

[2] R. Barber and M.A. Salichs. A new human based architecture for intelligent autonomous robots. In *Proceedings of the The Fourth IFAC Symposium on Intelligent Autonomous Vehicles*, pages 85–90, Sapporo, Japan, 2001.

[3] C. Breazeal, A. Brooks, D. Chilongo, J. Gray, A. Hoffman, C. K. H. Lee, J. Lieberman, and A. Lockered. Working collaboratively with humanoid robots. In *Humanoids '04: Proceedings of the 4th IEEE/RAS International Conference on Humanoid Robots*, pages 253 – 272, Los Angeles, CA, USA, 2004.

[4] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] International Electrotechnical Commission. Preparation of function charts for control systems: Sequential function chart. Technical Committee No. 65: Programmable Controllers - Programming Languages, IEC 61131-3, Ed 2.0, 2003.

[6] Hugo Costelha and Pedro Lima. Modelling, analysis and execution of multi-robot tasks using Petri nets. In *AAMAS '08: Proceedings 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1449–1454, Estoril, Portugal, 2008.

[7] E. Allen Emerson. *Temporal and modal logic. In Handbook of theoretical computer science (vol. B): formal models and semantics*, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[8] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *CAV '01: Proceedings of the 13th International Conference on Computer Aided Verification*, pages 53–65, London, UK, 2001.

[9] Javi F. Gorostiza and Miguel A. Salichs. Teaching sequences to a social robot by voice interaction. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2009)*, pages 797–802, Toyama, Japan, 2009.

[10] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[11] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. From structured english to robot motion. In *Proceedings of IROS '07: IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717 – 2722, San Diego, California, USA, 2007.

[12] Bruno Lacerda and Pedro Lima. LTL plan specification for robotic tasks modelled as finite state automata. In *Proceedings of Workshop ADAPT - Agent Design: Advancing from Practice to Theory, AAMAS '09: 8th International Conference on Autonomous Agents and Multi-agent Systems*, Budapest, Hungary, 2009.

[13] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[14] M.A. Salichs, R. Barber, A.M. Khamis, M. Malfaz, J.F. Gorostiza, R. Pacheco, R. Rivas, A. Corrales, E. Delgado, and D. Garcia. Maggie: A robotic platform for human-robot social interaction. In *RAM '06: 2006 IEEE Conference on Robotics, Automation and Mechatronics*, pages 1–7, Bangkok, Thailand, 2006.

[15] V. A. Ziparo, L. Iocchi, D. Nardi, P. F. Palamara, , and H. Costelha. Petri net plans: a formal model for representation and execution of multi-robot plans. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 79–86, Estoril, Portugal, 2008.

Planning under Uncertainty for Search and Rescue

Ana Rita Mendes, Matthijs T.J. Spaan and Pedro U. Lima

***Abstract*— Partially observable Markov decision processes (POMDPs) provide a framework for planning under uncertainty. We present an application of POMDPs in a search and rescue situation. Specifically, the aim of the problem is to find victims in a disaster environment. We will define, implement and test POMDP models that suit the problem and its characteristics. Part of the environment models and their features will be learned (by letting the robot interact with it) and taken into account when building the model. We compare different modeling approaches and examine their trade-offs in modeling power and computational complexity.**

I. INTRODUCTION

Disasters, and rescues from disasters represent a worrying social issue. Fire scenes, major traffic accidents or building explosions are just a few examples of situations that represent very hazardous environments, where many human lives are endangered (not only victims, but rescue teams also). More often now, robotics can help in these environments, even though most tasks are still the responsibility of humans. In order to be helpful, robots must learn how to behave in these environments. However, catastrophes and unplanned situations are not readily available for training. Hence, we will instead use a reliable simulator, USARsim (Unified System for Automation and Robot Simulation). Due to the existence of RoboCup Rescue Competition [2], many maps for rescue situation are available for this simulator, as well as stable code built by the participating teams in order to act in these environments.

In a simulated disaster environment, robots should be able to look for the victims in an optimal way even though their positions are unknown. Due to the uncertainty of victim positions, as well as other common uncertainty sources in robotics (sensors, actuators, etc.), the problem will be modeled as a Partially observable Markov decision process (POMDP) [4]. POMDPs form a powerful framework for planning under uncertainty and have been gaining in popularity in robotics [10], [9], [5]. For instance, in [8] POMDPs were applied to the problems of robot localization and navigation.

Even when a topological map of the site is known a priori, it should be kept in mind that the map represents a building in a rescue situation and as a result some of its structure might have changed. As such, we will learn some of the models of the POMDP, in particular the parameters of the topological map.

This work was funded by Fundação para a Ciência e a Tecnologia (ISR/IST pluriannual funding) through the PIDDAC Program funds and was supported by project PTDC/EEA-ACR/73266/2006.

All authors are with the Institute for Systems and Robotics, Instituto Superior Técnico, Lisbon, Portugal.

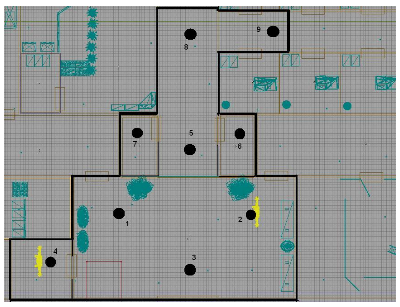


Fig. 1. USARsim map portion along with the nodes that will represent it. Moreover, two victims can be seen in nodes 2 and 4.

We will build our work on both code and a map from the competition of 2008. The code belongs to the Jacobs University team [6] and it will be responsible for all the low level control. The map represents the interior of an office building with several injured victims for the robot to find, shown in Fig. 1. A POMDP model will be built, learned and tested in this environment, interacting with the controller code, giving the robot instructions on what to do, while receiving observations of the environment (only of some of its features).

The rest of this paper is organized as follows. First, Section II provides the necessary background on POMDP models. Section III describes how we learn the transition models for the robot, while Section IV details the POMDP models we define for our victim-localization task. Section V presents some experiments using USARSim, and Section VI concludes.

II. BACKGROUND

A POMDP models an agent acting synchronously with the environment, aiming to find the best way to act at every time step. In order to do so, at every time step a reward (or a penalty) is given to the agent, so it can understand its goal, what it should do and what it should avoid doing. POMDPs can be described by [4]:

$\mathcal{S}$	a discrete and finite space state;
$\mathcal{A}$	a finite set of possible actions, where $\mathcal{A}(s)$ represents the actions available at state $s \in \mathcal{S}$;
$T_{sas'}$	the transitions probabilities, the probability of going to state s' when in state s and given the action taken a, $p(s'\|s,a)$;
R_{sa}	a reward function, giving the immediate reward for taking the action a when in state s;
Ω	the set of all possible observations;
$O_{oas'}$	the observation model, representing the probability

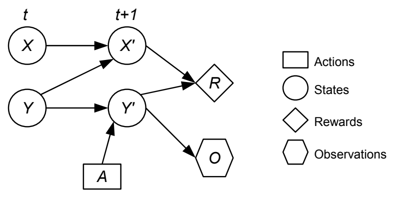


Fig. 2. Dynamic Bayesian network for a POMDP with two state variables.

of observing o while being at state s' after taking action a, $p(o|s',a)$.

Since the state is not known, to be able to make decisions, some kind of memory would be necessary to keep track of (possibly) the entire history of the process. However, using a probability distribution over all of the states has the same information as keeping track of the complete history. That probability distribution is called a belief and the POMDP is simply an MDP with belief states instead of nominal states.

When an action is performed and an observation received, the distribution has to be updated to reflect the new knowledge. More formally, let $b(s)$ denote the probability assigned to world state s by belief state b. Updating the belief is computing b' which can be obtained given the previous belief state b, the action executed a and the observation o through:

$$b'(s') = P(s'|o,a,b) = \frac{O(s',a,o)\sum_{s\in S} T(s,a,s')b(s)}{P(o|a,b)}, \quad (1)$$

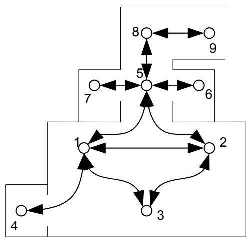
where $P(o|a,b)$ can be seen as a normalizing factor in order for b' to sum to one [4].

To represent and solve models in this paper we will use Symbolic Perseus [7], which is a point-based value iteration algorithm for POMDPs. Moreover, to represent transition, observation and reward functions in a compact way they are represented as Dynamic Bayesian networks (DBNs). A DBN is an acyclic graph that allows representation at the variable level instead of the state level [3]. Their graphs represent two consecutive time steps, with nodes as the variables (state, observation and reward variables) and connections as the probabilistic dependencies. Furthermore, Symbolic Perseus allows for exploiting context-specific independence by representing the conditional probabilities as Algebraic Decision Diagrams (ADDs) [1]. A DBN representing a simple POMDP is shown in Figure 2.

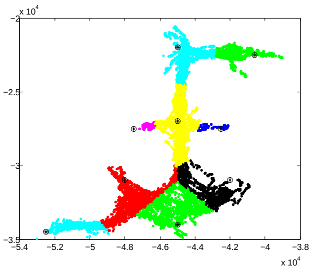
III. Learning robot transition models

In general, POMDP solvers require the transition, observation, and reward models to be known a priori. In this paper, we focus on methods for defining the transition model for a robot, by learning the transitions between nodes in a topological map. Figure 3(a) shows the topological map representation corresponding to Fig. 1.

Defining the transition model is one of the most important tasks when creating a POMDP model. The closer the transition model is to reality the better the model will work when in real situations. In this paper the environment is available for testing and learning the transition model, allowing the



(a) Topological map.



(b) Transition model data.

Fig. 3. (a) Topological representation of the map of Fig. 1 with 9 nodes and their connections. (b) Recorded points along paths used to learn the transition model. Nodes are printed in different colors and node centers are marked with a black circle.

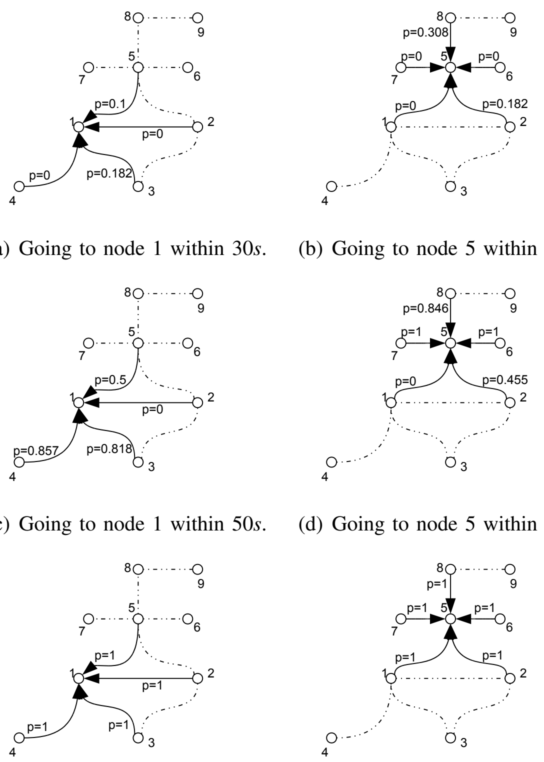
POMDP model to reflect that knowledge and the planning to be more accurate.

In order to learn the model, both the environment and the controller are used. The robot explores several times the paths between all the different considered nodes. In each run the time taken to reach the destination is recorded and with several runs is possible to obtain a good number of executions for each path in order to calculate reliable average results. Positions and time stamps were recorded along the entire path, in order to keep track of the whole path. Figure 3(b) illustrates the recorded positions along the tests.

The time step allowed for each move has to be defined and it should be chosen carefully. Having smaller time-steps results in larger planning horizons as more steps will be needed when executing the plan. On the other hand, using longer time-steps results in less responsive and less accurate plans. With a time step established all the paths are checked for the position of the robot at that time. The probabilities of going from one node to another adjacent node can be extracted from the recorded paths. Several time steps were used to obtain a few different transition models. Figure 4 shows the transition probabilities for two different actions and different time steps. It is very clear to see the difference in some paths for which $30s$ did not allow the robot to leave the node but with $50s$ it does indeed get to its destination with some probability. Moreover, with no time limit all the paths were executed successfully. However, in a more general case a robot will obviously not be able to reach each destination with probability 1.

Figure 5 represents the average of the probability of arriving at the destination node, for all paths, based on which we choose the time step to be used for the real tests. With smaller time steps such as $20s$ or $30s$ for most paths the robot would never get to the goal while with $>60s$ many paths were successfully completed. As we want the time step to be short to allow for quicker reactions, and given the POMDP can handle uncertainty in the transitions, we define the time step to be $50s$.

With the probabilities defined they have to be integrated in the generation of the model. However, probabilities that are 0 or 1 will not be used, as this would be a too strong statement,



(a) Going to node 1 within 30s. (b) Going to node 5 within 30s.

(c) Going to node 1 within 50s. (d) Going to node 5 within 50s.

(e) Going to node 1, no time limit. (f) Going to node 5, no time limit.

Fig. 4. Transition probabilities for 2 actions with 3 different time steps.

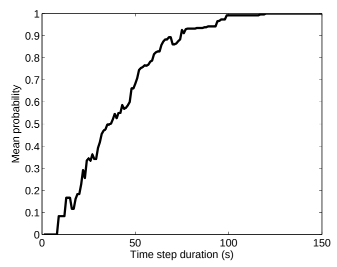
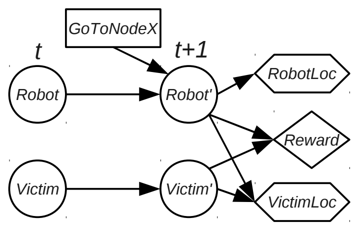


Fig. 5. Mean probability of arriving at the desired node for time step from 1 to 150s.

given the fact that we are considering an abstraction of reality. Keeping in mind that these environments always carry a great deal of uncertainty all probabilities that equal 1 will be set to a slightly smaller value and 0 probabilities will be set to a very small value, but larger than 0.

IV. POMDP MODELS FOR SEARCH AND RESCUE

In this section we will develop POMDP models for the victim localization task referred to in the introduction, which are based on the learned transition model as described in Section III. Every problem can be seen from different perspectives and therefore defined and designed through different approaches. As such, for this specific planning problem, two models will be developed and explored. We assume that a robot can detect victims with high probability when both are in the same node, however, from a POMDP point of view, this can easily be relaxed.



(a) DBN representation of the POMDP.

Variable Type	Name	Domain
State	Robot	9 nodes
State	Victim	9 nodes
State	Seen	yes/no/done
Observation	RobotLoc	9 nodes
Observation	VictimLoc	9 nodes+no observation

(b) POMDP variables.

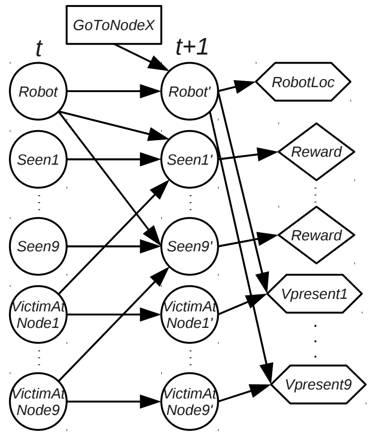
Fig. 6. POMDP representation for Model A.

A. Model A - victims as variables

First we consider a model in which the number of victims present in the environment is assumed be known. A Dynamic Bayesian network representing the model is shown in Figure 6(a). Considering the map presented in Figure 3(a) with one victim in one of the nodes (any node) the aim of the robot is to find the victim. With this setup, the state variables will be the location of the robot and each victim (only one for one victim). Moreover, a third state variable, *seen*, will be needed (one for each victim) to keep track of which victims have been detected and allow a correct distribution of rewards. Victim variables and the robot variable each have 9 possible values (as many as the number of nodes). *Seen* variables have 3 possible values, *no*, *yes* and *done*.

The transition model for each *seen* variable can be described by: it starts with with value *no*, once the corresponding victim is detected the value changes to *yes* and in the next time step it changes to *done*. A victim is considered to be detected once the robot and the victim are in the same node. Victim variables have the identity matrix as transition model, as they never change their state (i.e., victims remain in the same node for the entire problem). The transition model for the robot has to be learned from interacting with the environment as detailed in Section III. Regarding observations, the robot is considered to localize itself without any error (taking into account the controller used for testing, that is a valid assumption). For the victims, each has an observation model according to which if the robot and the victim are in the same node the victim is detected at that node with 99% chance, otherwise, the victim is not detected.

The reward function depends only on *seen* variables, and is the sum of a reward function per *seen* variable, as they are independent. When the variable is in state *no* or *done* the robot receives a small negative reward, when it is in state *yes*, the robot receives a large positive reward. Due to the fact that these variables only remain in state *yes* for one time step and can never go back to it, only one positive reward can be received per victim. In this way, we are preventing the robot from finding the same victim more than once, encouraging



(a) DBN representation of the POMDP.

Variable Type	(Quantity) Name	Domain
State	(1) Robot	9 nodes
State	(9) VictimAtNode	yes/no
State	(9) Seen	yes/no/done
Observation	(1) RobotLoc	9 nodes
Observation	(9) VictimPresent	yes/no

(b) POMDP variables.

Fig. 7. POMDP representation for Model B.

it to look for other victims.

To sum up, the model is described by its variables and their possible states (shown in Table 6(b)), the transition model (for variables *Robot* and *Seen*), the observation model (for variables *Robot*, *Victim*) and the reward function (for variable *Seen*).

B. Model B - Nodes as Variables

Another approach to the same problem with the same map (Figure 3(a)), is to consider the nodes as variables, instead of the victims. Each node is represented by a variable, e.g., *VictimAtNode01* for node one. This variable can assume two values, *yes* and *no*, representing the presence or absence (respectively) of a victim in that specific node. This representation is very useful for when the number of victims present is unknown. The previous model assumed the number of victims were known from the start and that is not necessarily the case.

The *Seen* variables are still needed to keep track of what has been seen before and what is new (allowing the correct distribution of rewards), representing which nodes have been seen. As there are 9 *VictimAtNode* variables, 9 *Seen* variables are needed, with the same 3 possible values, *yes*, *no* and *done*. This leads to 19 state variables, one from the robot, 9 from *VictimAtNode* variables plus the 9 *Seen* variables.

The transition model of the robot is the same as for model A (explained in Section III). The presence or absence of victims at a specific node does not change over time, as victims do not move. As a result, variables concerning victims (*VictimAtNode*) have identity as transition model. As the *Seen* variables keep track of visited nodes, the transition model is slightly different now. All nodes start with *Seen* in the *no* state. Once a node is visited if there is a victim in it *Seen* variable will switch to *yes* and in the next state it will switch to *done*. If a victim is not present, the node is most likely considered visited and without a victim, so with 99% chance *Seen* changes to *done* (staying in *no* with 1% to contemplate the small possibility that the victim was there but the robot did not see it).

Regarding observations, the robot is once more considered to localize itself without any error. For the victim nodes the idea is the same as with victim variables in model A. If the robot is in node x and a victim is there too ($victimAtNode_x = yes$) the observation for $VictimPresent_x$ will be *yes* with 99% chance (and *no* with 1% chance). The reward function is the same as for model A. Each *Seen* variable is responsible for a reward. A negative reward for states *no* or *done* and a big positive reward for state *yes*, which happens only once (or never) per variable.

To sum up, the model is described by the variables and their possible states (shown in Table 7(b)), the transition models (for variables *Robot*, *Seen1*, *Seen2*, ..., *Seen9*), the observation model (for variables *Robot*, *VictimAtNode1*, *VictimAtNode2*, ..., *VictimAtNode9*) and the reward function (for variables *Seen1*, *Seen2*, ..., *Seen9*). A Dynamic Bayesian network representing the model is shown in Figure 7(a).

V. Experiments

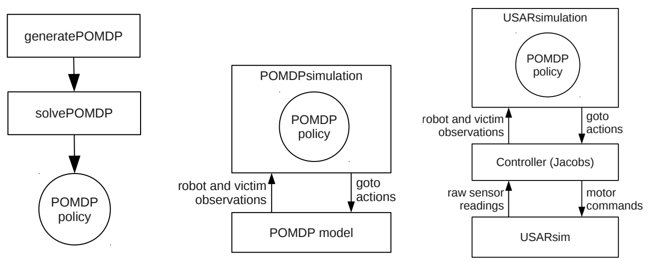
The basic structure of the whole system needed for the experiments is represented in Figure 8. The models defined in the previous section are generated by the *generatePOMDP* function. Then the model is solved (through *solvePOMDP*), which runs Symbolic Perseus [7] to compute the policy. There are two ways for testing a POMDP policy. The *POMDPsimulation* function, which simulates the environment, i.e., all variables are simulated according to their specifications and the POMDP policy is tested within its own model. The *USARsimulation* function, on the other hand, is ready to deal with real environments, that is, this function interacts with the Jacobs controller, which in its turn operates on the USARsim simulation.

With these two functions it is possible to observe and compare the estimated behavior with the simulated real behavior. The more similar their behaviors are the better the model describes the environment. That is, with more realistic and accurate transition and observation models reflecting the behavior of the system it is possible to obtain better plans. For using the models with the USARsim environment, the basic structure of the whole system is represented in Figure 8. Both the *USARsimulation* function and the controller code were adapted in order to be able to communicate with each other.

In this section the models will be tested for their behavior. Moreover, model B, due to its large number of variables is also studied regarding the model complexity.

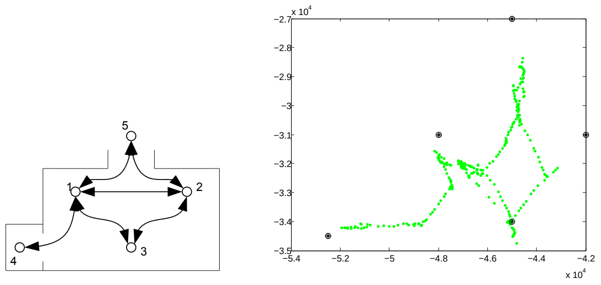
A. Experiments with model A

Model A, as described in Section IV, is solved through the *solvePOMDP* function. Given its reduced number of variables its complexity is low and a policy is computed



(a) Solve POMDP. (b) Evaluate with POMDP models. (c) Evaluate using USARSim.

Fig. 8. Experimental setup, showing the different components of the evaluation setup.



(a) Map with 5 nodes. (b) Path during the whole run.

Fig. 9. Testing area map with the positions of the robot during the test.

TABLE I
TESTING MODEL A WITH A VICTIM IN NODE 2 AND THE ROBOT STARTING IN NODE 8.

(a) USARsim testing environment.

Step Nr.	Action	Robot Loc.	Victim Loc.	Reward	Cum. Reward
1	goTo9	09	No	-0.65	-0.65
2	goTo8	08	No	-0.59	-1.24
3	goTo5	05	No	-0.52	-1.76
4	goTo6	06	No	-0.43	-2.19
5	goTo5	05	No	-0.97	-3.16
6	goTo7	05	No	-0.97	-4.13
7	goTo7	07	No	-0.29	-4.42
8	goTo5	05	No	-0.95	-5.37
9	goTo1	05	No	-0.95	-6.32
10	goTo1	01	No	-0.07	-6.39
11	goTo4	01	No	-0.93	-7.32
12	goTo4	04	No	-0.34	-7.66
13	goTo1	04	No	-0.90	-8.56
14	goTo1	01	No	-0.90	-8.46
15	goTo2	02	02	50.000	40.54

(b) POMDPsimulation simulation environment.

Step Nr.	Action	Reward	Robot Loc.	Victim Loc.	Cum. Reward
1	goTo9	-1.00	09	No	-1.00
2	goTo8	-0.65	08	No	-1.65
3	goTo5	-0.59	05	No	-2.24
4	goTo6	-0.52	06	No	-2.76
5	goTo5	-0.43	05	No	-3.19
6	goTo7	-0.97	07	No	-4.16
7	goTo5	-1.00	05	No	-5.16
8	goTo2	-1.00	05	No	-6.16
9	goTo2	-1.00	05	No	-7.16
10	goTo2	-1.00	05	No	-8.16
11	goTo2	-1.00	02	02	-9.16
12	goTo4	50.00	02	02	40.84

within a couple of minutes. The model is then submitted to a few tests, in order to check its behavior in the USARsim simulated environment. Moreover, this behavior is compared with simulations with *POMDPsimulation* where the position of the victims is given as an input, so both situations can be compared. Table I shows the behavior of the system for one test situation compared with the simulated behavior.

It is possible to observe some differences between the plans executed in USARsim and in *POMDPsimulation*. As we are dealing with uncertainty this is not unexpected, but comparing globally the number of iterations, the rewards obtained and the robot's behavior in both situations, the model (and mainly the transition probabilities) seems to represent reality with enough accuracy to have a good plan.

B. Experiments with model B

Model B, due to the high number of variables (and states), is too complex to solve with the specified settings in Section IV. The computational resources (RAM memory) needed to run the function *solvePOMDP* on this model are not available. In model B, as the number of variables grows with the number of nodes (at a rate of two state variables and one observation variable per node), large maps become impossible to solve, due to memory limitations. As such, the model was tested in simulation (with *POMDPsimulation*) and in USARsim (with *USARsimulation*) with a smaller version of the problem, for only 5 nodes, instead of 9 (Figure 9(a)).

The robot started in node 3 and victims could be found in nodes 4 and 2. Table II shows both runs, simulated with *POMDPsimulation* and tested in USARsim with *USARsimulation*. Figure 9(b) shows the path the robot executed during the USARsim test. It has to be reminded that the transition model used for this model is the same as for model A, as the environment is the same. Once more, the model seems to represent reality well enough for the policy to execute a good plan.

Reducing complexity can also be accomplished by lowering the number of observation variables and/or number of observations. This solution implies that some changes are made to the model. Two situations are studied in order to decrease the complexity of the model. In the first situation (Model5) the number of observation variables is decreased from as many as the number of nodes to only one. This variable represents observations for all nodes, thus the number of possible observations increases as it represents observing the victim in any of the nodes and not observing the victim. This solution is slightly limiting the problem, as it means in each time step no more than one observation is possible.

The second situation (Model6) is accomplished simplifying the previous one (Model5). Still only one observation variable is used to observe any victim in any of the nodes, however, that is achieved only through two possible observations, hence reducing considerably the number of

TABLE II

TESTING MODEL B, WITH 5 NODES, WITH VICTIMS IN NODES 2 AND 4 AND THE ROBOT STARTING IN NODE 3.

(a) USARsim testing environment.

Step Nr.	Action	Robot Loc.	Victim Pres.02	Victim Pres.04	Reward	Cum. Reward
1	goTo1	01	No	No	-4.87	-4.87
2	goTo3	03	No	No	-4.89	-9.76
3	goTo2	02	Yes	No	46	36.24
4	goTo5	05	Yes	No	-4.91	31.33
5	goTo1	03	Yes	No	-5	26.33
6	goTo1	01	Yes	No	-5	21.33
7	goTo4	01	Yes	No	-5	16.33
8	goTo4	04	Yes	Yes	46	62.33

(b) POMDPsimulation simulation environment.

Step Nr.	Reward	Action	Robot Loc.	Victim Pres.02	Victim Pres.04	Cum. Reward
1	goTo1	-5	01	No	No	-5
2	goTo3	-4.87	03	No	No	-9.87
3	goTo2	-4.89	02	Yes	No	-14.76
4	goTo1	46	01	Yes	No	31.24
5	goTo4	-5	04	Yes	Yes	26.24
6	goTo3	46	04	Yes	Yes	72.24

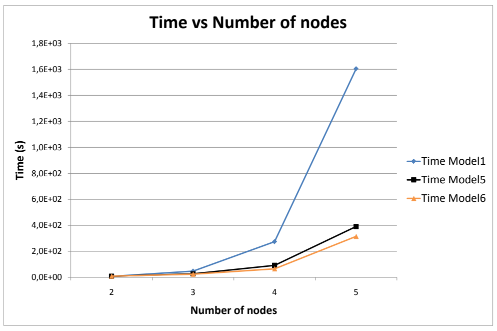


Fig. 10. Graphic representing the experiments for the two test cases with fewer observation variables.

observations. The possible observations are seeing or not seeing a victim. As such, information on the victim's position is not so clear. The robot is assumed only to detect a victim when in the same node as the robot and seeing a victim in a close node would be, in this model, interpreted as seeing it in the node the robot is in, as the observation for seeing victim does not allow any information regarding the position.

The complexity was measured through the time taken to solve the POMDP. Figure 10 allows the computation times for Model5 and Model6 to be compared with Model1. It is possible to observe that decreasing the number of observations represents a much lower complexity. However, the models representing the whole desired area (9 nodes) are still uncomputable given the available resources.

VI. CONCLUSIONS

In this paper POMDP models were developed for planning under uncertainty in search and rescue situations, in which a robot has to locate victims. Dealing with uncertainty is an area of great scientific interest, and can have a positive effect on socially relevant planning problems such as search and rescue.

We showed that the POMDP approach can be used in a victim-localization task. However, care needs to be taken when developing the POMDP models, in order to attain the correct level of abstraction, in order to keep the models manageable for solving. Two different POMDP models were developed, one which allows for a much faster computation but with the downside of needing some a priori knowledge regarding the number of victims present at site. The other model, on the other hand, allows for an arbitrary number of victims but with computational time growing very fast with the number of nodes in the map.

Transition probabilities for the robot were learnt, allowing the POMDP models to take into account environment features and the characteristics of both robot and controller. This is crucial for planning problems and it represented a major concern in this paper, as it models the way a plan would be executed by the system.

In future work, we would like to take into account the fact that in real rescue situations the exact same environment would never be available for learning the transition model previously. Even if the transition model was learned for the same area, some walls could be down, some new paths open and some hazardous obstacles along the way. Designing POMDP algorithms that can efficiently changing transition models remains an important and relevant challenge. Furthermore, a clear future direction is to learn as well the robot's observation model.

REFERENCES

[1] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. In *Proc. Int. Conf. on Computer-aided Design*, 1993.

[2] Stephen Balakirsky, Stefano Carpin, Alexander Kleiner, Michael Lewis, Arnoud Visser, Jijun Wang, and Vittorio Amos Ziparo. Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue. *Journal of Field Robotics*, 24:943–967, 2007.

[3] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence Journal*, 5(3):142–150, 1989.

[4] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[5] H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

[6] Yashodhan Nevatia, Todor Stoyanov, Ravi Rathnam, Max Pfingsthorn, Stefan Markov, Rares Ambrus, and Andreas Birk. Augmented autonomy: Improving human-robot team performance in urban search and rescue. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2103–2108, 2008.

[7] Pascal Poupart. *Exploiting structure to efficiently solve large scale partially observable markov decision processes.* PhD thesis, University of Toronto,, 2005.

[8] N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

[9] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[10] Nikos Vlassis, Geoff Gordon, and Joelle Pineau. Planning under uncertainty in robotics. *Robotics and Autonomous Systems*, 54(11), 2006. Special issue.

Autonomous Driving Competition: Perception Approaches used in the ATLAS Project

M. Oliveira, V. M. Santos, *Member, IEEE*

***Abstract*— Autonomous navigation in real roads has been a concern for several years now, especially for the Intelligent Transportation Systems community. Several interesting results have been obtained since the early 90's, but the problem is so vast and manifold that even today it keeps many persons at academic and industrial levels engaged. These researchers are focused in the search for more, and more general, approaches for road and lane detection, traffic lights and many other perturbations on the normal flow of perception from the road and its entourage. Perception and navigation are the main two components of this huge problem, where the second depends largely on the performance of the first. In Portugal, a competition for Autonomous Driving was created in 2001. It intended to promote the development of solutions for these perception and navigation problems. Currently, the competition accounts for several challenges and a University project named ATLAS has developed a series of robots that have outperformed all the competitors for five years in a row. The paper presents the main techniques and algorithms that lead to this success and formulates the bases to migrate the solutions to real road conditions. Perception of road and its features are given the main focus.**

I. Introduction

Autonomous navigation of cars and other vehicles on real roads is perhaps a dream as old as the modern automotive industry. Although many interesting results have emerged in the last 15 years, no definitive answer exists in terms of robust perception and even less in safe and robust driving. To promote developments and contributions to near the reality to that dream, several groups have devised many sorts of activities among students and research communities. One of such groups proposed in Portugal, in the late 90's, a competition 0 for autonomous robots where some sort of road plunged with obstacles and other perturbations was to be traversed intelligently by self-contained machines. The Portuguese Robotics Open (ROBOTICA) was then created and the competition of Autonomous Driving is now its oldest competition, among others created meanwhile.

The main goal of the competition is to complete two rounds of an 8-shaped path simulating a real road (Figure 1), although with some controlled parameters such as good line definition on the floor. Nonetheless, the challenge is quite demanding when details are observed closely. In its last level of complexity, the competition comprises a zebra crossing area defining precise stopping areas in case the traffic lights, suspended above it, force the stopping. Random generation of traffic signals also with random values for their duration make the problem demanding for entry level competitors. That is however negligible when compared to a tunnel where light conditions change dramatically and when the robots reach the road maintenance area where an alternative road delimiter is used simulating temporary path detours; in that case, the orange and white stripes and cones override the normal road white lines. Even more challenging is to have, above all this, obstacles in unknown positions and, finally, at the end of the trial, complete the challenge with the *pièce de resistance* which is a compelling parking area located somewhere outside the road track having two places to park but one of them randomly occupied by another obstacle. The entire challenge requires abundant vision systems and other techniques both for perception and navigation. Very few robots in the ten editions of the competition have managed to complete all the challenges. Being able to cope with these many challenges certainly is an indicator that real roads in real environments can also be dealt with. The task with real roads is obviously more difficult, but the problems share principles and methods to reach the solution.

M. Oliveira is a PhD Student at University of Aveiro. The author thanks FCT for funding under the grant SFRH/43203/2008. Email: mriem@ua.pt.
V. Santos is with the Department of Mechanical Engineering and the Centre for Mechanical Technology and Automation of the University of Aveiro. Email: vitor@ua.pt.

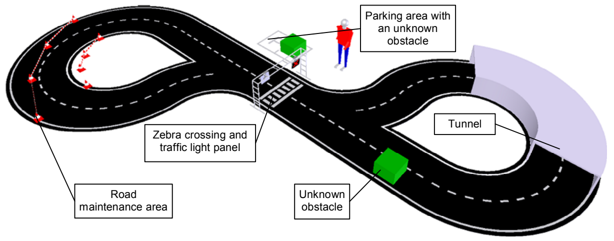

Parking area with an unknown obstacle
Tunnel
Zebra crossing and traffic light panel
Road maintenance area
Unknown obstacle

Figure 1 – Model of the autonomous driving competition environment.

The remainder of the paper introduces some related work in road perception, and then continues with the techniques used by the ATLAS robots, especially in road segmentation and its embedded obstacles and accessories. Conclusions and perspectives for future work are drawn at the end.

Since 2004, in seven editions, the ATLAS robots have obtained, 2 third places, 3 second places, and 5 first places in a row in the Portuguese national competition.

II. Previous work on road perception

Several approaches have been attempted in order to solve the problem of extracting the road using visual information only. There are two separate threads in what road detection is concerned. Road may be detected based on road color uniformity, or delimiting lines can be looked up.

Some authors try to explore morphological operators in order to extract the road or lines. Liu *et al.* make use of Canny and Sobel operators to remove shadows from the image [5], while Zhang *et al.* employ Hough Transforms to perform line detection [6]. Some others employ neural networks. Pomerleau uses neural networks that process a low resolution image of a particular area of the road and classify the image according to its similarity to several hypothetical models of road curvature [3]. The low resolution image is extracted from a trapezium, which is positioned vertically according to the current vehicle's speed. The horizontal size of the trapezium is decided on top of perspective transformation considerations, so that a particular region of the image can be remapped to a top view of the road. Foedisch *et al.* [7] also use neural networks to classify a set of pixels as "road" or "not road" based on their colour similarity to some example pixels taken from sub-windows of the image where the road is most likely to be. The positioning of such windows is, obviously, critical to the systems performance. For each analyzed pixel, the neural network's inputs are the pixels RGB colour (down-sampled) and the pixels position in the image.

In Italy, Bertozzi and Broggi also use perspective transformation to obtain a bird's eye view of the road. In more recent years, authors have made use of state of the art techniques like directional filters, namely Gabor filters. The convolution with these filters extracts several trends of macroscopic features which are then accumulated for different directions [8]. Intel's cooperation with Stanford University has also resulted in some road navigation state of the art techniques. The laser scanner defines a polygon that is sure to comprise the nearby road. The color of the pixels inside that polygon is then used as a seed for learning and modelling clusters of road-colored pixels 0. In the scope of the DIPLODOC project, Lombardi *et al.* have developed a model switching architecture that uses pixel color to find the most similar model from a finite database [10].

III. The robots from the ATLAS Project

To face the challenge posed by the ROBÓTICA Autonomous Driving competition, the ATLAS project was started in 2003, and besides all the hardware and structural issues, the main challenges, still the relevant ones in the present time, are the perception of the road and its accessories, and navigation based on that perception. Figure 2 shows the two ATLAS robots: series Atlas-2000 and series Atlas-MV. The earliest robots used either a single camera with mirrors, or two independent cameras to monitor the road. Latest versions use multi-camera images which are merged to form wider images.

For road navigation, the ATLAS robots use two cameras with the purpose of obtaining a very wide angle image. Cameras are not tightly registered so, in order to merge the

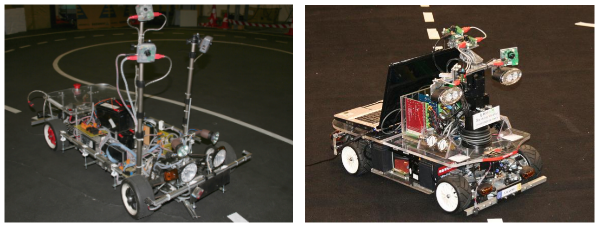


Figure 2 - Examples of robots from the ATLAS project. Atlas 2009 (*left*) Atlas-MV (*right*).

images from the cameras, image transformations have to be accounted for. Furthermore, since each camera possesses wide angle lenses, a relevant amount of distortion occurs. At first glimpse, the full modelling of both the lenses parameters and the perspective transformations would be expected in order to obtain a perfect (geometrically accurate) merging of the information. However, in a first approach, the authors have come to the conclusion that these calibration procedures were fairly demanding and could be easily lost due to unstable camera physical fixations and other hardware issues at the time the system was developed. Based on this observation, an approach was attempted where a rough image merging would be enough. In fact, image merging is required to be accurate only if precise geometrical features are to be extracted from the combined image. That is not the case on this approach for road navigation. The rough image combination suits well if the subsequent road filters are only minimally affected by it. Since only a rough combination of both images was required by the algorithm, a manual calibration of the distortion parameters is performed for each camera in order to combine them; this procedure is executed offline, therefore without any effect on the efficiency of the navigation algorithm. An interactive application was then developed to allow this manual calibration and its interface. It should be noted that this method is entirely empiric, since for more rigorous combinations a more precise perspective transformation should also be taken into account. Nonetheless, this methodology gave good results [13].

In more recent developments, instead of using two fixed cameras as in earlier versions, one of the latest ATLAS robot is equipped with a multi-camera system mounted atop a pan and tilt unit to allow for more complex perception algorithms and strategies, such as active vision. The multi-camera perception unit includes four cameras and a servo-actuated pan and tilt unit, as shown in Figure 4. Two of the cameras, Cam_0 and Cam_1 of Figure 3, are positioned on the far sides of the unit, and are equipped with 2.1 mm focal length wide angle lenses. They are intended solely for navigation and should therefore provide a view of the road's full width. For these two cameras, the supporting structure allows to position the following parameters: vergence, torsion about the principal axis and distance to the unit's centre. Cam_2 and Cam_3 add up to define a foveated system. Cam_2 is also equipped with a wide angle lens. It's intended to have a wide field of view so that it can effectively search for known objects of interest. It is also called the peripheral camera.

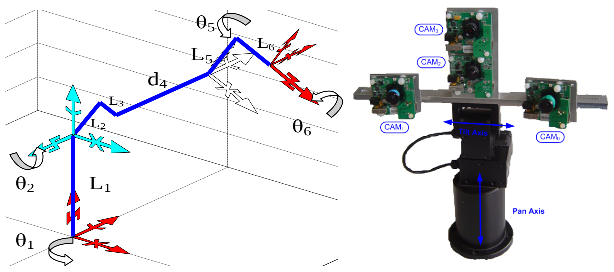


Figure 3 - Kinematics chain (*left*) for the “left” part of the Multi-camera active perception unit (*right*).

Aligned on the same vertical axis is Cam$_3$, whose large focal distance lens allows the extraction of a high detail view of a particular object. This is also called the foveated camera. This foveated set-up is intended for future developments.

For the purpose of navigation and more precise geometric evaluation of the environment, each camera undergoes a camera distortion correction namely barrel and similar intrinsic issues. The procedure uses the classic chessboard approach available in several software packages, as in the OpenCV libraries (*http://opencv.willowgarage.com/wiki*) used in this project.

The entire pan-and-tilt unit along with its cameras can be described as several kinematics open chains and can be modelled using the Denavit-Hartenberg (DH) notation for serial open kinematics chains [15], as illustrated in a partial model in Figure 3 (*left*). A complete description of the model can be found in [12]. After deriving the model of the perception unit, the transformation and merging of the camera’s images is then possible. This transformation is performed for each camera by mapping image pixels onto the ground plane, Z=0. This generates a distribution of the input images’ pixels in the ground plane, as depicted in Figure 4. However, some pixels of the input images may not have their correspondent real point on the ground plane. This depends on the camera’s lens vertical opening angle and on the current tilt angle. Additional conditions are defined to cope with these and other constraints [12].

IV. Perception on the ATLAS robots

Within the ATLAS project, two major algorithms have been devised for road segmentation. One is based on the road homogeneity and the existence of at least one border line [13], and a second one is based on line extraction from perspective corrected images [12].

A. *Road Homogeneity*

This approach does not seek to extract the road border lines but the space between them, which is the actual road to navigate in. Border line connectivity is important to obtain the limits of the track: having both lines in the image is the ideal situation, but one line still allows the partial definition of the track and the absence of both lines results in an unlimited track (usually of short term effect). The definition of a virtual horizontal line limits the upper part of the image

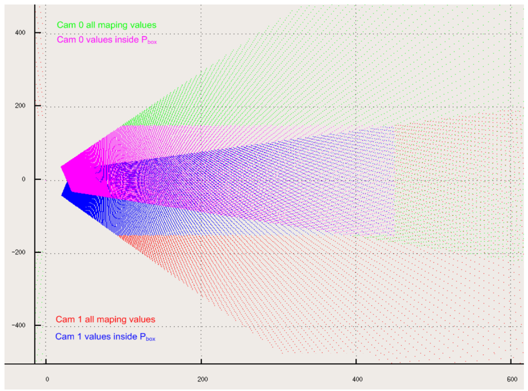


Figure 4 - A map of the projection of the cameras onto the ground plane.

and completes the plausible area for the vehicle to traverse Figure 5. In fact, the lateral lines, the horizon line and the bottom of the image form some sort of free space trapezium, which is a concept found also in other works [3][8]0. The advantage of the process is that with the resulting information the issue becomes a matter of forbidden/allowed space. The final image holds important information for the navigation process since it describes which pixels are inside the road and which are not. Furthermore, the pixels on the edge of the road may easily report the curvature of the road. The area of the road can also be easily calculated, if needed. Naturally, some special situations of the road trapezium occur and are dealt accordingly [13].

B. *Lane marker Detection*

Line detection and validation has been used as an alternative to the earlier road segmentation algorithms. The process takes advantage of the top view of the road obtained trough perspective transformation and multi-image merging, as described before. The process consists of building up a set of line candidates, and then tests some statistical indicators against a model of what a line should be. For quick comparison, the model is defined based on those same statistics according to the values typically found. Further details are available in [16]. Examples of line detection (highlighted in green) can be seen in Figure 6 and Figure 7.

C. *Obstacle Detection*

In the 2006 edition of the ROBOTICA competition, the obstacles were painted in white. The obstacle was detected as an integrant part of the line and, therefore, no exceptional processing was required to map the obstacle and define the new road. In the following editions of the competition, the obstacles were made in green colour. The obstacle must then be segmented from the background, which was done using simple HSV colour filters. However, the segmentation is never optimal. The reason for this may be linked to the low cost cameras or to the YUV compression during frame acquisition. Therefore, a method that can handle imprecise colour segmentation had to be devised. This was achieved using the assumption that, if not all, at least most of the

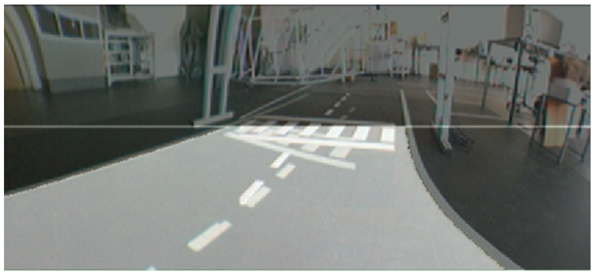


Figure 5 - The original merged image. The segmented road area is overlaid in white.

segmented green pixels belong to the object. Bearing this in mind, a simple centroid of the segmented pixels provides a fair enough indication of where the obstacle is located; it is now possible to come to a decision regarding the positioning of the obstacle on the road. The rules of the competition establish that the object must always be placed on one side or the other of the road, never in the middle. In consequence, the robot needs only to decide whether an object is present and on which side of the road it is. The first part is achieved by a fixed threshold for the number of segmented pixels. The second part of the problem is not so straightforward because one must first compute the position of the robot on the road.

As discussed before, both road and lane detection is performed, producing information regarding the centre of the road. Using this information, one can use a simple criterion that finds whether the object's mass centre is to the right or left of the line that passes in the robot's centre (middle bottom of image) and the points indicating the middle of the road (Figure 6).

D. Zebra Crossing Detection

The ATLAS project has developed a couple of algorithms for zebra crossing detection. The general idea is to look for a pattern similar to the zebra crossing area or some relevant part of it. Since the search may become computationally demanding, it is necessary to define a delimited region (which is the region enclosed by the lane markers) where the zebra area is to be looked for. With any of the two algorithms mentioned previously, it is easy to distinguish the pixels that are inside the road. The zebra crossing area needs to be detected by the vision system by means of white blobs of pixels. Earlier results on distorted images from the cameras [13] did not allow a rigorous template matching scheme since the zebra crossing area varies greatly with the point of view; for that approach a simple area/perimeter (form factor) limit was used. With the undistorted perspective image introduced in the ATLAS MV series robots, the real geometry of the zebra area can now be sought. The technique relies on pattern matching performed on low resolution images. A template picture of the zebra area is previously taken off-line and then correlated against the current image.

Template matching is known to be an effective pattern detection method, but has two limitations: it is sensitive to

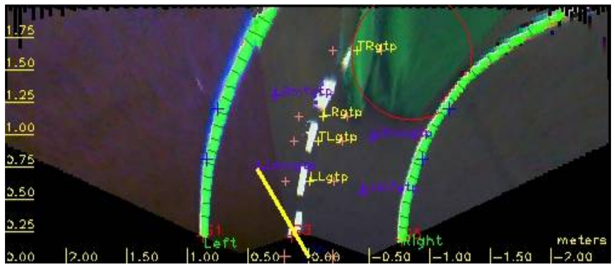


Figure 6 – Obstacle detection based on the mass centre calculation of rough colour segmentation. The mass centre is represented by the centre of the circle superimposed onto the image.

pattern scaling and rotation. In the undistorted perspective image, the zebra area size is constant. Nonetheless, it does appear rotated in the image when the robot is not properly aligned with the road. In this case, the template matching would clearly fail. In order to solve this problem, the algorithm makes, once again, use of the line detection module information. The low resolution image is rotated around the robot position, i.e., the bottom centre of the image. The angle of rotation is defined by the mean normal vector angle of one of the lane markers. Hence, the usage of the mean angle does not introduce a significant error. Nevertheless, we make use of the line whose normal vectors variance is smaller, which is equivalent to take the straightest line as reference. In summary, pattern matching is also effective when the zebra crossing area appears rotated. Figure 7 shows the detection trough template matching with previous rotation compensation.

E. Road Maintenance Area

One of the most challenging tasks of the ROBOTICA competition is to deal with the road maintenance area. This area is defined by a set of road maintenance cones, coloured orange and white. Because the cones have a white stripe, it is not possible to segment them completely in a simple step. As a consequence, the road maintenance navigation algorithm must be able to cope with partial cone detection. Two algorithms have been developed: one that uses the original perspective deformed image, and another that takes advantage of the geometry corrected top view image of the road. The problem is that orange segmentation highlights regions of the pins that are close to the road, i.e. the base of the pins, but also the top of the cones which appear separated since they are at higher heights. Because of this, a simple search from left to right, for segmented orange, will not keep a geometrically accurate representation of the road when regions of the top of the cones are found. Hence, a method that discards the top of the cones must be implemented. For this, we have used an *a priori* knowledge stating that the robot is always in between the route defined by the cones. Taking into account that the robot is always inside the area defined by the cones, it calculates a polar transformation of the orange segmented pixels, anchored on the assumed robot position (the middle bottom of the image). The polar representation of the colour mask is shown on Figure 9 (*top*).

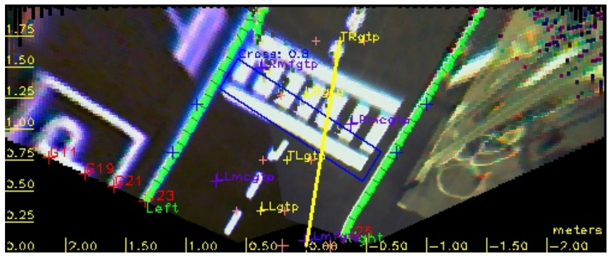


Figure 7 – Zebra cross area detection using template matching.

A convex hull of the region defined by the segmented pixels in polar space is able to detach the top of the cones. In fact, the lower poly line of the convex hull always stops at the base of the cones. This phenomenon is due to the fact that the projection of the cones onto the image places the cones' top regions further away from the middle bottom part of the image (Figure 9, bottom). Despite the fact that it is a top view of the road, the statement holds also for images without perspective transformation, such as Figure 8.

In Cartesian space, the lower part of the poly line portrays a region unaffected by the top of the cones. This region, despite the fact that it is not optimal, is sufficient for low speed navigation. This region is depicted as the blue area marked on Figure 8.

A second algorithm was developed to increase navigation performance in the road maintenance area, which takes advantage of the perspective corrected view of the road (Figure 9, *bottom*). The first objective was to classify the segmented orange pixels as belonging to right or left side cones. Because of the competition's road size, images showing tight turns occur very often. This poses a problem since the left side of the road (or the left side delimiting cones) often appears on the right side of the image and vice versa. It is imperative to distinguish left from right side cones.

To achieve this, a simple dilation-based algorithm is performed. Dilation of the colour segmentation mask is performed a fixed number of times. The outcome produces a new image with two separate blobs. These blobs result from the merging, through dilation, of the colour segmented portions of the cones, and provide a good indication of which regions of the colour mask belong to which side. Up until now, the number of dilations is fixed. The results are fairly good since the cones of one side are always closer to each other than to the cones on the other side. Considering that the cones of each side of the road have the major separation in the horizontal direction of the image, an appropriate structuring element for dilation can ease the merging of regions through a vertical connection and encumbers the horizontal connection. For this purpose, a rectangular, vertically elongated, structuring element is used for dilation. The results produce blobs that are better separated, helping to avoid the connection of both sides. The final operation is to perform a AND operation between the original colour mask and each of the blobs, obtaining a mask

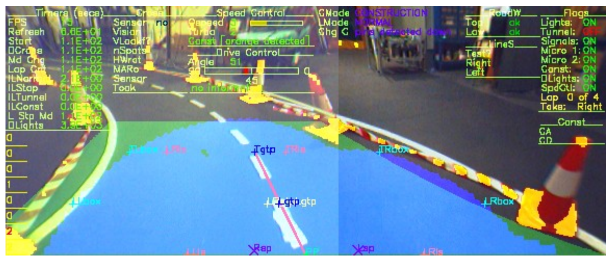


Figure 8 – On the top, a robot's view from inside the road maintenance area. Orange cones segmentation is superimposed as yellow. The blue region is obtained after the log-polar space convex hull operation.

for the colour segmented pixels of each side of the road. Afterwards, a convex hull operation is executed on the segmented pixels of each side. Then, the points of the convex hull first found by a radial search starting from the robot position are stored to define a poly line that portraits the delimiters defined by the bases of the cones. Convex hull connects the bases of the cones (only a portion of it, of course, but then the important line is extracted by the radial search) and because of that the top of the cones is discarded, enabling an accurate reconstruction of the area delimited by the cones. The colour segmentation does not have to be very accurate. In fact, it should be strictly tuned since that noise (in the sense of segmented orange colour without being so) on the inside of the road may disrupt the poly line defined by the bases of the cones. Figure 10 (*top*), shows the convex hull obtained for the left side mask of the orange segmented pixels, and a schematic of the radial search.

Figure 10 (*bottom*) shows the final output of the algorithm. The radial search highlighted the portion of the convex hull that connects the bases of the cones.

V. Conclusions and Future Work

This paper focuses the perception approaches used in the ATLAS Robots, which are based entirely on vision techniques. The ATLAS robots have shown in the last years, during national robotic competitions, that it is possible to perform efficient road perception and navigation using low cost cameras and systems. The entire set of algorithms described to extract the road and other features are processed at over 15 Hz in common 1.6 GHz Centrino laptops. These techniques have been developed for the competition but can be transposed to real road conditions, which is the authors' intention for the near future. The latest version of the ATLAS robot makes use of a multi camera active perception unit. The apparently more demanding setup is overcome by real time dynamic perspective correction and shows up several advantages, namely the active perception capabilities which will be crucial for tracking moving targets, and also the real geometric perception of ground based elements of the road, allowing for better pattern matching and size dependent feature extraction. These capabilities altogether installed on the latest ATLAS robot offer high potential for intelligent perception and navigation on roads, not only

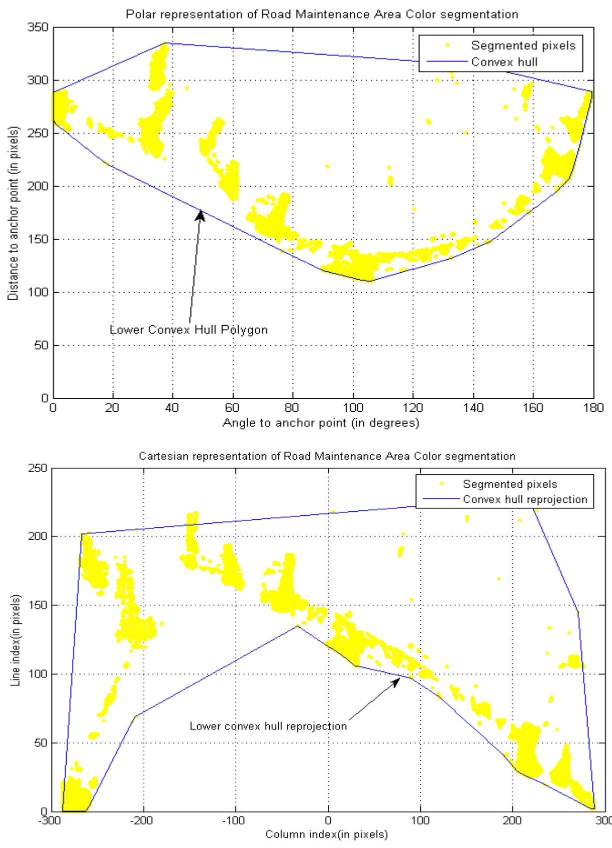
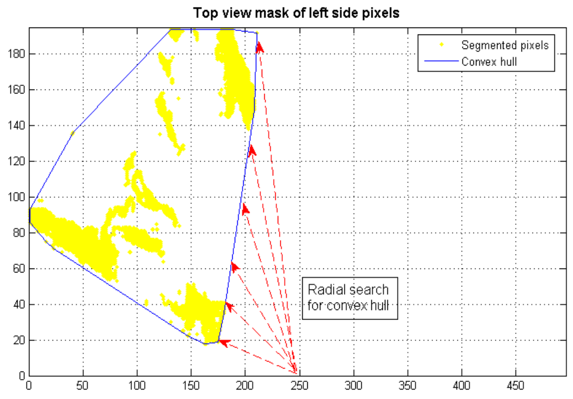


Figure 9 – Polar representation of the maintenance area's colour mask and the convex hull of the segmented pixels (*top*). Cartesian representation of the maintenance area's colour mask and the transformation of the convex hull calculated on the Polar space (*bottom*).

focusing on road or lane detection, but aiming also at active tracking of moving objects while still keeping a calibrated view of the road that is being traversed.



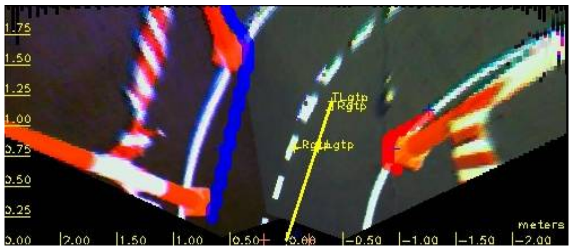


Figure 10 – Convex hull of the blobs obtained trough dilation of the orange colour's segmentation mask (*top*). Final output of the algorithm, where the bases of the cones are marked (*bottom*).

REFERENCES

[1] Almeida, L., Azevedo, J. , Cardeira, C., Costa, P., Fonseca, P. Lima, P., Ribeiro, F., Santos, V., 2000. Mobile Robot Competitions: Fostering Advances in Research, Development and Education in Robotics. *Proc. of the 4th Portuguese Conference on Automatic Control, CONTROLO2000*, 4-6 October 2000, Guimarães, Portugal, pp. 592-597

[2] Bertozzi M., Broggi A., Fascioli A., 2000. Vision-based intelligent vehicles: State of the art and perspectives, *Robotics and Autonomous Systems 32 (2000) 1-16.*

[3] Pomerleau D. 1995, RALPH: Rapidly Adapting Lateral Position Handler, Proceedings of the Intelligent Vehicles '95 Symposium, p. 506-511, 25-26 Sep 1995.

[4] Fletcher L., Petersson L., Zelinsky A. 2003, Driver Assistance Systems based on Vision In and Out of Vehicles, *Proceedings of the Intelligent Vehicles Symposium, p. 322-327, 9-11 June 2003.*

[5] Liu T., Zheng N., Cheng H., Xing Z., 2003, A Novel Approach of Road Recognition Based on Deformable Template and Genetic Algorithm, Proceedings of the IEEE *Intelligent Transportation Systems, p. 1251-1256, vol. 2, 12-13 Oct 2003, ISBN:0-7803-8125-4.*

[6] Zhang H., Yuan K., Mei S., Zhou Q., 2004, Visual Navigation of an Automated Guided Vehicle Based on Path Recognition, *Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004.*

[7] Foedisch M., Schlenoff C., Shneier M., 2005, Towards an Approach for Knowledge-based Road Detection, Proceedings of the 2005 CIKM Conference: Workshop on Research in Knowledge Representation for Autonomous Systems, October 31 – November 5, Bremen, Germany.

[8] Antolovic D., Leykin A., Johnson S., 2005, Vanishing Point: a Visual Road-Detection Program for a DARPA Grand Challenge Vehicle, *Indiana University.*

[9] Bradski G., Kaehler A., Pisarevsky V., 2005 - Learning-Based Computer Vision with Intel's Open Source Computer Vision Library, *Intel Technology Journal, Volume 9, Issue 2, 2005.*

[10] Lombardi P., Zanin M., Messelodi S., 2005, Switching Models for Vision-based On-Board Road Detection, Proceedings of the IEEE Intelligent transportation Systems, p. 67-72, 13-15 September, ISBN: 0-7803-9215-9.

[11] Apostoloff N., Zelinsky A., 2003, Robust Vision based Lane Tracking using Multiple Cues and Particle Filtering, *Proceedings of the IEEE Intelligent Vehicles Symposium, p. 558-563, 9-11 June 2003, ISBN: 0-7803-7848-2.*

[12] Oliveira M., Santos V., Multi-Camera Active Perception System with Variable Image Perspective for Mobile Robot Navigation, *8th Conference on Mobile Robots and Competitions, Festival Nacional de Robótica 2008, Aveiro, April 2008.*

[13] M. Oliveira, V. Santos , A Vision-based Solution for the Navigation of a Mobile Robot in a Road-like Environment, *Robótica, nº 69, 2007 p. 8 (ISSN: 0874-9019).*

[14] R. Cancela, M. Neta, M. Oliveira, V. Santos, 2005. ATLAS III: Um Robô com Visão Orientado para Provas em Condução Autónoma, *Robótica, nº 62, pp. 4-11, (ISSN: 0874-9019).*

[15] Denavit, J., Hartenberg, R. S., 1955 - A Kinematics Notation for Lower-Pair Mechanisms Based on Matrices. *J. App. Mech., v.77, p.215-221.*

[16] Oliveira M., Santos V., 2008, Real Time Road Line Extraction with Simple Statistical Descriptors, *IEEE International Conference on Intelligent Robots Systems (IROS 2008) 22-26 September 2008, Nice,* France.

Robotic Color Image Segmentation by Means of Finite Mixture Models

Nicola Greggio $^{\oslash,\dagger}$- *IEEE Member*, Alexandre Bernardino † - *IEEE Member*
José Santos-Victor † - *IEEE Member*

$^{\oslash}$ *ARTS Lab - Scuola Superiore S.Anna, Polo S.Anna Valdera*
Viale R. Piaggio, 34 - 56025 Pontedera, Italy

† *Instituto de Sistemas e Robótica, Instituto Superior Técnico*
1049-001 Lisboa, Portugal

$^{\oslash}$*nicola.greggio@sssup.it* - †*ngreggio@isr.ist.utl.pt*

***Abstract*— Image segmentation for robots requires to be fast, in order to deal with ever more powerful processors. Moreover, it is assumed to be robust to environmental changes, such as light conditions. In this paper we propose the application of a couple of unsupervised learning algorithms for the estimation of the number of components and the parameters of a mixture model for image segmentation. These serve for the unsupervised identification of multiple different objects in a visual scene, such as for a subsequent localization and tracking. We compare our previous technique against the new one. The distinctive aspect of our new approach is related to a top-down hierarchical search for the number of components by means of a binary tree decision structure. This work analyzes both approaches, two previous work of ours, in terms of applicability to object detection for robotic applications. Besides, we propose the computational burden evaluation for the two algorithms.**

Index Terms - *Robotics, Computer Vision, Object Segmentation, Unsupervised Learning, Self-Adapting Expectation Maximization*

I. Introduction

Nowadays, computer vision and image processing are involved in many practical applications. The constant progress in hardware technologies leads to new computing capabilities, and therefore to the possibilities of exploiting new techniques. Image segmentation is a key low level perceptual capability in many robotics related application, as a support function for the detection and representation of objects and regions with similar photometric properties. Several applications in humanoid robots [1], rescue robots [2], or soccer robots [3] rely on some sort on image segmentation [4]. Additionally, many other fields of image analysis depend on the performance and limitations of existing image segmentation algorithms: video surveillance, medical imaging and database retrieval are some examples [5], [6].

A. Related Work

Two main principal approaches for image segmentation are adopted: Supervised and unsupervised. The latter one is the one of most practical interest. It may be defined as the task of segmenting an image in different regions based on some similarity criterion among each region's pixels.

Several techniques have been proposed in the literature for unsupervised learning, from Kohonen maps [7], Growing Neural gas [8], [9], k-means [10], to Independent component analysis [11], [12], etc. Particularly successful is the Expectation Maximization algorithm applied to finite mixture models. Fitting a mixture model to the distribution of the data is equivalent, in some applications, to the identification of the clusters with the mixture components [13].

One of the most widely used distributions is the normal, or Gaussian, distribution. The normal distribution can be used to describe, at least approximately, any variable that tends to cluster around the mean. If data is generated by a mixture of Gaussians, the clustering problem will reduce to the estimation of the number of Gaussian components and their parameters. Expectation-Maximization (EM) algorithm is well known and attractive approach for learning the parameters of mixture models [13], [14]. It always converges to a local optimum [15], especially for the case of Normal mixtures [13], [16]. However, it also presents some drawbacks. For instance, if requires the *a-priori* specification of the model order, namely, the number of components and its results are sensitive to initialization. The selection of the right number of components is a critical issue. The more components there are within the mixture, the better the data fit will be. Unfortunately, increasing the number of components will lead to data overfitting and to increase in the computational burden. Therefore, finding the best compromise between precision, generalization and speed is an essential concern. A common approach to address this compromise is to try different hypothesis for the number of components, and then selecting the best model according to some appropriate model selection criteria.

Different techniques can be used to select the best number of components in a mixture distribution. These can be divided into two main classes: *off-line* and *on-line* techniques.

The first ones evaluate the best model by executing independent runs of the EM algorithm for many different initializations, and evaluating each estimate with criteria that penalize complex models (e.g. the Akaike Information Criterion (AIC) [17], the Schwarz's Bayesian Information Criterion [18], the Rissanen Minimum Description Length (MDL) [19], and Wallace and Freeman Minimum Message Length (MML) [20]). All of these criteria, in order to be effective, have to be evaluated for every possible number of models under comparison. Therefore, it is obvious that, for having a sufficient search range the complexity goes with the number of tested models as well as the model parameters.

The second ones start with a fixed set of models and sequentially adjust their configuration (including the number of components) based on different evaluation criteria. Pernkopf and Bouchaffra proposed a Genetic-Based EM Algorithm capable of learning gaussians mixture models [21]. They first selected the number of components by means of the minimum description length (MDL) criterion. A combination of genetic algorithms with the EM has also been explored. Simulating annealing has also been explored as a possible solution for mixture selection, with Ueda and Nakano who proposed the deterministic annealing (DAEM), in which a modified posterior probability parametrized by *temperature* is derived to avoid local maxima [22]. Ueda *et Al.* proposed a split-and-merge EM algorithm (SMEM) to alleviate the problem of local convergence of the EM method [23].

Greedy algorithms take part within the second class of unsupervised classification techniques. They are characterized by making the locally optimal choice at each stage with the hope of finding the global optimum. Applied to the EM algorithm, they usually start with a single component (therefore side-stepping the EM initialization problem), and then increase their number during the computation. At the time, no precise solution has been posted to address this drawback. In 2002 Vlassis and Likas introduced a greedy algorithm for learning Gaussian mixtures [24]. They start with a single component covering all the data. Then they split an element and perform the EM locally for optimizing only the two modified components. Nevertheless, the total complexity for the global search of the element to be split $O(n^2)$. Subsequently, Verbeek et al. developed a greedy method to learn the gaussians mixture model configuration [25]. Their search for the new components is claimed to take $O(n)$, while our recursive search by means of the binary tree only $O(\log n)$.

B. *Our contribution*

We want to find a procedure for the unsupervised identification of multiple different objects in a visual scene, for a further localization and tracking. Therefore, we first need to segment the color image in a unsupervised way in order to detect the different targets and distinguishing them from the background. Then, we need to identify them. We decided to use Gaussian mixture models due to their accuracy and general applicability. Besides, in order to sidestep the shortcoming of high computational burden together with the need of the $a-priori$ decision of the number of components, we opted for a greedy self-organizing approach.

However, greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. Our new technique tries to overcome this limitation by using a binary tree for deciding which component has to be replicated in an exhaustive way. The optimization of the parameters is done with expectation maximization (EM) and the search for the best number of parameters is done in a top-down manner, by starting with a single component and progressively adapting the mixture by adding new elements according to a binary tree structure. We compare a previous greedy algorithm we developed in [26], and then refined in [27], versus our new technique, presented in [28]. The restriction of the previous approach relies on the excessive number of input parameters to be tuned before the computation, and the heuristic stopping criterion. The latter may cause that more components than those effective necessary may be employed, resulting in an excessive mixture complexity and long computation.

C. *Outline*

In sec. II we introduce the analyzed algorithms. Besides, in sec. II-F and II-G we propose a computational complexity analysis. Then, in sec. III we describe our experimental set-up for testing the validity of our new technique and we compare our results against our previous alternative. Finally, in sec. V we conclude.

II. The finite mixture learning algorithms

In this section we provide a description about the differences between the two approaches. Since now, we will refer to these as:

- FASTGMM: The previous approach [27];
- FSAEM: The new algorithm [28].

In the following we present a brief overall description of both approaches, and a deeper analysis of the adding a new component process, together with the stopping criterion.

A. *FASTGMM Overall Description*

The basic idea is to incrementally estimate the mixture parameters and the number of components simultaneously. This algorithm starts with a single component and only increments its number as the optimization procedure progresses. The number of components is incremented at certain stages of the optimization procedure but the values of the mixture parameters are incrementally changed and not reinitialized.

The key issue of our technique is looking whether one or more Gaussians are not increasing their own likelihood during optimization.

For this algorithm we need to introduce a state variable related to the state of the gaussian component:

- Its age, that measures how long the component's own likelihood does not increase significantly;

Then, the split process is controlled by the following adaptive decision thresholds:

- One adaptive threshold Λ_{TH} for determining a significant increase in likelihood;
- One adaptive threshold A_{TH} for triggering the split process based on the component's own age;
- One adaptive threshold ξ_{TH} for deciding to split a gaussian based on its area.

It is worth noticing that even though we consider three thresholds to tune, all of them are adaptive, and only require a coarse initialization.

However, one of the biggest limitations is that this is suitable only for Gaussian mixtures, and not for generic ones.

B. *FSAEM Overall Description*

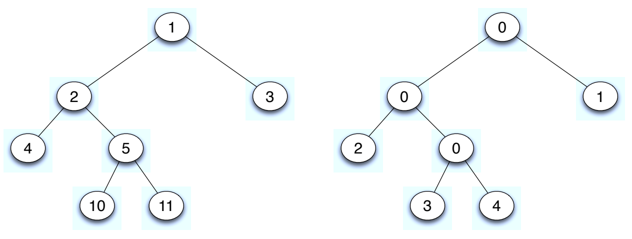
This algorithm starts with a single component. Then, by following a binary tree structure new classes are added by means of replicating existing ones, once a time. Subsequently, our modified EM algorithm is run in order to achieve the current mixture best configuration [28]. Furthermore, the cost function is evaluated in order to decide whether keeping or discarding the current mixture. In the first case, the binary tree will be updated with the new solution. When a new mixture element is added, it will become a child together with the original one. Therefore, within our representation, its father dies, and only the two children survive. Otherwise, in the second case (i.e. when the new mixture configuration is discarded), the previous mixture will be restored as a starting point for a new component replication, and that node will never be proposed to have children anymore. Finally, when there will no node eligible to have children (i.e. when all the combinations have been tried), the algorithm terminates.

It is worth noticing that this technique can be applied to any mixture, rather than Gaussian ones, merely.

C. *Adding a new mixture class*

One the one hand, FASTGMM splits the component with highest covariance matrix determinant (and therefore that covering the highest number of data), when this overcome a threshold. The latter one follows a decreasing law, in order to avoid stationary situations. However, both the splitting process itself is ill-posed (there are infinite solutions [23]), and the chosen decision criterion is arbitrary. Finally, the law leading the thresholds variation is heuristic.

On the other hand, FSAEM employs a replication process rather than a splitting procedure. The new component will be the exact copy of that candidate to be replicated. This allows to a unique solution, with the only exception of a variation in the mean of these components (in order to not have them exactly superimposed, situation not allowing the EM procedure to escape the current local minimum [28]). Besides, instead of relying on empirical thresholds for determining the most suitable component to be replicated, all the classes are replicated in sequence, in order to exploit all the mixture combinations possibilities. This is achieved thanks to the binary tree decision structure, showed in Fig. 1.



(a) Binary tree indexes (b) Binary tree contents

Fig. 1. Binary tree mixture distribution structure example: On the left the binary tree indexes representation, used as decision structure; on the right the contents of the binary tree used as look up table. The array correspondent representation is: $[0, 0, 1, 2, 0, 3, 4]$. The 0 contents are relative to the parents have been eliminated after creating their children.

D. *Model Selection Criterion: Minimum message length (MML)*

Since no merging or annihilating operation ha been envisioned, it is worth being sure about a new component insertion. FSAEM integrates a derivation of the MML criterion in order to evaluate whether the new mixture configuration (i.e. that after the last component replication) provides a better data description in terms of the MML evaluation. In the affirmative case, the current mixture configuration is kept, a new replication is performed following the updated binary tree. Otherwise, the old mixture configuration is resumed (therefore voiding the last component insertion). The binary tree is updated in order to not replicate the same component anymore once the correspondent mixture has been discarded.

We adopted the minimum message length (MML) criterion developed in [29], which formulation is:

$$\bar{\vartheta}_{opt} = arg\min_{\bar{\vartheta}} \Big\{ -L\left(X|\bar{\vartheta}\right) + \frac{N}{2}\sum_{i=1}^{c} \ln\left(\frac{n \cdot w_i}{12}\right) + \frac{c}{2}\left(N + 1 - \ln 12n\right)\Big\} \tag{1}$$

In eq. (1):

- $-L\left(X|\bar{\vartheta}\right)$ is the log-likelihood of the whole distribution;
- N is the number of parameters specifying each component (e.g. in case of a normal distribution they are the mean and the covariance matrix, counted as 1 parameter for each input dimension for the mean and 1 for each the covariance among each dimension, resulting in $N = d + (d+1) * d/2$);
- c is the number of mixture components;
- n is the number of input data;
- w_i is the *a-priori* class, or component, probability, therefore resulting $n \cdot w_i$ the number of components of the class i.

E. *Stopping criterion*

On the one side, FASTGMM relies on an empirical stopping criterion. This, together with the splitting procedure may result in having the best mixture data description.

On the other side, FSAEM stops when all the mixture replication combinations have been exploited. Besides, the MML criterion described in sec. II-D ensure that when the insertion of a new component do not improve the data description, this is discarded.

F. *FASTGMM Computational complexity evaluation*

Taking D as the input dimension, the computational burden of each iteration is:

- the original EM algorithm takes $O\left(N \cdot D \cdot nc\right)$ for each step, for a total of $O\left(4 \cdot N \cdot D \cdot nc\right)$ operations;
- the algorithm takes $O\left(nc\right)$ for evaluating all the single Gaussians log-likelihood;
- the split operation requires $O\left(D\right)$.
- the others take $O\left(1\right)$.
- the optional procedure of optimizing the selected mixture takes $O\left(4 \cdot N \cdot D \cdot nc\right)$, being the original EM.

Therefore, the original EM algorithm takes:

- $O\left(4 \cdot N \cdot D \cdot nc\right)$, while our algorithm adds $O\left(D \cdot nc\right)$ on the whole, or $O\left(4 \cdot N \cdot D \cdot nc\right)$, giving rise to $O\left(4 \cdot N \cdot D \cdot nc\right) + O\left(D \cdot nc\right) = O\left(4 \cdot N \cdot D \cdot nc + D \cdot nc\right) = \left(nc \cdot D \cdot \left(4N+1\right)\right)$ in the first case;
- $2 \cdot O\left(4 \cdot N \cdot D \cdot nc\right) + O\left(D \cdot nc\right) = O\left(8 \cdot N \cdot D \cdot nc + D \cdot nc\right) = \left(nc \cdot D \cdot \left(8N+1\right)\right)$ in the second case, with the optimization procedure.

G. *FSAEM Computational complexity evaluation*

The computational burden of each iteration of the algorithm is:

1. The original EM algorithm takes $O(k \cdot D \cdot nc)$ for the whole mixture log-likelihood evaluation, $O(k \cdot D \cdot nc)$ for the E-step, and approximatively the same amount for the M-step, and $O(nc)$ for the prior re-estimation, therefore resulting in $O(k \cdot D \cdot (nc+1))$ for each step;
2. The binary tree, if complete, takes $O(\log nc)$ for the insertion, the selection, and the removal operation.
3. Our algorithm takes $O(nc)$ for evaluating all the components possible ill-conditioning (this would result in evaluating the covariance matrix determinants in case of Gaussian components, which requires $O(D!)$ additionally, and another $O(D)$ for replicating along all the input dimensions whether necessary);
4. Our replication operation requires $O(D!)$ for the SVD decomposition, plus other $O(nc)$ for the components reallocation.

This gives rise to the total computation: $O(k \cdot D \cdot (nc+1)) + O(\log nc) + O(nc) + O(D!) + O(D) + O(D!) + O(nc) + O(1) = O(k \cdot (D+1) \cdot (nc+3) + \log nc) + 2O(D!))$.

Considering that usually $D << k$ and $nc << k$, we can assume that the computational complexity does not differ considerably between the general case and the application to the mixtures of Gaussians. Therefore, we assume that the total computational burden goes with $O(k \cdot D \cdot (nc+1) + \log nc)$.

III. Experiments

Due to its robotic application, we tested our algorithm on camera images taken from our robotic platform, the iCub. The iCub cameras are two DragonFly with VGA resolution and 30 fps speed. The acquired images have a resolution of $320x240$. We will segment these images by means both algorithms, in order to compare them both in terms of accuracy and, processing speed.

We segmented the images as 5-dimensional input in the (R,G,B) space and (x,y), i.e. a generic input point was of kind: $p \in (R, G, B, x, y)$. Then, we applied a simple Gaussian blur and the connected component labeling.

The color image segmentation results are shown in Fig. III. We highlight the blob findings, centering them on their mean and surrounded by their covariance matrix (represented as an ellipse in 2D, green for the binary images and red for the color ones). Input $(1),(2),(3)$ have been chosen to have a considerably lower light contrast than the last two ones, $(4),(5)$. For each row, the first three images on the left, for each column, are obtained with the FASTGMM algorithm, while the other three on the right with the FSAEM approach. Here, the original image, the mixture learning segmented image, and the connected component resultant image are shown, respectively.

We made the same experiments with the same input images both with the RGB color segmentation and the HSV one. Then, we also show the results for the same images in the (H, S, V, x, y) representation in Fig. IV.

Finally, we also measure the elapsed time of both algorithms. This has been performed by means of the *time profile* matlab function. However, although this is claimed to be not sensitive to the other running applications (it counts the number of float operations), we experimented that this is not true, indeed. Therefore, this test cannot be considered as precise ad exhaustive, but indicative, only.

IV. Discussion

The accuracy of the image segmentation greatly depends on the number of employed mixture components. The higher it is, the more accurate the image reconstruction will be. However, using too many components may lead to an overfitting of the input set, together with an excessive increase of the computational burden. Therefore, finding the $best$ compromise is a must. With FSAEM the $best$ compromise is decided by the MML criterion of eq. (1).

In tab. I the results of FASTGMM and FSAEM applied to the selected images with the RGB color segmentation are shown, while in tab. II there are the results for the HSV color space. In each table we report:

- The number of detected components;
- The actual number of components, i.e. that of the generation mixture;
- The number of total iterations;
- The elapsed time (this is relative to the image segmentation only, and not to the connected component labeling);
- The percentage difference in time for the new algorithm $(Time_{FSAEM})$ with respect to the old formulation $(Time_{FSAEM})$, evaluated as $\frac{Time_{FSAEM} - Time_{FSAEM}}{Time_{FSAEM}} \cdot 100$;
- The final log-likelihood;
- The percentage difference in final log-likelihood for the new algorithm versus the previous approach.

A. RGB *versus* HSV *color space*

Comparing the results shown in in Fig. 2 and in Fig. 3 it can be seen that generally both algorithm perform better under

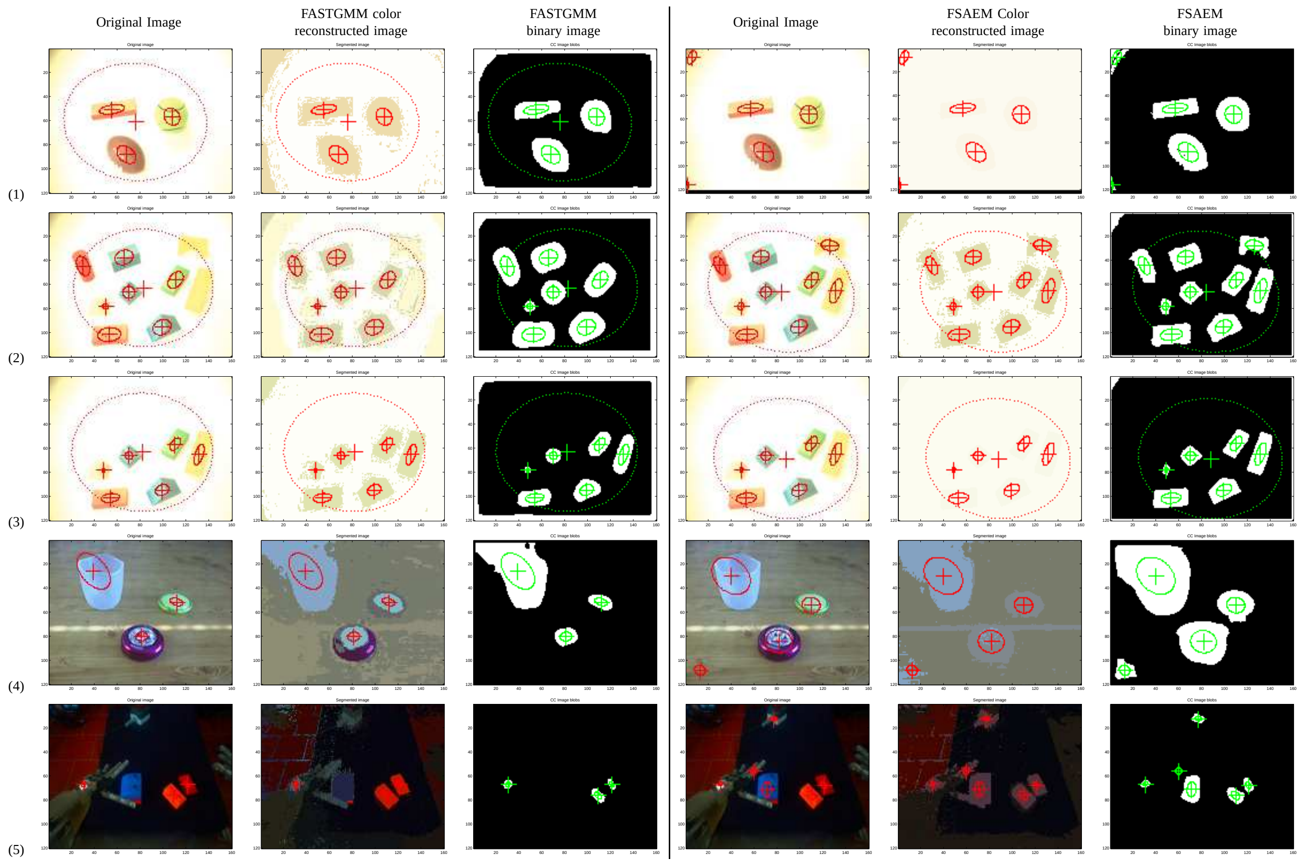


Fig. 2. Image segmentation in the RGB color space. FASTGMM results are shown on the left, while FSAEM outcomes are represented on the right. For each image, there is a subset composed by the original image, the color image reconstruction, and the binary image labeled with the connected components, respectively. Each image contains the objects of interest highlighted in red for the color output, and green for the binary one superimposed. The objects have been marked with their mean and covariance, represented as a regular ellipse in 2D, obtained with the connected components labeling.

RGB Color Segmentation Results							
Image number	Algorithm	Detected number of Gaussian components	Number of iterations	Elapsed Time [s]	Percentage time difference	Final log-likelihood	Percentage difference on log-likelihood
(1)	FASTGMM	2	38	1.244413	-37.05064155	-278959.4334	-55.54316696
	FSAEM	2	26	1.705476		-433902.3373	
(2)	FASTGMM	3	21	0.933944	-129.1852616	-313336.7632	-12.9245556
	FSAEM	2	38	2.140462		-353834.1474	
(3)	FASTGMM	2	62	2.244048	-101.9897524	-285480.1774	-54.44871057
	FSAEM	2	121	4.532747		-440920.453	
(4)	FASTGMM	11	332	46.743716	95.54896534	-403884.4389	-2.432889246
	FSAEM	3	31	2.080579		-413710.5	
(5)	FASTGMM	11	285	34.017004	82.4117315	-387677.3402	-4.231222296
	FSAEM	4	106	5.983002		-404080.8302	

TABLE I
EXPERIMENTAL RESULTS ON REAL ROBOTIC IMAGES. SEGMENTATION PERFORMED IN THE RGB COLOR SPACE.

the HSV color segmentation than the RGB one. It is well-known that the HSV representation is more robust to light changes, for instance. Images $1, 4$, and 5 are better segmented in terms of number of effective objects detected. Specifically, input 3 gives rise to better results for the FSAEM approach, while resulting less precise with the FASTGMM segmentation. However, image 2 is confused for both algorithms within the HSV color space. Besides, image 4 is better segmented by FASTGMM in both color spaces. Finally, it is wort noticing that segmenting within HSV color space requires less iterations.

Comparing the segmented images, it may seem at a first glance that FASTGMM performs better than FSAEM. This is because generally FSAEM uses less mixture components for representing the image (this is clear both in Fig. 2 and in Fig. 3, where the FSAEM reconstructed images - the middle column on the right set - are less precise). However, the final results in terms of object recognition may seem similar. Nevertheless, FSAEM is capable of extracting the important features as well as FASTGMM, while requiring less computational resources.

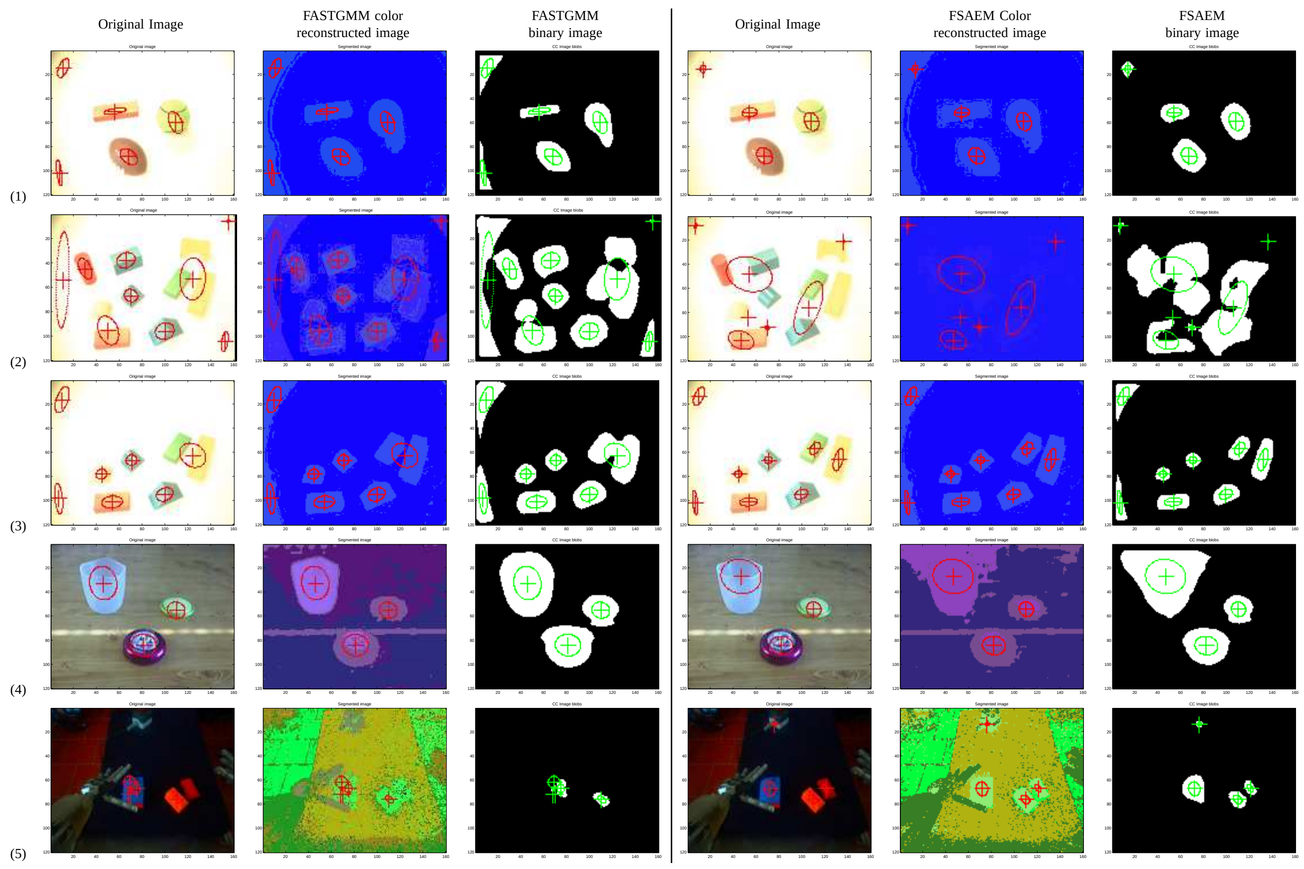


Fig. 3. Image segmentation in the HSV color space. FASTGMM results are shown on the left, while FSAEM outcomes are represented on the right. The input images are the same as in Fig. III, and so are the output subsets organization. As for the images in the previous figure, the recognized objects of interests have been highlighted and superimposed to the original pictures.

HSV Color Segmentation Results							
Image number	Algorithm	Detected number of Gaussian components	Number of iterations	Elapsed Time [s]	Percentage time difference	Final log-likelihood	Percentage difference on log-likelihood
(1)	FASTGMM	2	25	0.755909	-171.7325763	-338069.989677	-33.18992954
	FSAEM	2	48	2.054051		-450275.181	
(2)	FASTGMM	3	22	0.877972	-288.2384632	-340969.8623	-17.78399023
	FSAEM	2	76	3.408625		-401607.9093	
(3)	FASTGMM	2	25	0.754844	-117.6407311	-352858.824	-29.54999667
	FSAEM	2	33	1.642848		-457128.5947	
(4)	FASTGMM	7	80	6.484346	0.1370223	-455537.6481	-3.387910601
	FSAEM	4	117	6.475461		-470970.8564	
(5)	FASTGMM	10	274	48.567681	93.45860059	-435224.6992	-2.347807229
	FSAEM	3	56	3.177006		-445442.9361	

TABLE II

EXPERIMENTAL RESULTS ON REAL ROBOTIC IMAGES. SEGMENTATION PERFORMED IN THE HSV COLOR SPACE.

B. *Log-Likelihood*

Fig. 4 shows the final log-likelihood of both approaches. Here it is possible to see when the FASTGMM or FSAEM add a component, i.e. corresponding to the spikes that lower the curve. Then, when the log-likelihood does not increase anymore significantly the FASTGMM computation stops, while FSAEM stops when there are no more components to be replicated.

C. *Cost Function*

Finally, we provide the cost function evolution of the MML information criterion as function of the number of components for the FSAEM algorithm (FASTGMM do not provide an information criterion). Fig. 5 shows a couple of examples, namely those of the image no. 4 and, both present in Fig. 2 and in Fig. 3.

D. *Elapsed time*

Generally FSAEM performs faster than FASTGMM. The results in tab. I and in tab. II demonstrate that generally

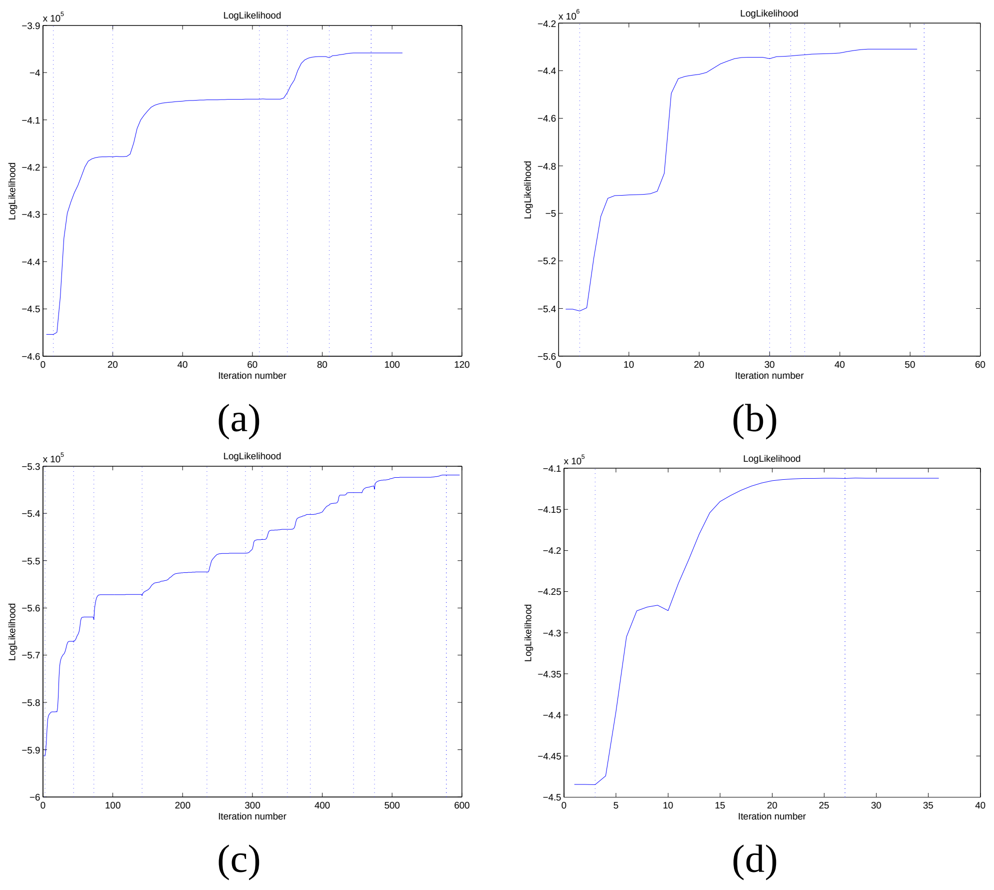


Fig. 4. The final log-likelihood evolution as function of the number of iterations of two different kinds of input data used within the experiments: The image (4) - (a) FASTGMM and (b) FSAEM, and the image (5) - (c) FASTGMM and (d) FSAEM.

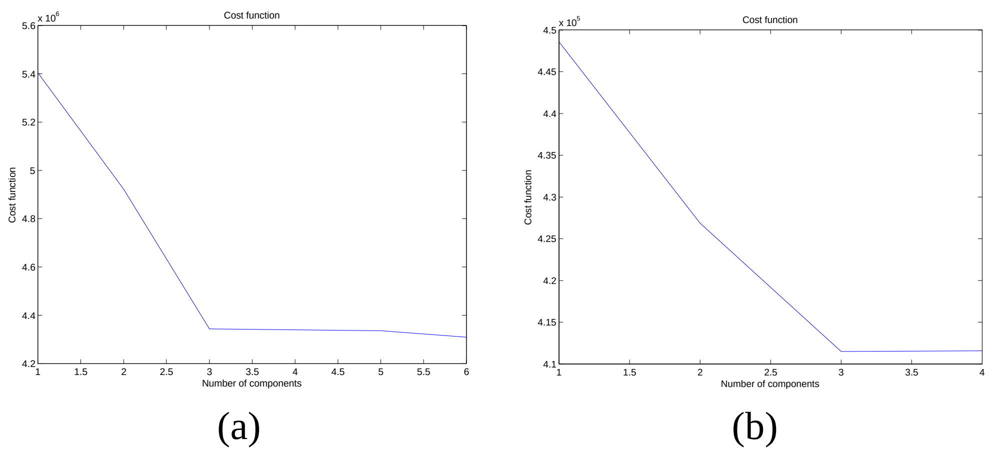


Fig. 5. The cost function evolution as function of the number of components of two different kinds of input data used within the experiments for the FSAEM algorithm: (a) The image (4), (b) and the image (5).

FSAEM runs faster. Indeed, for the first three images the elapsed time can be considered comparable, if not double for FSAEM with respect to FASTGMM. Nevertheless, the elapsed time is so low that it makes this comparison not accurate. As a confirmation the last two images, 4 and 5, that require a longer computation, advantage FSAEM.

Our explanation relies both on the splitting procedure and the stopping criterion, which are more heuristic in FASTGMM. The FSAEM replication process, that exploit all the mixture classes by means of the binary tree, may be the reason for the slower computation with the first three images. However, this also provides a better accuracy in the choice of the component to be replicated, and this in long term gives rise to a fewer EM iterations, and then to a lower elapsed time.

Moreover, FSAEM does not depend on the FASTGMM heuristic parameters and thresholds.

Finally, it can be argued that the approaches take always more than 750 ms to process each image. In a robotics framework, this time could be excessively high (it is near to 1-1.2 fps) and it could be a source of security problems. However, the algorithms have been implemented under Matlab herein, which is a interpreted language rather than C++, which is a compiled one. This means that the first one will result much slower than the second one. Generally, and this is for the iCub repository software, for this reason robotics applications are written in C/C++, and not in Matlab. We choose the latter for a sake of practicality.

V. Conclusion

In this paper we compared two unsupervised algorithms that on-line learns a finite mixture model from multivariate data, presented in a couple of previous work of ours. We briefly summarized the algorithms, with respect to their more salient characteristics. Besides, we also presented an accurate computational complexity analysis of both approaches. Finally, we discuss our results, arguing for a more general validity of the more recent approach.

Acknowledgements

This work was supported by the European Commission, Project IST-004370 RobotCub and FP7-231640 Handle, and by the Portuguese Government - Fundação para a Ciência e Tecnologia (ISR/IST pluriannual funding) through the PIDDAC program funds and through project BIO-LOOK, PTDC / EEA-ACR / 71032 / 2006.

References

[1] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory motor maps to imitation," *IEEE Trans. on Robotics*, vol. 24, no. 1, 2008.

[2] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Bridging the gap between simulation and reality in urban search and rescue"," in *Robocup 2006: Robot Soccer World Cup X*, 2006.

[3] N. Greggio, G. Silvestri, E. Menegatti, and E. Pagello, "Simulation of small humanoid robots for soccer domain." *Journal of The Franklin Institute - Engineering and Applied Mathematics*, vol. 346, no. 5, pp. 500–519, 2009.

[4] M. Vincze, "Robust tracking of ellipses at frame rate," *Pattern Recognition*, vol. 34, pp. 487–498, 2001.

[5] J. G. G. Dobbe, G. J. Streekstra, M. R. Hardeman, C. Ince, and C. A. Grimbergen, "Measurement of the distribution of red blood cell deformability using an automated rheoscope," *Cytometry (Clinical Cytometry)*, vol. 50, pp. 313–325, 2002.

[6] H. Shim, D. Kwon, I. Yun, and S. Lee, "Robust segmentation of cerebral arterial segments by a sequential monte carlo method: Particle filtering," *Computer Methods and Programs in Biomedicine*, vol. 84, no. 2-3, pp. 135–145, December 2006.

[7] T. Kohonen, "Analysis of a simple self-organizing process." *Biological Cybernetics*, vol. 44, no. 2, pp. 135–140, 1982.

[8] B. Fritzke, "A growing neural gas network learns topologies." *Adv ances in Neural Inform ation Processing Systems 7 (NIPS'94), MIT Press, Cambridge MA*, pp. 625–632, 1995.

[9] J. Holmström, "Growing neural gas - experiments with gng, gng with utility and supervised gng," 2002.

[10] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations." *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability.*, pp. 281–297, 1967.

[11] P. Comon, "Independent component analysis: a new concept?" *Signal Processing, Elsevier*, vol. 36, no. 3, pp. 287–314, 1994.

[12] A. Hyvärinen, J. Karhunen, and E. Oja, "Independent component analysis," *New York: John Wiley and Sons*, vol. ISBN 978-0-471-40540-5, 2001.

[13] G. McLachlan and D. Peel, "Finite mixture models." *John Wiley and Sons*, 2000.

[14] H. Hartley, "Maximum likelihood estimation from incomplete data." *Biometrics*, vol. 14, pp. 174–194, 1958.

[15] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood estimation from incomplete data via the em algorithm," *J. Royal Statistic Soc.*, vol. 30, no. B, pp. 1–38, 1977.

[16] L. Xu and J. M., "On convergence properties of the em algorithm for gaussian mixtures," *Neural Computation*, vol. 8, pp. 129–151, 1996.

[17] Y. Sakimoto, M. Iahiguro, and G. Kitagawa, "Akaike information criterion statistics," *KTK Scientific Publisher, Tokio*, 1986.

[18] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[19] J. Rissanen, "Stochastic complexity in statistical inquiry." *Wold Scientific Publishing Co. USA*, 1989.

[20] C. Wallace and P. Freeman, "Estimation and inference by compact coding," *J. Royal Statistic Soc. B*, vol. 49, no. 3, pp. 241–252, 1987.

[21] F. Pernkopf and D. Bouchaffra, "Genetic-based em algorithm for learning gaussian mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1344–1348, 2005.

[22] N. Ueda and R. Nakano, "Deterministic annealing em algorithm," *Neural Networks*, vol. 11, no. 2, pp. 271–282, 1998.

[23] N. Ueda, R. Nakano, Y. Ghahramani, and G. Hiton, "Smem algorithm for mixture models," *Neural Comput*, vol. 12, no. 10, pp. 2109–2128, 2000.

[24] N. Vlassis and A. Likas, "A greedy em algorithm for gaussian mixture learning," *Neural Processing Letters*, vol. 15, pp. 77–87, 2002.

[25] J. Verbeek, N. Vlassis, , and B. Krose, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.

[26] N. Greggio, A. Bernardino, and J. Santos-Victor, "Image segmentation for robots: Fast self-adapting expectation maximization." *International Conference on Image Analysis and Recognition (ICIAR), Povoa de Varzim, Portugal, June 21-23*, 2010.

[27] N. Greggio, A. Bernardino, C. Laschi, P. Dario, and J. Santos-Victor, "Fast estimation of gaussian mixture models for image segmentation," *Machine Vision and Application*, Accepted for publication 2011.

[28] N. Greggio, A. Bernardino, C. Laschi, J. Santos-Victor, and P. Dario, "Unsupervised greedy learning of finite mixture models." *IEEE 22th International Conference on Tools with Artificial Intelligence (ICTAI 2010), Arras, France*, October 27-29 2010.

[29] A. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 24, no. 3, 2002.

The IIIA30 Mobile Robot Object Recognition Dataset

Arnau Ramisa†, David Aldavert§, Shrihari Vasudevan‡, Ricardo Toledo§ and Ramon Lopez de Mantaras†

† Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, Bellaterra, E-08193, Spain.
Email: aramisa@iiia.csic.es and mantaras@iiia.csic.es
§ Computer Vision Center, Campus UAB, Bellaterra, E-08193, Spain.
Email: aldavert@cvc.uab.cat and ricard@cvc.uab.cat
‡ Australian Center for Field Robotics, the University of Sydney, NSW 2006, Australia
Email: shrihari.vasudevan@ieee.org

Abstract—**Object perception is a key feature in order to make mobile robots able to perform high-level tasks. However, research aimed at addressing the constraints and limitations encountered in a mobile robotics scenario, like low image resolution, motion blur or tight computational constraints, is still very scarce. In order to facilitate future research in this direction, in this work we present an object detection and recognition dataset acquired using a mobile robotic platform. As a baseline for the dataset, we evaluated the cascade of weak classifiers object detection method from Viola and Jones.**

I. Introduction

Currently there is a big push towards semantics and higher level cognitive capabilities in robotics research. One central requirement towards these capabilities is to be able to identify higher level features like objects, doors, etc. For example, in [1], the authors investigate underlying representations of spatial cognition for autonomous robots. Although not specifically addressed in that work, object perception is an essential component that the authors reported to be the most limiting factor.

Although different modalities of perception (e.g. laser range-finder, color camera, haptics) can be used, in this work we focus on passive vision, as it is interesting for several reasons like an affordable cost, passive and, thus, low power consuming, compatibility with human environments, richness of perceived information or usable both indoors and outdoors.

Recently several methods have been quite successful in particular instances of the problem, such as detecting frontal faces or cars, or in datasets that concentrate on a particular issue (e.g. classification in the Caltech-101 [2] dataset). However in more challenging datasets like the detection competition of the Pascal VOC [3] the methods presented achieved a lower accuracy. This low performance is not surprising, since object recognition in real scenes is one of the most challenging problems in computer vision [4]. The visual appearance of objects can change enormously due to different viewpoints, occlusions, illumination variations or sensor noise. Furthermore, objects are not presented alone to the vision system, but they are immersed in an environment with other elements, which clutter the scene and make recognition more complicated. In a mobile robotics scenario a new challenge is added to the list: computational complexity. In a dynamic world, information about the objects in the scene can become obsolete even before it is ready to be used if the recognition algorithm is not fast enough. Despite the importance of the problem, very few initiatives, have started addressing it; exceptional examples are the Semantic Robot Vision Challenge[1] or, very recently, the Solutions in Perception Challenge[2]. Nevertheless the above competitions concentrate more in what can be achieved with current state of the art techniques than in highlighting open problems.

Besides these competitions, we are not aware of any publicly available dataset where the particular problems of mobile robotics are well represented. To help improve this situation, we have created the IIIA30 dataset, which consists of several sequences acquired navigating with a mobile robot, as well as manually generated ground truth bounding box annotations for 29 different objects.

Moreover, the problems encountered in mobile robotics and embodied in this dataset are very similar to those found in mobile computing, a currently very relevant area of research where low processing power, limited storage space and bad image quality are the rule rather than the exception.

The rest of the paper is divided as follows. First, the IIIA30 dataset and the performance metrics we recommend for it are described in Section II. Next, the dataset is evaluated using the well known cascade of weak classifiers method from Viola and Jones in Section III. Finally, in Section IV, the conclusions are presented.

II. IIIA30 Dataset for Mobile Robot Object Detection

We contribute the IIIA30 dataset[3], that consists of three sequences (IIIA30-1 to IIIA30-3) of different length acquired at approximately ten frames per second by our mobile robot while navigating at approximately 50 cm/s in a laboratory type environment. These sequences are intended to be used as test

[1]http://www.semantic-robot-vision-challenge.org/
[2]http://pr.willowgarage.com/wiki/SolutionsInPerceptionChallenge
[3]http://www.iiia.csic.es/~aramisa/iiia30.html

data to evaluate object detection methods in realistic robot vision conditions in an office type environment.

The camera mounted in the robot is a Sony DFW-VL500 and the image size is standard VGA resolution (i.e. 640×480 pixels). Figure 1 shows some example images with ground truth annotations, and Figure 2 the robotic platform used to acquire the sequences. The environment has not been modified in any way and the object instances in the test images are affected by lightning changes, blur caused by the motion of the robot, occlusion and large viewpoint and scale changes. Since the objective was to deal with a non-trivial multiclass problem, a total of 30 categories (29 objects and background) that appear in the sequences have been considered. The objects have a large range of sizes, and cover a wide range of appearance characteristics: some are textured and flat, like the posters, while others are textureless and only defined by its shape.

Since for most object recognition methods the training stage is performed offline, using more training data does not affect the frame rate of the system at test time. As a consequence, it is natural to try to improve the accuracy by using additional data for training, either downloading it from online repositories or acquiring it manually. The total number of frames is 567 with 2448 annotated objects, that break down as appears in Table I. The average window size are 101.8×133.3 pixels, with a std of 75.4×87.3.

In order to evaluate the influence of additional, good quality, training data, twenty high resolution images and a video were taken with a standard digital camera for each considered object category. Figure 3.a shows the good quality training images for all the object categories, and 3.b shows some cropped object instances from the test images. Of course, it is also equally possible to explore the case of bad quality in both testing and training data by using one of the sequences for training an testing in the remaining two. In order to increase the autonomy of robots in the future, it is highly desirable to have perception algorithms capable of autonomously detecting new object types and learning to recognize them in an unsupervised way. However, this is not what we aim at with the present dataset, since these approaches will possibly need a much larger stream of data than what is available here.

Object	#	Avg area ± Std	Object	#	Avg area ± Std
Grey battery	66	4414 ± 3166	Monitor 2	98	23056 ± 11013
Red battery	307	4025 ± 5696	Monitor 3	78	8041 ± 4166
Bicycle	71	69830 ± 50002	Orbit box	88	2774 ± 1527
Ponce book	85	6877 ± 4016	Dentifrice	87	2524 ± 1332
Hartley book	87	7483 ± 6880	Poster CMPI	42	21108 ± 10759
Calendar	55	6265 ± 3105	Phone	94	7360 ± 3714
Chair 1	112	46828 ± 21795	Poster Mys-trands	73	6233 ± 8189
Chair 2	94	39549 ± 31035	Poster spices	96	17843 ± 10850
Chair 3	91	46731 ± 26653	Rack	89	31467 ± 12540
Charger	19	10051 ± 5481	Red cup	63	3815 ± 1472
Cube 1	28	11660 ± 10473	Stapler	82	5371 ± 3432
Cube 2	18	8152 ± 3895	Umbrella	92	13761 ± 6996
Cube 3	34	11778 ± 3404	Window	211	39337 ± 21453
Extinguisher	30	2312 ± 472	Wine bottle	80	3377 ± 1715
Monitor 1	78	14253 ± 8226			

TABLE I
NUMBER OF INSTANCES AND AVERAGE AREA (IN PIXELS) OF THE OBJECTS IN THE IIIA30 DATASET.

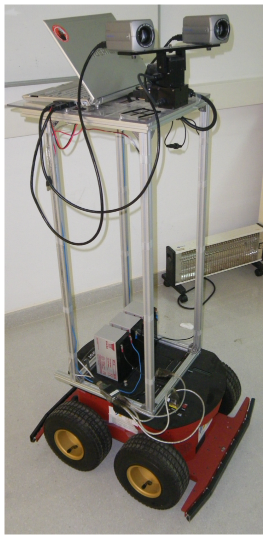


Fig. 2. Robotic platform used in the experiments.

The ground truth information, necessary to evaluate the object detection methods, has been manually generated by annotating with a bounding box each occurrence of an object in each frame of the video sequences, along with its particular image characteristics (e.g. blurred, occluded...).

Although this dataset is of the same-object recognition type, the bad image quality makes it significantly more difficult than most of the other similar datasets.

In order to evaluate the performance of the methods we recommend several standard metrics that are briefly explained next. Precision is defined as the ratio of true positives among all the positively labeled examples, and reflects how accurate our classifier is.

$$Pre = \frac{TruePositives}{FalsePositives + TruePositives} \quad (1)$$

Recall measures the percentage of true positives that our classifier has been able to label as such. Namely,

$$Rec = \frac{TruePositives}{FalseNegatives + TruePositives} \quad (2)$$

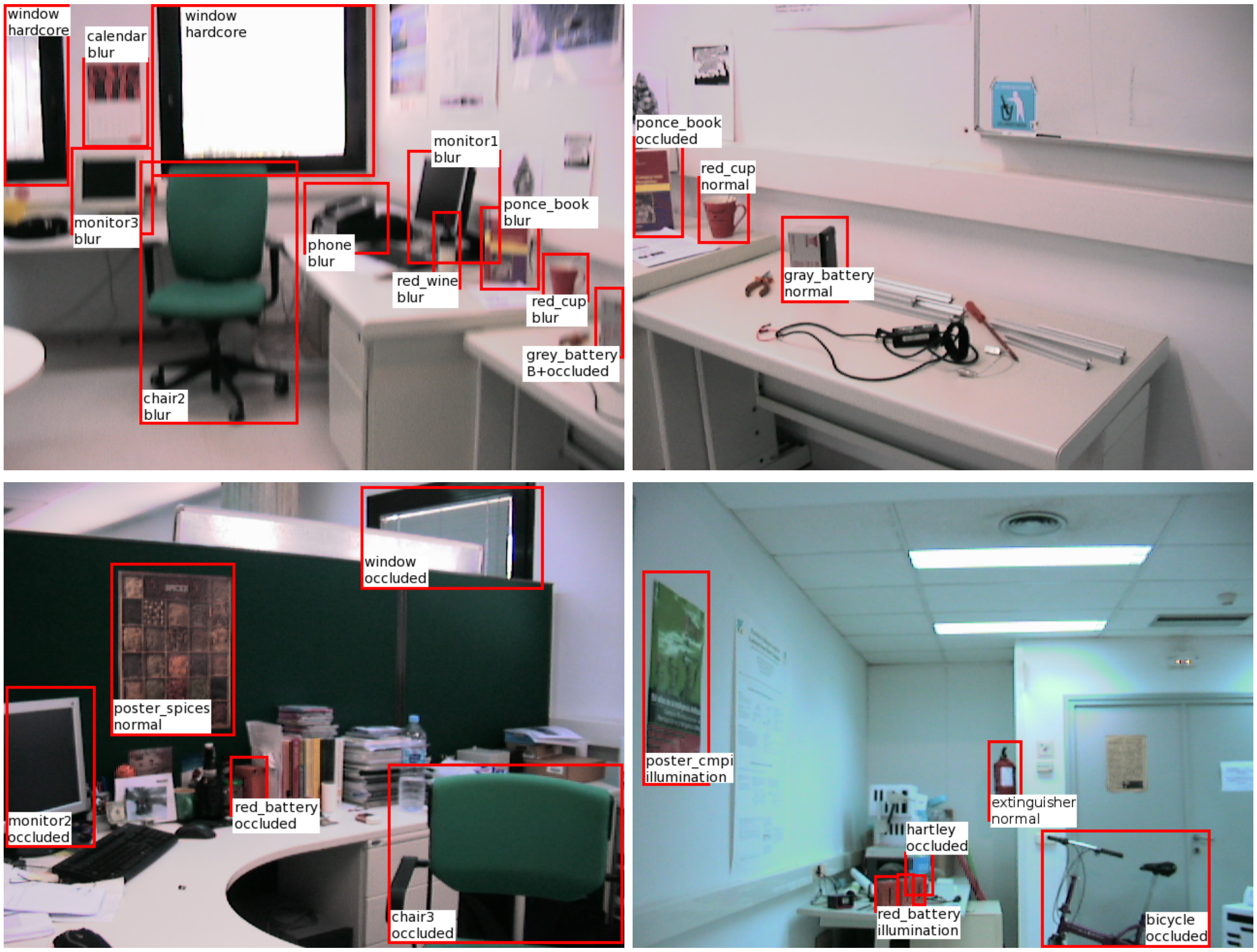


Fig. 1. Four example images of a sequence from the IIIA30 dataset. The top-left one is affected by typical motion blur, which may make detection of the present objects more difficult. Superimposed ground truth bounding boxes can be seen in the bottom-left one.

When it is equally important to perform well in both metrics, we also considered the f–Measure metric:

$$f - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (3)$$

This measure assigns a single score to an operating point of our classifier weighting equally precision and recall, and is also known as f_1–measure or balanced f–score. If the costs of a false positive and a false negative are asymmetric, the general f–measure can be used by adjusting the β parameter:

$$f_g - measure = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \quad (4)$$

In the object detection experiments, we have used the Pascal VOC object detection criterion [3] to determine if a detection hypothesis is a false or a true positive. In brief, to consider an object as a true positive, the bounding boxes of the ground truth and the detected instance must have a ratio of overlap equal or greater than 50% according to the following equation:

$$\frac{BB_{gt} \cap BB_{detected}}{BB_{gt} \cup BB_{detected}} \geq 0.5 \quad (5)$$

where BB_{gt} and $BB_{detected}$ stand for the ground truth and detected object bounding box respectively. For objects marked as occluded only the visible part has been annotated in the ground truth. Since the type of annotation is not compatible with the output of algorithms that estimate the pose of the whole object from the visible part (like the SIFT object recognition method [5]), for the case of objects marked as occluded, we have modified the above formula in the following way:

$$\frac{BB_{gt} \cap BB_{detected}}{BB_{gt}} \geq 0.5 \quad (6)$$

As can be seen in the previous equation, it is only required that the detected object bounding box overlaps 50% of the ground truth bounding box. Another option to deal with

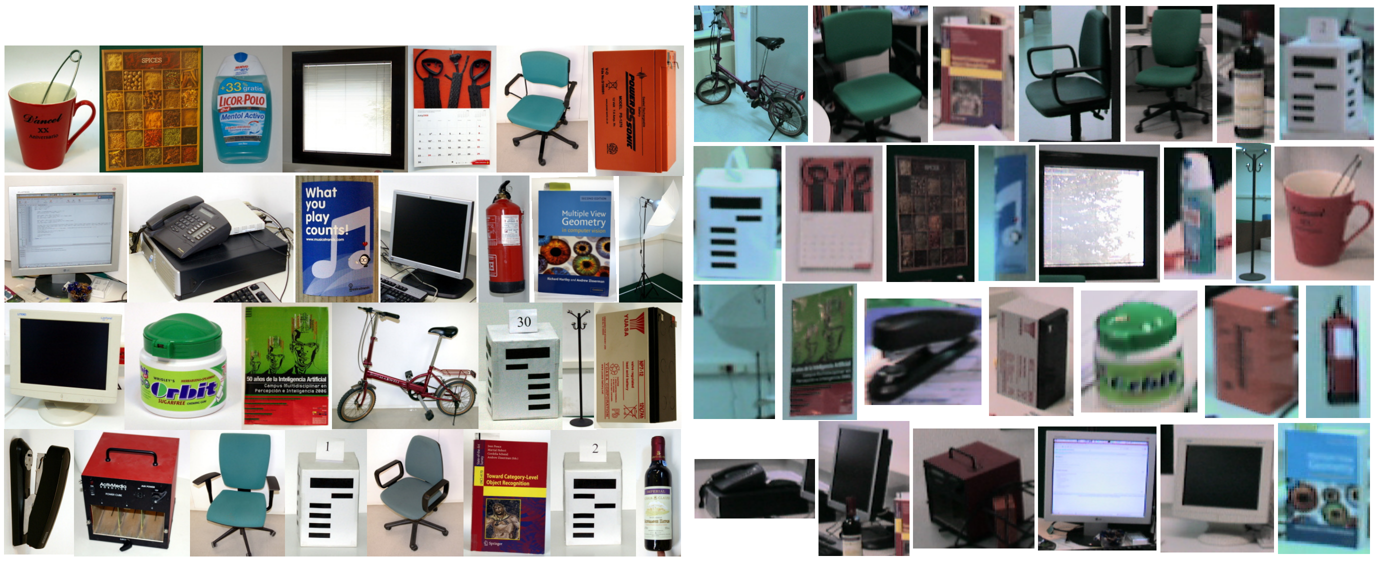


Fig. 3. (a) Training images for the IIIA30 dataset. (b) Cropped instances of objects from the test images.

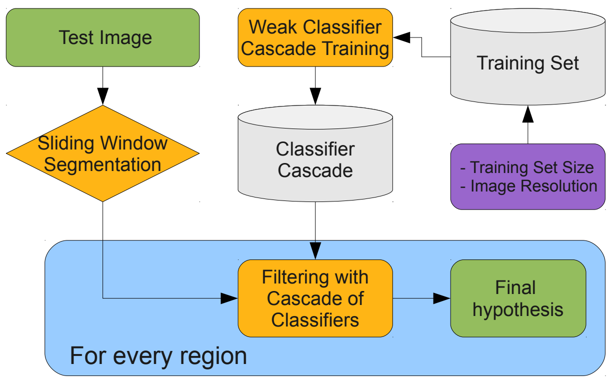


Fig. 4. Diagram of the Viola and Jones Cascade of Weak Classifiers method, with tests shown as purple boxes. Orange boxes refer to steps of the method and green to input/output of the algorithm.

the occluded objects problem could be modifying the object detection algorithm to restrict the predicted bounding box to the part of the object that is believed to be visible, although that would arise other theoretical and technical difficulties.

III. Baseline Results

The cascade of weak classifiers proposed by Viola and Jones [6] is a commonly used object recognition method because of its good performance and low computational cost. A diagram of the steps of the method and the tests conducted can be seen in Figure 4. This method constructs a cascade of simple classifiers (i.e. simple Haar-like features in a certain position inside a bounding box) using a learning algorithm based on AdaBoost. Speed was of primary importance to the authors of [6], and therefore every step of the algorithm was designed with efficiency in mind. The method uses rectangular Haar-like features as input from the image computed using Integral Images, which makes it a constant time operation regardless of the scale or type of feature. Then, a learning process that selects the most discriminative features constructs a cascade where each node is a filter that evaluates the presence of a single Haar-like feature with a given scale at a certain position in the selected region. The most discriminative filters are selected to be in the first stages of the cascade to discard windows not having the object of interest as soon as possible. At classification time, the image is explored using sliding windows. However, thanks to the cascade structure of the classifier it is only at interesting areas where processor time is really spent.

Notwithstanding its well known advantages, this approach suffers from significant limitations. The most important one being the amount of data required to train a competent classifier for a given class. Usually hundreds of positive and negative examples are required (e.g. in [7] 5000 positive examples, derived using random transformations from 1000 original training images, and 3000 negative examples where used for the task of frontal face recognition). Another known drawback is that a fixed aspect ratio of the objects is assumed with this method, that may not be constant for certain classes of objects (e.g. cars). Another drawback is the difficulty of generalizing the approach above 10 objects at a time [8]. Finally, the tolerance of the method to changes in the point of view is limited to about 20°. In spite of these limitations, the Viola and Jones object detector has had remarkable success and is widely used, especially for the tasks of car and frontal face detection.

Since the publication of the original work by Viola and

Object	Recall	Prec	Object	Recall	Prec
Grey battery	0.0	0.0	Monitor 2	0.14	0.14
Red battery	0.28	0.02	Monitor 3	0.03	0.01
Bicycle	0.46	0.07	Orbit box	0.03	0.01
Ponce book	0.0	0.0	Dentifrice	0.0	0.0
Hartley book	0.03	0.01	Poster CMPI	0.17	0.15
Calendar	0.19	0.01	Phone	0.0	0.0
Chair 1	0.11	0.22	Poster Mystrands	0.36	0.27
Chair 2	0.71	0.05	Poster spices	0.46	0.06
Chair 3	0.0	0.0	Rack	0.0	0.0
Charger	0.0	0.0	Red cup	0.0	0.0
Cube 1	0.0	0.0	Stapler	0.03	0.01
Cube 2	0.0	0.0	Umbrella	0.03	0.02
Cube 3	0.0	0.0	Window	0.36	0.2
Extinguisher	0.0	0.0	Wine bottle	0.0	0.0
Monitor 1	0.0	0.0			

TABLE II
RECALL AND PRECISION VALUES OBTAINED TRAINING THE VIOLA & JONES OBJECT DETECTOR USING IMAGES EXTRACTED FROM THE IIIA30-3 SEQUENCE AND EVALUATING IN SEQUENCES IIIA30-1 AND IIIA30-2.

Object	Recall	Prec	Object	Recall	Prec
Grey battery	0.01	0.02	Monitor 2	0.41	0.20
Red battery	0.08	0.04	Monitor 3	0.40	0.18
Bicycle	0.01	0.10	Orbit box	0.10	0.16
Ponce book	0.08	0.31	Dentifrice	0.01	0.03
Hartley book	0.04	0.08	Poster CMPI	0.10	0.05
Calendar	0.11	0.27	Phone	0.07	0.08
Chair 1	0.02	0.30	Poster Mystrands	0.71	0.12
Chair 2	0.01	0.34	Poster spices	0.05	0.05
Chair 3	0.02	0.05	Rack	0.06	0.55
Charger	0.0	0.08	Red cup	0.01	0.05
Cube 1	0.06	0.21	Stapler	0.02	0.20
Cube 2	0.0	0.56	Umbrella	0.05	0.58
Cube 3	0.03	0.24	Window	0.10	0.08
Extinguisher	0.09	0.13	Wine bottle	0.03	0.32
Monitor 1	0.02	0.01			

TABLE III
RECALL AND PRECISION VALUES FOR EACH OBJECT CATEGORY FOR THE VIOLA AND JONES OBJECT DETECTOR WHEN USING A TRAINING SET OF SEVERAL GOOD QUALITY IMAGES PER OBJECT AND WITH SYNTHETICALLY GENERATED IMAGES.

Jones, many improvements to the method have appeared, for example to address the case of multi-view object detection [9], [10]. In this work the original method has been evaluated using a publicly available implementation[4]. We selected the best parameters for the method among the available ones via cross-validation.

Training Set Size and Image Quality: As previously mentioned, one of the most important limitations of the Viola and Jones object recognition method is the amount and quality of the training data. In this work we have evaluated three different training sets. The first one consists of images extracted using the ground truth bounding boxes from the sequence IIIA30-3. The second one consists of 20 good quality training images per object type, and additional synthetic views automatically generated by applying projective transformation to these images. Finally, the third training set is a mix between good quality images extracted from videos recorded with a digital camera (for 21 objects, between 700 and 1200 manually segmented images per object), and a single training image plus 1000 new synthetic views (for 8 objects).

a) ***Training data from robot sequences:*** The training set used for the first test was assembled using images from the IIIA sequences. Consequently, the images were both of low quality and low resolution, and of only between 50 and 70 images for each type of object were available. In Table II the results obtained applying the trained classifiers to sequences IIIA30-1 and IIIA30-2 are shown. As can be seen in the table, the Viola and Jones classifier is able to find only some instances of 11 out of the 29 object categories. This poor performance is expected due to the limited amount and bad quality of the training data.

b) ***One good quality image and synthetic new views:*** Table III shows the results obtained with twenty good quality training images, but further enhancing the set by synthetically generating a thousand extra images for each training sample. As it can be seen, the usage of high quality images and the synthetic views significantly improved the results with respect to the first training set. Most of the objects are found at least a few times, and the precision in particular improved more than 10% with this new training set.

c) ***Large and good quality training set:*** Finally, Table IV shows the results obtained using the third training set, which consisted of hundreds of good quality images extracted from video recordings done with a conventional camera. The usage of additional training data clearly improved both recall and precision by more than 6%.

One known drawback of the Viola and Jones cascade of classifiers is its inability to handle partial occlusions of objects. This is also observed for the IIIA30 dataset in Table IV in the columns that list separately the precision and recall for the partially occluded and the non-occluded object instances. In contrast, blurring and illumination variations did not affect performance significantly. Regarding the object types, (textured, untextured and repetitively textured) textured objects obtained an overall recall of 26% and precision of 33%, similar to that of repetitively textured objects (24% recall and 36% precision). Finally, untextured objects obtained 14% of recall and 19% precision.

The performance on the posters is surprisingly low, as they are usually considered "easy" objects. The most probable explanation is the large changes in point of view that the posters suffer through the video sequences. The time necessary to apply the classifiers for all the classes to one test image is 728 ms on average in a Pentium IV computer using a single core.

IV. CONCLUSIONS

We have presented a publicly available and hard object detection dataset acquired with a mobile robot, that faithfully represents the typical problems encountered in mobile robotics and mobile computing in general (i.e. low resolution, motion blur, etc.). The dataset contains three sequences of varying length, with bounding box annotations for 29 challenging object types, as well as good quality training images.

[4]We have used the implementation that comes with the OpenCV library: http://opencv.willowgarage.com/wiki/

	All		Non-Occluded		Occluded	
Object	Recall	Prec	Recall	Prec	Recall	Prec
Grey battery	0.36	0.24	0.41	0.24	0.0	0.0
Red battery	0.37	0.82	0.44	0.82	0.0	0.0
Bicycle	0.0	0.0	0.0	0.0	0.0	0.0
Ponce book	0.81	0.88	0.86	0.86	0.25	0.02
Hartley book	0.66	0.94	0.70	0.94	0.0	0.0
Calendar*	0.33	0.08	0.38	0.08	0.0	0.0
Chair 1	0.0	0.0	0.0	0.0	0.0	0.0
Chair 2*	0.0	0.0	0.0	0.0	0.0	0.0
Chair 3	0.0	0.0	0.0	0.0	0.0	0.0
Charger	0.12	0.08	0.12	0.08	0.0	0.0
Cube 1	0.22	0.43	0.23	0.29	0.2	0.15
Cube 2	0.23	0.11	0.20	0.09	0.34	0.03
Cube 3	0.28	0.53	0.37	0.48	0.09	0.06
Extinguisher	0.0	0.0	0.0	0.0	0.0	0.0
Monitor 1*	0.0	0.0	0.0	0.0	0.0	0.0
Monitor 2*	0.23	0.57	0.39	0.57	0.0	0.0
Monitor 3*	0.04	0.13	0.05	0.13	0.0	0.0
Orbit box*	0.15	0.03	0.17	0.03	0.0	0.0
Dentifrice	0.0	0.0	0.0	0.0	0.0	0.0
Poster CMPI	0.11	0.34	0.19	0.34	0.0	0.0
Phone	0.05	0.09	0.0	0.0	0.3	0.09
Poster Mystrands	0.0	0.0	0.0	0.0	0.0	0.0
Poster spices	0.04	0.38	0.12	0.38	0.0	0.0
Rack	0.0	0.0	0.0	0.0	0.0	0.0
Red cup	0.89	0.89	0.89	0.89	0.0	0.0
Stapler	0.24	0.21	0.24	0.21	0.0	0.0
Umbrella	0.0	0.0	0.0	0.0	0.0	0.0
Window	0.03	0.40	0.10	0.40	0.0	0.0
Wine bottle*	0.10	0.06	0.10	0.06	0.0	0.0

TABLE IV
RECALL AND PRECISION VALUES FOR EACH OBJECT CATEGORY USING THE VIOLA & JONES OBJECT DETECTOR AND THE THIRD TRAINING SET DESCRIBED . WHEN WE DECOMPOSE THE PRECISION-RECALL VALUES FOR OCCLUDED AND NON-OCCLUDED OBJECTS, RESULTS SHOWS A PERFORMANCE DROP FOR OCCLUDED OBJECTS. THE ASTERISK MARK DENOTES OBJECTS TRAINED FROM SYNTHETIC IMAGES.

In order to set a baseline for future evaluations, we have run the Viola and Jones Cascade of classifiers object detector on the three sequences. This study is part of a larger work evaluating three state of the art object detectors suitable for mobile robotics [11]: the SIFT object recognition system [5], the Vocabulary Tree method [12] and the Viola and Jones Cascade of Classifiers method [6].

Despite the use of very simple image features, the Viola and Jones Cascade of classifiers attains a good level of recall for several objects in a low runtime. Its main drawbacks are the large (in comparison with other techniques) training dataset required to obtain a good performance level, and the limited robustness to changes in the point of view and occlusions of the method, as well as a significant number of false positives that have to be filtered out in later stages. Furthermore, some theoretically "easy" objects, such as the posters, proved to be troublesome to the Viola and Jones method. This is probably due to overfitting to some particular view, or to too much variability of the very rich Haar feature distribution when changing the point of view, where the method was unable to find any recognizable regular pattern.

Nevertheless, the idea of a boosted cascade of weak classifiers is not limited to the very fast but simple Haar features, but any kind of classifier can be used for that matter. A very interesting alternative is using linear SVMs as weak classifiers, since it allows to add a non-linear layer to an already efficient linear classifier. Such idea has been already successfully applied in a few cases [13], [14], and we believe it is a very interesting line to investigate.

ACKNOWLEDGMENTS

This work was supported by the FI grant and by the 2009-SGR-1434 project from the Generalitat de Catalunya, by the Spanish Ministry of Education and Science under projects TRA2010-21371-C03-01 and Consolider Ingenio 2010: MIPRCV (CSD200700018) and by the Rio Tinto Centre for Mine Automation and the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government.

REFERENCES

[1] S. Vasudevan, S. Gachter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots - an object based approach," in Robotics and Autonomous Systems, Volume 55, Issue 5, From Sensors to Human Spatial Concepts, 31 May 2007, Pages 359-371., 2007.

[2] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2007.

[4] N. Pinto, D. D. Cox, and J. J. Dicarlo, "Why is real-world visual object recognition hard?" *PLoS Computational Biology*, vol. 4, no. 1, pp. e27+, January 2008. [Online]. Available: http://dx.doi.org/10.1371/journal.pcbi.0040027

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Interantional Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001, p. 511.

[7] R. Lienhart, E. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *In DAGM 25th Pattern Recognition Symposium*, 2003, pp. 297–304.

[8] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 854–869, 2007.

[9] M. Jones and P. Viola, "Fast multi-view face detection," in *IEEE Conference on Computer Vision and Pattern Recognition*. Citeseer, 2003.

[10] C. Huang, H. Ai, B. Wu, and S. Lao, "Boosting nested cascade detector for multi-view face detection," in *Proceedings of the 17th International Conference on Pattern Recognition*, 2004, pp. 415–418.

[11] A. Ramisa, "Localization and object recognition for mobile robots," Ph.D. dissertation, Universitat Autonoma de Barcelna, 2009.

[12] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," *Conf. Computer Vision and Pattern Recognition*, vol. 2, pp. 2161–2168, 2006.

[13] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *International Journal of Computer Vision*, vol. 63. Springer, 2005, p. 153161. [Online]. Available: http://www.springerlink.com/index/T61K38U53J531344.pdf

[14] D. Aldavert, A. Ramisa, R. Toledo, and R. L. D. Mantaras, "Fast and Robust Object Segmentation with the Integral Linear Classifier," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Optimizing a Humanoid Robot Skill

Luís Rei [1], Luís Paulo Reis [1,2] and Nuno Lau [3,4]

Abstract— **Controlling a humanoid robot with a large number of joints and thus, degrees of freedom, presents a complex problem that requires knowledge in multiple fields, including biology, mechanics, physics, electronics and computer engineering. The challenge of making a humanoid robot play soccer adopted by the RoboCup initiative, a task that was created specifically for humans, is ideal for testing the performance of the robot's motor skills. This work aims to improve the performance of these skills by applying to them an automated optimization process. The robot is the simulated RoboCup 3D agent, a simulated NAO robot, implemented by the FCPortugal3D team. Several different optimization algorithms, in particular Hill Climbing, Simulated Annealing, Tabu Search and Genetic Algorithms are adapted to this problem, used and compared in the optimization of a particular skill of the FCPortugal agent. Furthermore, the skill optimized, which allows the robot to get up after falling on its front, is also compared to the original, unoptimized, skill as well as to those of other teams participating in the RoboCup simulated 3D league. The achieved results are good, providing skill that performs considerably better than the original skill.**

I. INTRODUCTION

The RoboCup international competition uses soccer as a standard problem to foster research in the fields of artificial intelligence and robotics [1]. FC Portugal team project was conceived as an effort to create intelligent players, capable of thinking like real soccer players and behave like a real soccer team [2] competing in the RoboCup simulation leagues. The 3D humanoid soccer simulation league was created in order to promote research in the necessary techniques in order to make humanoid robots play soccer. The topics of research include physics, biology, control theory and machine learning with the aim of developing stable biped skills such as walk, turn, get up and kick. The result of this kind of research may be extended to other domains, such as the use on real humanoid robots, which may be able to perform social tasks such as helping a blind to cross a street or elderly people to perform tasks that became impossible to do alone. The use of simulated environments is popular since it allows the developers to make arbitrary or complex tests in the simulator without using the real robot thus avoiding expensive material to get damaged [3]. The teams have consisted of only a few agents, research in coordination has not been very important in the 3D league. Developing efficient low-level skills has been the main decisive factor in the 3D league [4]. Despite this fact, FC Portugal main focus has been on the high-level, multi-agent coordination [5], [6], [7]. This work instead focus on the low-level performance of a robot and aims to show how the robot's low-level skills can be improved with the aid of automatic optimization methods and provides a comparison between different optimization algorithms adapted to this problem, namely, Hill Climbing, Simulated Annealing, Tabu Search and a Genetic Algorithm. The particular skill being optimized provides the robot with the means to get up after falling on its front and attempts to minimize the time it takes to perform it while maximizing the stability of the robot while, and immediately after, performing this skill. Section II provides information about the problem, specifically the simulation system, the agent, skills in the FC Portugal agent in general and the skill to be optimized in particular. Section III gives an overview of the optimizer developed and the algorithms used. Section IV describes the experiment conducted and its results. Finally, section V analyses the results of the experiment and extracts the conclusions.

1 DEI/FEUP - Informatics Engineering Department, Faculty of Engineering of the University of Porto, Rua Dr. Roberto Frias s/n, 4200 465 Porto, Portugal

2 Artificial Intelligence and Computer Science Lab. Porto, Portugal

3 University of Aveiro Campus Universitário de Santiago, 3810 193 Aveiro, Portugal

4 IEETA - Institute of Electronics and Telematics Engineering of Aveiro, Portugal

II. PROBLEM FORMULATION

A computer simulation, also known as a computer model, is a computer program that attempts to simulate an abstract model of a system. A system is a set of structured interacting or interdependent components that forms an integrated whole. Computer simulations have become a useful part of mathematical modeling of many natural systems in physics (computational physics), astrophysics, chemistry and biology, human systems in economics, psychology, social science, and engineering.

The official Robocup 3D simulation server is Simspark [8] a generic physical multi-agent simulator system for agents in three-dimensional environments. It simulates the laws of physics in the real world in the context of a soccer game (i.e. the soccer field, ball and rules of the game) as well as the robots (physical dimensions, look, sensors and joints).

The robotic platforms (either the most simple articulated arms or the most complex humanoid robots) are usually very expensive. The use of simulation environments for research, development and test in robotics provides many advantages over the use of real robots [9], [3], [10]. The main advantages of the simulation are:

- Less expensive than real robots;
- Easy development and testing of new models of robots;
- Easy testing of new algorithms;
- Less development and testing time;
- All tests can be done without damaging the real robot;

- For repetitive tests (e.g. optimization processes), the use of a virtual model is better because the robot will not need assistance to reinitialize every iteration;
- With the quality of simulation environments that exist today, is is possible to use the results obtained by simulation in the real robots with just a few changes;
- Control over the simulation time;

The obvious disadvantage, in this case, is that the simulated system is never equal to the real system. As such, software agents developed for the simulated system will require tweaks in order to be used in the real system. How much will have to change depends on the accuracy of the simulation.

A. *The Simulation System*

Simspark is a generic simulation platform for physical multi-agent simulations. This simulator was developed over a flexible application framework (Zeitgeist) to be a generic simulator, capable of simulating anything, since the launch of a projectile to a big soccer game. The framework facilitates exchanging single modules and extending the simulator [11]. The simulation consists of three important parts [8], [12]: the server, the monitor and the agents.

The SimSpark server hosts the simulation process that manages the simulation. It is responsible to advance the simulation, i.e. modify the simulation state in a continuous run loop. Objects in the scene can change their state which includes properties like position, speed or angular velocity due to several influences. The objects are under the control of a rigid body physical simulation that resolves collisions, applies drag, gravity etc. Agents that take part in the simulation also modify objects with the help of their effectors. Another responsibility of the server is to keep track of connected agent processes. Each simulation cycle the server collects and reports sensor information for each of the sensors of all connected agents. It further carries out received action sequences that an agent triggers using its available effectors.

The SimSpark monitor, seen in figure 1, is responsible to render the current simulation. It connects to a running server instance from which it continuously receives a stream of update data that describes the simulation states either in full or as incremental updates relative to the preceding state. The monitor can further be configured to read a protocol of scene updates from a file and act as a logplayer. In this mode it does not connect to a server instance but replays a recorded game. The format of the logfile is identical to the monitor protocol used on the network.

B. *The Simulation Agent*

RoboCup 3D humanoid soccer league currently uses a virtual model of the NAO robot which is manufactured by Aldebaran Robotics. Its height is about 57cm and its weight is around 4.5Kg. Its biped architecture with 22 degrees of freedom allows NAO to have great mobility. The simulated version is quite similar to the real robot, as can be seen in figure 2.

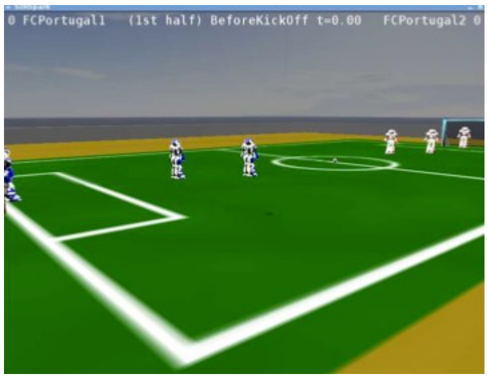


Fig. 1. Simspark Monitor screenshot.

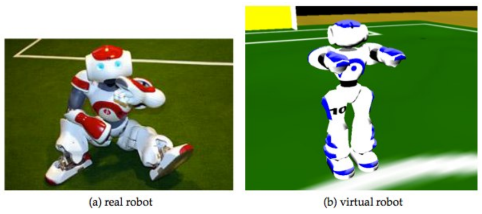


Fig. 2. The NAO humanoid robot: real vs simulated. Adapted from [8]

The agent used for this work is the implementation of this RoboCup simulated agent made by the FC Portugal team.

C. *Humanoid Robot Skills*

A skill or behavior consists in a purposeful, repeatable action taken by the executing agent such as getting up from a fall, kicking the ball or walking around the ball. These can be specified once and executed as many times as necessary. In the FC Portugal Simulated Humanoid agent, these behaviors are specified and stored in an XML file, created by the team, which provides the parameters for an automatic behavior generation method. Each skill has its own XML file and all skill specification files are loaded when the agent starts.

Skills, specified in XML files, are executed by an automatic trajectory planner. A trajectory can be defined as the set of points followed by an object over an interval of time. In the case of robotic joints, trajectory planning consists of breaking the joint space into many start and end points during the time interval. A trajectory planner can generate skills (e.g. walk, turn, get up) by computing different joint trajectories [10]. There are various methods for

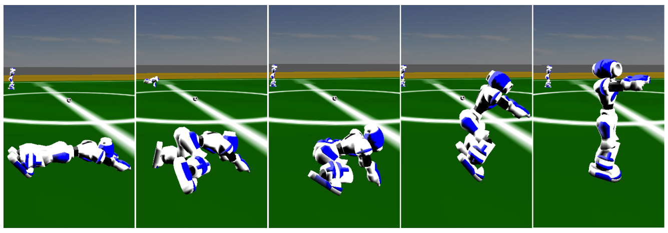


Fig. 3. Sequence of images showing the execution of the behavior specified by GetupFront.xml.

SlotBehavior Variables	
Scope	**Parameter**
Slot	delta (δ)
Joint	joint identifier target angle (θ)

TABLE I
VARIABLES THAT CAN BE SPECIFIED IN A SLOT BEHAVIOR.

generating trajectories automatically, some of those used by the FC Portugal team include a Step-based method [10], Sine Interpolation, a simplified version of the method proposed in [13] and a Central Pattern Generator, based on the work of Behnke [14].

The skill optimized in this work, denominated GetUpFront, is implemented using Sine Interpolation which is meant to give control over each joint trajectory by defining an interpolation of some smooth function for an interval of time between the current angle and the target angle. It allows defining not only the target angles, but also the time in which those angles should be achieved, as well as control the initial and final angular velocities. The central concept of this method is the slot, an interval of time from 0 to δ, where several joints are moved in parallel. In each slot, the controller will interpolate between the current angle and the desired angle by performing a sine-like trajectory in a specified amount of time.

In the FCPortugal agent, the skills that use this generator are called Slot Behaviors, here the term “Behavior” is interchangeable with the term “skill”. A skill specification file consists of a series of slots, which will be executed sequentially. Each slot can specify movements for different joints as well as other parameters. The following variables can be specified for a skill:

The GetUpFront skill consists of 9 different slots, with the first slot being a “reset slot” which places the robot’s joints in an initial state to execute the rest of the skill and the last slot placing the robot in a start position for walking.

III. THE OPTIMIZER

The optimizer was developed to allow any skill of the agent to be optimized using either a single computer or multiple computers connected via a network. Figure 4 is a diagram representing the configuration of the optimizer.

Once started with the proper parameters, the optimizer executes the simulator script, a shell script which starts the simulator and the monitor. It may start multiple simulators and their respective monitors in different computers if specified. It also executes the agent script which in turn starts the FCP agents. The agents connect to the simulation server and to the optimization server which will provide the agents with the behavior to execute and receive from the agents the performance data from that execution. Both these connections are made via TCP sockets.

The optimization server is the core of the optimizer. Apart from being responsible for the execution of the other components via the execution of scripts, It contains the functions

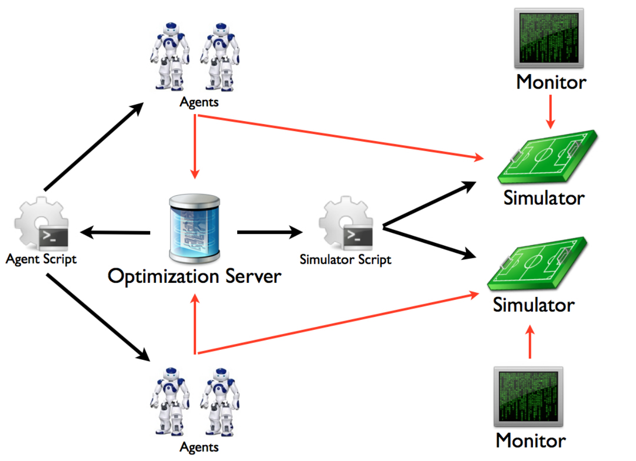


Fig. 4. A diagram of the optimizer’s configuration. Red arrows indicate data transmission with the tip of the arrows indicating the direction in which the connection is established. Black arrows indicate an execution call via a system execution command with the tip indicating which component is being executed.

that read a behavior, run the optimization algorithm, modify the behavior according to the algorithm, send the modified behavior (proposed solution) to the agents for execution, receive the execution data from the agents, evaluate the data according to the specified objective function and, finally, terminates the optimization process via another script.

The optimization server is actually multi-threaded. A thread is started for each agent which creates a specific TCP server which provides an agent with the behavior to optimize, waits for the agent to finish executing the behavior, receives the experimental data sent by the agent and evaluates it, giving that generated behavior a score according to an objective function.

In subsection II-C it was explained that behaviors are specified and stored as XML files which are loaded when an agent starts. The optimization server has functions which read the behavior to be optimized from its XML file and load the behavior into an array in memory. It is this array representation of the behavior that will be modified by the optimization algorithm. This representation is then transmitted to the agents, via a socket, to be executed. To perform this function, the optimization server creates a TCP server to which the agents connect.

An agent script, specific to each behavior to optimize, is responsible for starting the FCP agents and providing them with their parameters. This allows the agents to be started on the local machine or in a remote machine. Another shell script is responsible for terminating the agents. In the actual implementation, for convenience, the simulator script is started from the agent script. The simulator script starts both the simulation server and the monitor.

The scripts exist for both flexibility and to isolate the optimization server from the specifics of running the other components. Recompiling the optimization server each time we wanted to change the parameters of the FCP agents such as the IP addresses of the simulation or optimization server changed would be a poor design choice.

A. *Optimization Algorithms*

Several problems aim at finding the best configuration of a set of parameters to achieve a goal (or some goals). The goal is either to minimize or to maximize some quantity. This quantity is a function of one or more variables and is known as the **objective function**, f. The independent variables that can change while searching for an optimum are known as the **decision variables**. If the goal is to minimize f, then f is called the cost function (or sometimes, the penalty function). When the goal is maximization, f is referred as the utility function (or sometimes, the benefit function). For some problems, the values that decision variables can take are further specified through a number of conditions known as constraints.

The search space of a problem is defined as the set of all candidate solutions. A candidate solution corresponds to an instantiation of the decision variables, if it satisfies all the constraints of the problem then it is called a feasible solution or just a solution. The solution space is the set of all (feasible) solutions. An optimal solution is a solution for which the objective function evaluates to an optimum (minimum or maximum). Computer algorithms that iteratively traverse the search space with the purpose of finding optimal solutions are called automatic optimization algorithms. A candidate solution can also be referred to as an individual. Optimization problems may be broadly divided into two main categories: individual-based methods and population-based methods. Individual-based methods deal with only one current solution. Conversely, in population-based methods counts with a set of individuals (population) that are handled simultaneously.

The Hill Climbing (HC) algorithm [15], [16] is the simplest algorithm implemented. Starting with the initial solution, the skill specification file created by the team, it generates a neighboring solution from the current solution by copying it and randomly modifying a single random decision variable. The solution is then evaluated and if it is better than the current solution, the algorithm makes it the current solution. This algorithm easily gets stuck in a local optima solution, i.e., none of the neighbors has a better evaluation than the current solution, which might be a poor quality solution. A possible strategy to solve this problem is to, sometimes, accept worse solutions.

Simulated Annealing (SA) [17] implements a mechanism for escaping from local optima, accepting a solution worse than the current one with a probability, depending on a specified value known as *temperature*, that decreases along the optimization progress, by a factor known as *cooldown*, and the difference in evaluation between the current score and the new (worse) solution. Aside from the *temperature*, another value, known as *restart* can be specified which returns the algorithm to the best solution found after the specified number of iterations.

Another problem is the effect of cycling through solutions. This may be solved by introducing memories to remember the nodes already visited, as with Tabu Search (TS) [18]. The main characteristic of TS is the systematic use of memory. While most exploration methods keep in memory essentially the best solution and its evaluation score, TS also keeps a list of the last solutions visited. In essence, TS declares each node already visited as a *tabu*. Tabus are stored in a list, the *tabu list*, and the search in the neighborhood is restricted to the neighbors that are not in the *tabu list*. The implemented TS algorithm also uses an additional *tabu list* for the search direction (determined via the gradient) to prevent searching in the direction of previous, worse, solutions.

All methods presented so far were local and individual-based, unlike Genetic Algorithms (GA) [19], an optimization method inspired by the evolution of biological systems and based on global search heuristics. In spite of being different, evolutionary algorithms share common properties since they are all based on the biological process of evolution. Given an initial population of individuals (also called chromosomes), the environmental pressure, applied by the fitness (objective) function, causes the best fitted individuals to survive and reproduce more. Each individual (chromosome) is a set of variables (genes) and represents a possible solution to the optimization problem. The algorithm starts by creating a new population of individuals. Typically, this population is created randomly but any other creation function should be acceptable. The genes of each individual should be inside a range of acceptable values (variable domain) that is defined for each gene. The algorithm then starts the evolution which consists of creating a sequence of new populations. At each step, the algorithm uses the individuals in the current population to create the next population by applying selection, crossover and mutation operators. These genetic operators can be described as follows:

- **Selection**: Specifies how the GA chooses parents for the next generation. The most common option is the roulette option which consists of choosing parents by simulating a roulette wheel, in which the area of the section corresponding to an individual is proportional to its fitness value;
- **Elitism**: Defines the number of individuals in the current generation that are guaranteed to survive in the next generation;
- **Crossover**: A crossover function performs the crossover of two parents to generate a new child. The most common are the scattered function and uniform crossover, the first creates a random binary vector and selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent.
- **Mutation**: The mutation function produces the mutation of children. The most common is the uniform mutation which applies random variations to the children using an uniform distribution. Uniform mutation receives a parameter, p_m which corresponds to the probability that an individual entry has of being mutated.

Algorithm	Parameter	Value
Common	Number of Experiments	14
	Threads	6
	Minimum Change for Angles	-5.0
	Maximum Change for Angles	5.0
	Minimum Change for Deltas	-0.4
	Maximum Change for Deltas	0.4;
HC	Number of Iterations	400
SA	Number of Iterations	400
	Temperature	450.0
	Cooldown Factor	0.95
	Restart	30
TS	Number of Iterations	400
	Tabu List Size	1000
Genetic Algorithm	Number of Generations	100
	Population Size	24
	Number of Elite Solutions	6
	Mutation Probability	0.5
	Crossover Probability	0.2

TABLE II
ALGORITHM PARAMETERS.

IV. Experiments and Results

A. Experiment Setup

The GetUpFront skill consists of 9 different slots, all slots where optimized except the first slot which serves as a ”reset slot”. For the last slot, which sets the position of the robot to initial position for the walk, only the delta was optimized. Because the behavior was designed so that the right side and left side joints moved simultaneously to the same angles, this constraint was enforced during the optimization process. A total of 39 different variables, consisting of joint angles and slot deltas (see table I), are used in the optimization of this skill.

Table II shows the algorithm parameters for the experiment. The number of experiments means that each time a solution is evaluated, the skill is tested that many times and the skill's resulting score is an average of the result for each test. The minimum and maximum change parameters are the lower and upper bounds, respectively, for the random change applied to a solution in order to create a neighboring solution. All other parameters were explained in subsection III-A. The following evaluation function is used:

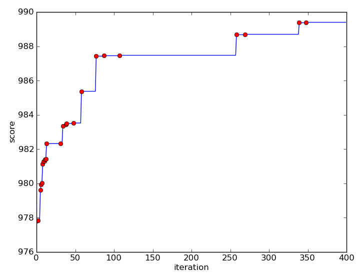
$$f = 1000 - t * w_t - s * w_s \tag{1}$$

Where t is the time taken to execute the GetUpFront skill, $w_t = 10$ is the weight associated with the time, s, with an associated weight $w_s = 500$, is a binary measure of the stability of the agent and is set according to:

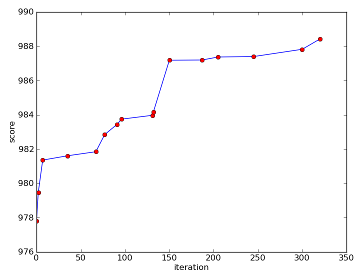
$$s = \begin{cases} 1 & \text{if the agents falls during execution of the skill} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

B. Results

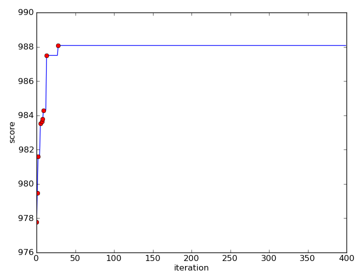
Figure 5 shows the evolution of the objective function's score over the course of the algorithm's iteration. The red dots show where better solutions were found. Table III shows the performance of the skill, measured by the execution time, after being optimized by each algorithm as well as the original skill's performance.



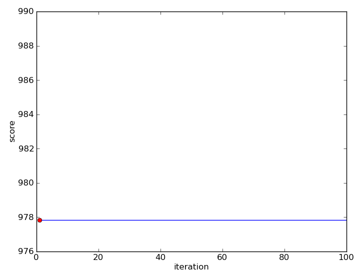
(a) Hill Climbing



(b) Simulated Annealing



(c) Tabu Search



(d) Genetic Algorithm

Fig. 5. Optimization of the GetUpFront skill: Score vs Iterations

The best result, obtained by optimization with HC, is 1.06s which represents an improvement of approximately 50% over the original skill. It is also useful to compare the best result obtained to other teams participating in the RoboCup challenge which is done in table IV. For this end, the execution time of the equivalent skill of the champion and vice-champion teams of the 2010 edition of the RoboCup

Algorithm	Execution Time
Original	2.00 (s)
Hill Climbing	1.06 (s)
Simulated Annealing	1.37 (s)
Tabu Search	1.19 (s)
Genetic Algorithm	2.00 (s)

TABLE III
RESULTS FOR THE OPTIMIZATION OF GETUPFRONT - VARIOUS ALGORITHMS COMPARED.

World Soccer Cup 3D simulation league, Apollo3D and Nao Team Humboldt, respectively, are also presented. The execution times for these teams are approximations obtained by analyzing the logs files of the final matches.

Team	Skill Execution Time
FC Portugal (original)	2.00 (s)
FC Portugal (optimized)	1.06 (s)
Apollo3D	2.00 (s)
Nao Team Humboldt	2.00 (s)

TABLE IV
TIME FOR THE AGENT TO GET UP AFTER FALLING ON ITS FRONT.

V. CONCLUSION

Humanoid robots are not currently on par with humans in performing the most basic of tasks, such as getting up or walking sideways. This thesis offers an automated process of reducing this difference by improving existing skills with the aid of automatic optimization methods. The initial work was made to create an optimization framework for the skills of the FC Portugal 3D humanoid agent which inhabits the simulated world of the RoboCup 3D simulation server. This required both modifications to the agent, the simulation server and the creation of the optimizer itself. The configuration of the optimizer allowed for easy distributed optimization using multiple agents and multiple simulations executing concurrently in multiple computers over a network. Subsequently, different optimization algorithms were implemented, namely Hill Climbing, Simulated Annealing, Tabu Search and a Genetic Algorithm. These algorithms were then used to optimize the GetUpFront skill, which uses Sine Interpolation for automatic trajectory planing The results were definitively good. The skill improved its execution time by 50% from around 2s to around 1s. The resulting skill is faster than the original skill as well as the equivalent skill implemented by other teams participating in the RoboCup 3D simulation league.

The two best algorithms, in terms of end results, were Hill Climbing which found the best solution and Tabu Search which found a solution very close to that found by HC. TS was, however, much faster than HC at the finding solution, as evidenced by figure 5, and even though the solution it found was not as good, it was only marginally worse. The implemented Genetic Algorithm failed to find better results for the GetUpFront skill. Its global search strategy does not seem adequate to optimize skills which, in spite of a very large search space, have an initial state very close to the optimum.

VI. ACKNOWLEDGMENTS

This work was supported by the Fundação para a Ciência e Tecnologia through the project ”INTELLWHEELS - Intelligent Wheelchair with Flexible Multimodal Interface” (grant ref. FCT/RIPD/ADA/109636/2009) and DEI/FEUP - Department of Informatics Engineering of the Faculty of Engineering of the University of Porto.

REFERENCES

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawai, and H. Matsubara, “RoboCup: a challenge problem for AI and robotics,” in *RoboCup-97: Robot Soccer World Cup I*, 1998, pp. 1–19.
[2] L. P. Reis and N. Lau, “FC portugal team description: RoboCup 2000 simulation league champion,” *ROBOCUP-2000: ROBOT SOCCER WORLD CUP IV*, pp. 29—40, 2001.
[3] H. H. Lund and O. Miglino, “From simulated to real robots,” in *International Conference on Evolutionary Computation*, 1996., pp. 362–365.
[4] N. Shafii, L. P. Reis, and N. Lau, “Biped walking using coronal and sagittal movements based on truncated fourier series,” in *RoboCup-2010: Robot Soccer World Cup XIII*, ser. LNAI, no. 6556. Heidelberg: Springer, 2010, pp. 324–335 (to appear).
[5] N. Lau and L. P. Reis, *FC Portugal - High-level Coordination Methodologies in Soccer Robotics*. Vienna, Austria: Itech Education and Publishing, December 2007, pp. 167–192.
[6] L. Mota, L. P. Reis, and N. Lau, “Multi-robot coordination using setplays in the middle-size and simulation leagues,” *Mechatronics, Elsevier*, 2010 (in press).
[7] L. Rei and L. P. Reis, “Brain: A deliberative cibermouse agent,” in *Proceedings of the 10th Conference on Mobile Robots and Competitions in conjunction with the 10th Portuguese Robotics Open - ROBOTICA 2010*, Leiria, Portugal, March 2010, pp. 35–40.
[8] J. Boedecker, K. Dorer, M. Rollmann, Y. Xu, and F. Xue, *SimSpark User’s Manual*, 1st ed., 2008.
[9] J. Nicolas, “Artificial evolution of controllers based on non-linear oscillators for bipedal locomotion,” Master’s thesis, École Polytechnique Féderale De Lausanne, 2004.
[10] H. Picado, M. Gestal, N. Lau, L. P. Reis, and A. M. Tomé, “Automatic generation of biped walk behavior using genetic algorithms,” in *10th IWANN 2009*, ser. LNCS, vol. 5517. Salamanca, Spain: Springer, June 2008, pp. 805–812.
[11] O. Obst, M. Rollmann, and M. Rollmann, “Spark - a generic simulator for physical multi-agent simulations,” in *Computer Systems Science and Engineering*, 2004.
[12] J. Boedecker and M. Asada, “Simspark - concepts and application in the robocup 3d soccer simulation league,” in *SIMPAR-2008 Workshop on The Universe of RoboCup Simulators*, vol. CD-ROM, 2008.
[13] J. Lima, J. Gonçalves, P. Costa, and A. Moreira, “Humanoid robot simulation with a joint trajectory optimized controller,” in *Emerging Technologies and Factory Automation*, ser. ETFA. IEEE, 2008, pp. 986–993.
[14] S. Behnke, “Online trajectory generation for omnidirectional biped walking,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, FL, USA, May 2006, pp. 1597–1603.
[15] O. the Mapping Problem, “Shahid bokhari,” *IEEE Transactions on Computers*, vol. C-30 (3), pp. 207–214, 1981.
[16] D. S. Johnson, C. H. Papadimtriou, and M. Yannakakis, “How easy is local search?” *Journal of Computer and System Sciences*, vol. 37, no. 1, pp. 79–100, 1988.
[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.
[18] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computer and Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.
[19] J. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

Humanoid Soccer Robot Motion Planning using GraphPlan

Nima Shafii, Lucio Sanchez Passos, Luis Paulo Reis,
Artificial Intelligence and Computer Science Laboratory
Faculty of Engineering of the University of Porto - FEUP
Porto, Portugal
{*nima.shafii, pro09026, lpreis* }*@fe.up.pt*

Nuno Lau
Instituto de Engenharia Electronica e Telematica de Aveiro (IEETA) ,Universidade de Aveiro,
Aveiro, Portugal
Nunolau@ua.pt

Abstract— Humanoid robotics is being studied by various fields, because it can be an interesting test bench for different technologies. Furthermore, methodologies for planning actions in advance are widely used for mobile robot's movements in static environments and are being extended and adapted to dynamic environments, such as soccer games between humanoid robots. The choice of the methodology to achieve the plan is vital to accomplish the final goals. This paper reports the modelling and implementation of a path planning methodology for humanoid robots. The main contribution relies on applying planning techniques and iterative deepening search to determine the best sequence of actions that lead the robot to traverse the optimal path to the ball so that it can kick the ball to a given pre-defined target. Some results are presented and discussed to prove the efficiency and reliability of the approach.

Keywords - Path Planning; Humanoid Robots; Robotic Soccer.

I. INTRODUCTION

Robotics is a branch of engineering that involves various fields as mechanics, electronics and computer science. Thus, robots with a human form always aroused curiosity and, in the last few years, the idea to have a perfect humanoid robot is becoming more real. In the humanoid robot field, RoboCup which includes a soccer competition using NAO humanoid), is giving huge contributions.

One interesting way to achieve better results is to apply Artificial Intelligence (AI), in scenarios as learning how to walk, planning objects manipulation, path planning, and so forth. The context of this work is path planning for humanoid robots in a soccer based environment.

In such continue and real-time environment, Footstep Planning [1] may used in the path planning task. It discretizes the possible actions of the robot into a small set of well-chosen actions as different foot placements is not a proper solution due to the fact that its search graph grows exponentially with the number of steps. Only a small number of footsteps can be explored in a real-time system [2].

We have to compute a trajectory for the body-center of a humanoid robot by approximating the shape of the robot. The shape of this trajectory depends on the humanoid robots' ability of low level biped locomotion skills, such as walking curving and etc.

In previous works in this research topic, humanoids often have basic low skills for performing their tasks and following their path trajectory, including forward walk, turn, and different curve skills [3]. Fortunately, in our recent project on biped locomotion we can achieve to omni directional walking, it means that the robot is able to walk in forward and backward direction, different type of walking while curving and turning on the spot [4], [5]. Therefore, it has the capability of following many path trajectories. This flexibility is an advantage, however it increases the complexity of the path planer dramatically.

In motion planning the number of degrees of freedom is usually small which opens the door for the application of search techniques. Humanoid robots usually have more than ten degrees of freedom. Their feet can be placed with a great precision and changing the body posture allows to overcome obstacles that wheeled robots fail in passing through.

In this paper, we present a scenario where the robot, starting from a position far from a ball, has to approach it and then kick it to the target, using the best possible path. To execute this, the robot has to construct the space state and search for the solution, spending the shortest possible time. Also, environment and robot constraints were reviewed for system implementation. We simulated it with RoboCup official 3D simulator to evaluate the planner performance.

The remaining of this paper is organized as follows. Section II presents the characteristics of an humanoid robot and the humanoid simulator used In the next section, we discuss the motion planning related work to understand what already is developed relating it with humanoid robotics. The fourth section presents our methodological approach, experimental results, and finally section V presents the conclusions and suggestions for future work.

II. HUMANOID ROBOTICS

A humanoid robot is a robot with its overall appearance, based on human body, and with the ability to move on its own legs. The number of joint actuators indicates the number of Degree of Freedom (DOF). Like humans, humanoid's body moves in three planes, including transverse (axial), frontal (coronal) and sagittal Planes. Sagittal plane indicates the vertical plane running from front to back and dividing the body into left and right sides. Frontal or Coronal plane is a plane, perpendicular to the sagittal plane, running from side to side and dividing the body into front and back.

The creation of humanoid robots has been motivated by the idea of having a device capable of operating in environments made by and for humans with minimal change to those environments. These machines are expected to perform autonomously most of the functions a person is capable of. These include climbing stairs, reaching for objects, etc.

One of the best known applications of humanoid robotics is the humanoid league of RoboCup. The goal of the RoboCup initiative is to develop a team of humanoid robots able to win against the official human World Soccer Champion team until 2050. The humanoid league is concerned with skills as: dynamic walking, running, kicking the ball while maintaining balance, teamwork, and self-localization. The smaller competition category is KidSize (30-60cm). There is also a humanoid standard platform league which uses the NAO humanoid robot [6].

The French company Aldebaran Robotics developed NAO. The first prototype was released in 2005. Fig. 1 shows NAO, a 58cm height robot with 22 degree of freedom (DOF), counting on different sensors distributed in key parts of it [7]. The Nao model is used in the RoboCup 3D simulation league, using Linux and based on spark technology. In this work we used the RoboCup 3D simulator to implement and test our prototype. Further information about the simulator will be given on subsection 5.2.

Height		58 cm
Mass (including batteries)		4,3 kg
Degrees of Freedom		22
CPU		x86 AMD GEODE 500 MHz
Sensors	Head	2 cameras and microphones
	Chest	Ultrasound, 3-axis accelerometer and a 2-axis gyro-meter
	Legs	4 Force Sensitive Resistors
Autonomy		90 min. (constant walking)
Programming Languages		C++, C, Python, Urbi, .Net

Figure 1. General Caracterestics of Nao

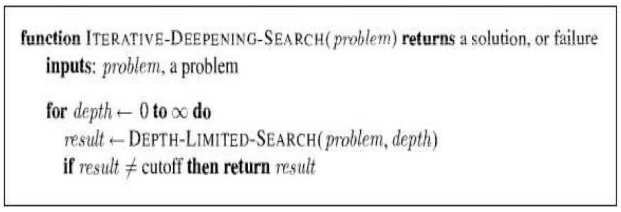
III. Related Concepts and Technologies

Robots have four basic components: sensing, actuating, planning, and control. Since we are focusing in the planning component, it is logic to ask: what is planning? To plan is abstract the process to choose and, also, organize a sequence of actions to achieve some goal. Nevertheless, to construct a plan for the real world is a hard task, so the classical planning often uses some constraints, such as: atomic time, closed world, deterministic effects of actions, and the planner has omniscience of knowledge [8].

There are several strategies of planning; however, we are going to focus on the one used in this work. First, explaining the data structure used to represent the problem (and consequently the search space), then the search technique and the approach used to model the problem.

There are various ways to search a tree structure and here we are going to focus on uninformed search (also called blind search), specifically on Iterative deepening depth-first search (or Iterative deepening search). The uninformed search means that the algorithm has no additional information other than the capacities to generate successors and to distinguish a goal state from a non-goal state.

Iterative deepening search is a general strategy, often used in combination with depth-first search, which finds the best depth limit. Its algorithm (shown in Figure 2) also combines the benefits of depth-first search (modest memory requirements, precisely $O(b^d)$), and breadth-first search (branching factor is finite and optimal when the path cost is a non-decreasing function of the depth of the node) [8]. However, it may seem wasteful, because states are generated multiple times, being this not very costly. Also, based on the total number of nodes the time complexity is $O(b^d)$.



```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution, or failure
  inputs: problem, a problem

  for depth ← 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
    if result ≠ cutoff then return result
```

Figure 2. Pseudo-code of Iterative deepening search [8]

Albeit we did not use GRAPHPLAN directly in this work, it has a great importance for this project in the task of problem modeling. GRAPHPLAN is a planning technique, which represents its plans on graph form. Two nodes compose this graph: propositional nodes (indicate the world state in a time instant. i.e. preconditions), and action nodes (represents the possible actions that can be performed with the indicated preconditions); also, the actions create a new propositional nodes with their effects. So, this idea of preconditions and possible action was used to model our problem (presented in subsection 5.1). Here we just gave a brief overview of GRAPHPLAN's planning graph and for further information consult reference [9].

IV. Methodological Approach

In this section, we discuss the followed steps, covering from modeling to system implementation. In general, the methodology consists of: defining our scenario, finding all constraints related to it, designing the system architecture, integrating the developed module with the simulator, and validating the approach showing robot's trajectory constructed with the planner.

A. Modeling

In this section, we describe how our problem was modeled, and which abstractions were used to achieve it. We also describe how to identify complexities and constraints and, from that, simplify the model without losing key aspects, such as a certain degree of realism. We divided the modeling in various steps to be more intelligible, because it has a long sequence of details to be understood.

The problem was retired from soccer games with humanoid robots, being the chosen situation extremely frequent during these games. Thus, solving this situation efficiently is vital in the team overall performance. The scenario is presented in Figure 3, consisting it in a robot distant *d* from the ball with a different orientation of the ball's. The final goal is to put the ball in the target by kicking it and achieving to the condition of kicking in the most optimal way.

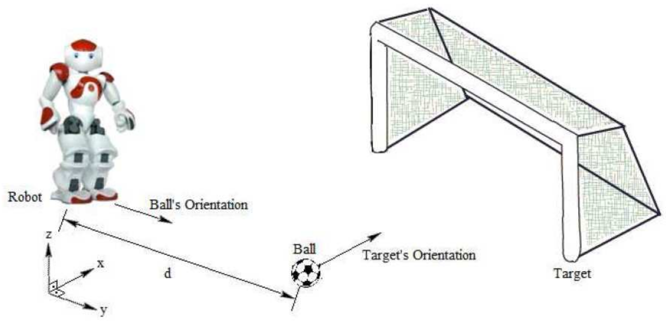


Figure 3. Used scenario

Further, observing the real robot operations during a soccer game the following actions were identified:

- Straight walk
- Curve walk
- Turn in place
- Kick

Also, the possible states were abstracted using some questions. We found all states cited above and the questions their try to answer:

- Having ball (Ha Ball) - is the ball near to robot or not?
- Ball aligned (Ball Dir) - is robot's direction aligned with the direction of ball to target?
- Ball in the Target (Ball Targ) - is the ball in target or not? (after kicking).

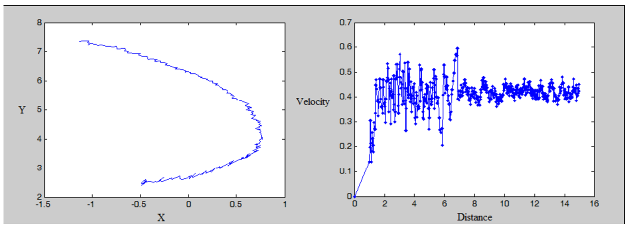
Table 1 shows the correlation between action and states. It was based on GRAPHPLAN modeling, extracting concepts of precondition to execute an action and effects of each action. Thus, to construct all correlations we enumerate various situations and extract them. The action Curve Walk is more complex than others, so we need to analyze it deeper.

TABLE I. PRECONDITIONS AND EFFECTS OF ALL ACTIONS

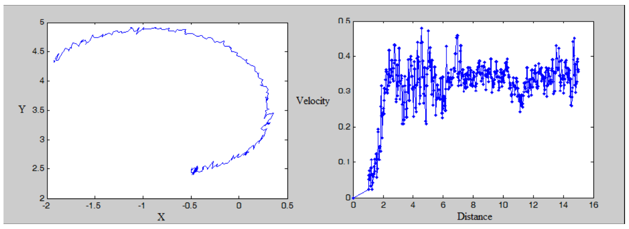
Actions	Preconditions	Effects
Straight walk	¬Ha_Ball, Ball_Dir, ¬Ball_Targ	Ha_Ball, Ball_Dir, ¬Ball_Targ
Curve walk	¬Ha_Ball, ¬Ball_Dir, ¬Ball_Targ	Ball_Dir, ¬Ball_Targ (Cannot determine state of Ha_Ball)
Turn in place	¬Ball_Dir, ¬Ball_Targ	Ball_Dir, ¬Ball_Targ
Kick	Ha_Ball, Ball_Dir, ¬Ball_Targ	¬Ha_Ball, Ball_Dir, Ball_Targ

The robot Curve Walk has restrictions. It cannot perform curves with all radius, only discrete and established curves, due to the characteristics of the robot's joints. Consequently, we cannot determine the effect of this action with respect of having or not the ball. To suppress this, a tree structure was used. Additionally, the number of possible curves needed to be decreased because of the limited capacity robot's processing power, being 3 curves the optimal number found.

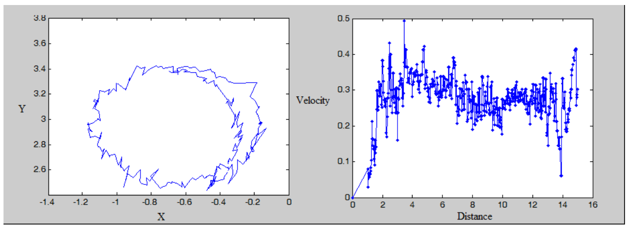
Therefore, the trajectory and velocity of the used curves are not perfect, due to the learning process of the robot. In Figure 4, the variation in the performed path can be observed. Finally, after modeling the problem the prototype was developed and will be explained in detail in the next section.



(a) Radius of 1.5



(b) Radius of 1



(c) Radius of 0.5

Figure 4. Left) Path trajectory for different Curves walkings. Right) Velocity for each radius of curve walking

B. *Prototype Developement*

The next step in our approach is to implement the prototype using the model described before. In order to fully achieve a functional system, more than just the planner itself is needed and the presentation of the complete architecture is the goal of this section.

In the broad view, the prototype will be a module inside the robot's agent already built, i.e., the humanoid robot is controlled by a software agent, counting on different modules (or behaviors), to guide the robot's action. This feature makes the platform scalable, so new modules can easily be added. In addition, to code the prototype C++ language was used.

Focusing in the system, the environment, as told before, is extremely dynamic and stochastic, because of that we can not cannot plan with some world state and then execute it without verifying again if the world state changed. We want a plan for the next 10 seconds, but how to solve it considering the environment characteristics? By using an architecture, which will control the current plan execution and analyze if the planner's goal was achieved, and even predict if it can be achieved in the current conditions. However, to plan is a very time-consuming task. Because of planning complexity, and in order to have the robot operating in real-time, the robot has to avoid to plan frequently.

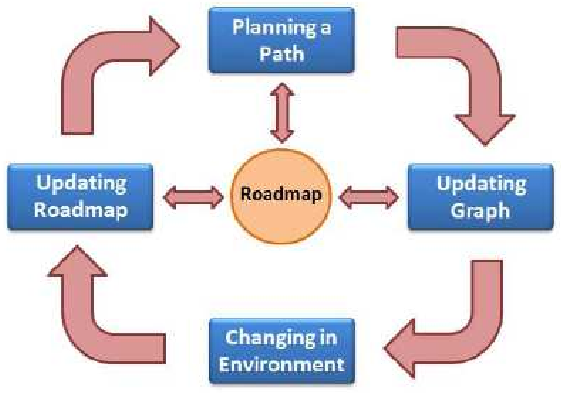


Figure 5. Process sequence in encountering a change in the environment

Figure 5 describes the process sequence in encountering a change in the environment; bein its dynamic explained as:

- The environment changes, so the roadmap (or strategy) must be updated to plan a path or not. After that the path planning block, by checking the updated roadmap, is going to create a new path graph. Further, the graph is updated and put in the roadmap to be checked again after the environment changing. The previous plan and the time of creating the plan is going to be in the roadmap.

Our general approach is presented in Figure 6. The sensor signals enter in the Monitor block, which decides if the plan must continue or not based on the world conditions and information of the current plan. Then, the modified world state has to go to the Planner, also details of the plan are sent to the Monitor for future controlling task described above. The planner's output is a sequence of actions for the robot, in our case they are represented by a set of speed and angle information.

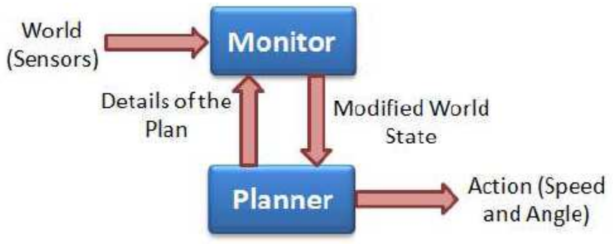


Figure 6. General approach diagram

Concerning the changing in the environment, we define three situations that determine that the plan has to be rebuilt: the ball position changed more than a threshold, the difference between robot's predicted position and robot's current position is more than a threshold, and if after 10 seconds the robot does not reach the ball. Thus, define thresholds for ball and robot's relative positions is a hard task, demanding high numbers of experiences.

Inside the Planner, iterative deepening search is used to explore the tree. The tree expansion is shown in Figure 7, for each node the n possible actions are applied (preconditions must be considered), generating at most n prediction conditions. The graph growth is executed until the goal has been found, with maximum of 10 cycles (i.e. 10 seconds).

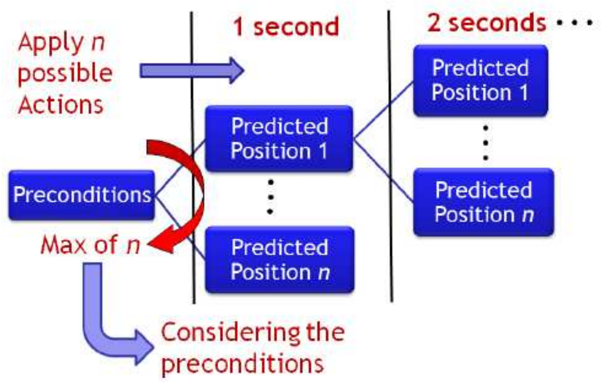


Figure 7. Manner to explore the tree

For sure, our goal state will be one of the states present in the tree. Therefore, combining the exploring the tree (shown before), with the sequence of time quantas, the first time the goal state in the tree is reached, we obtain the optimal and fastest solution.

We faced a time complexity constraint, presented in the modeling section. Thus, the robot plans for the next 10 seconds, each action is executed in 1 second. So, using the equation $O(b^d)$ the d is 10 (extension of the plan) and b is 6 (number of possible actions). In respect of that, we can increase time efficiency of the algorithm in 3 ways: increasing execution time for the actions, reducing possible actions (using heuristics), discovering more preconditions and constraints.

This section aims, first, to prove the prototype functionality and, then evaluate the performance of our approach in some established situations. Several illustrative examples, showing how the system behaved in simulated environment, are presented here. We presented a tree-based approach integrated with planning paradigms for exploring and finding optimum path to achieve to ball in order to kick the ball to defined target.

The Planner tries to plan different skills with different characteristics in order to control the path trajectory of the humanoid robots. Further, two experiences were realized to see how the prototype behaves when it is coupled together with others modules of the agent. The same scenario was used and just the robot's position was varied in order to confirm the plan construction for different situations. The simulator Rcssserver3D [10] was used to test the scenario described above.

In this scenario, the robot is initialized in a position r in Cartesian coordinates where its center is the center of the field.

It plans to move the ball (position *b*) to the goal target where its global position in the field is *t*. For the first experience, the values were *r* = (−1.7; −1.11), *b* = (0; 0), and *t* = (7; 0). The second experience is similar to the first one, however the value of *r* is (−3.09; 0.37).

Figure 8, illustrates the expanded tree used to explore and search aiming to find the optimal plan and, also, shows the resultant plan as a set of positions extracted from the tree. The route of the tree is the Cartesian position of the player in the field. Therefore, Figure 9 shows the robot's position trajectory while it utilizes the produced path plan to reach the ball. As plan's results, we can see the ball's positions in the field when it was kicked by the robot being its trajectory also shown.

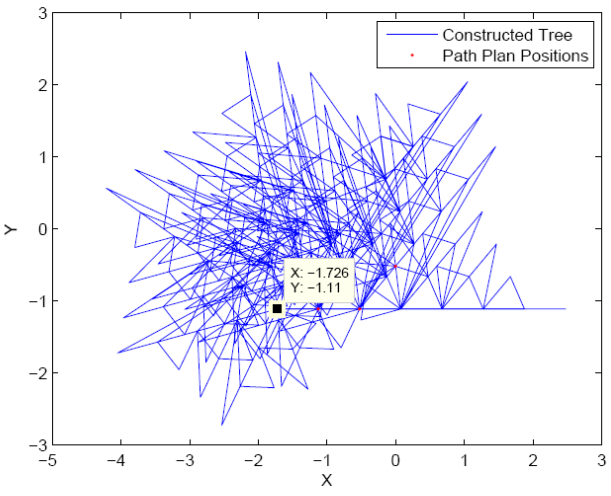


Figure 8. Tree of plans and chosen plan

In Table 2, the details of the path plan can be found. The plan starts from the robot's position and ends having the ball, which satisfies the precondition for the kick. Also, Figure 9 illustrates the robot's trajectory while it uses the path.

TABLE II. SEQUENCE OF POSITION FOR THE DESIGNED PLAN

Action Order	Position X	Position Y
1	−1.72624	−1.11048
2	−1.12624	−1.11048
3	−0.526237	−1.11048
4	−0.526237	−1.11048
5	−0.526237	−1.11048
6	−0.00301497	−0.515413

We observed certain advantages as: the robot using the prototype could accomplish the goal, i.e. walk to the ball in the shortest time possible (optimal path), then kick the ball in the target direction. Nonetheless, as observed in Figure 8, the search space is overly explored resulting in a loss of processing time (e.g. the robot explore and search also in some direction opposite the ball).

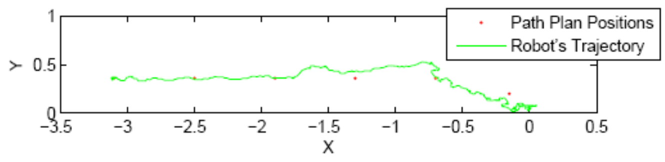


Figure 9. Robot's and ball's trajectories (situation 2)

Finally, comparing Figure 8 and 9, we can see that the robot's speed interferes in the path plan execution, i.e., when the robot walks at high speeds it tends to leave the projected trajectory. This result that in the end of the plan the robot cannot achieves to the desired point and, consequently, monitor has to replan.

V. CONCLUSIONS

Planning aims to find a sequence of actions to reach to a final goal state, being its techniques extremely used for motion planning. A complex environment to deal with is the soccer game. It is even more complex if the scenario includes humanoid robots, which are being increasingly applied in this environment. Also, for humanoid robots, the planning techniques are not developed enough.

A new approach was presented to correlate path planning for humanoid robot in the soccer game environment. Further, the situation of a robot trying to reach the ball and kick it to the target, was modeled using preconditions and actions concepts which results in a realist, but simple, model of the problem. Furthermore, we implemented the prototype (modeled as a iterative deepening search), that could achieve the optimal solution. Experimental results validate the approach, in path planning issue, for humanoid soccer robots, to perform their tasks in the best manner.

There is a sequence of works that could follow the present work, we point some of them. First, use learning methods to discover unknown parameters, such as optimize the planner's time window. Also, add more heuristics and analyze more preconditions to reduce possible actions, i.e., we need to render the model in more detail to search new relations between preconditions and actions. To insert capabilities for obstacle avoidance is a simple next step because the system already has access to the robot position, so it just needs the obstacle positions and filter the plans that originate collisions.

ACKNOWLEDGEMENTS

The first author is supported by FCT under grant SFRH/BD/66597/2009. This work has been partially funded by FCT Project ACORD - Adaptive Coordination of Robotic Teams (PTDC/EIA/70695/2006).

REFERENCES

[1] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, H. Inoue, Online footstep planning for humanoid robots. In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA03. (2003) 932–937

[2] O. Lorch, A. Albert, J. Denk, M. Gerecke, R. Cupec, J. Seara, W. Gerth, G. Schmidt, Experiments in vision-guided biped walking. Vol. 3. (2002) 2484 – 2490

[3] J. Gutmann, M. Fukuchi, M. Fujita, Real-time path planning for humanoid robot navigation. In Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI05. (2005) 1232–1237

[4] N. Shafii, L. Paulo, N. Lau, Biped walking using coronal and sagittal movements based on truncated fourier series. In Proc. of the RoboCup-2010: Robot Soccer World Cup XIII. (2010)

[5] N. Shafii, A. Khorsandian, A. Abdolmaleki, B. Jozi: An optimized gait generator based on fourier series towards fast and robust biped locomotion involving arms swing. (aug. 2009) pp. 2018 –2023

[6] Robotics, A.: Nao webpage (2010) Available in http://www.aldebaranrobotics.com/en/naoeducation, accessed in September 14.

[7] A.J.S.B. Tay: Walking nao omnidirectional bipedal locomotion (August 2009) Master Thesis, University of New South Walse, School of Computer Science and Engineering.

[8] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach. Pearson Education (2003)

[9] S. Kambhampati, E. Parker, E. Lambrecht, Understanding and extending graphplan. In Proc. 4th European Conference on Planning. (1997) 260–272

[10] SourceForge: The robocup soccer simulator (2010) Available in http://www.aldebaranrobotics.com/en/naoeducation, accessed in September

Using Quadric-Representing Neurons (QRENs) for Real-Time Learning of an Implicit Body Model

Manfred Hild, Matthias Kubisch, and Sebastian Höfer
Neurorobotics Research Laboratory
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin, Germany
{hild|kubisch|hoefer}@informatik.hu-berlin.de

***Abstract*—Controlling the body requires self-explorative behavior as well as the ability to build a model of the body. This model does not need to explicitly encode body shape, joint positions, and so on, but can just as well be built up implicitly. We introduce Quadric-Representing Neurons (QRENs) and show why they are very well-suited to model the plurality of body morphologies. QRENs can be either learned in a batch manner or on-the-fly in real-time. They possess the important property to extrapolate behavioral manifolds from a reduced and localized sensorimotor data set. QRENs can be used to elegantly control a robot's body in a straightforward way. We comment on how QRENs have the potential to allow for modular and hierarchical learning strategies.**

I. Introduction

Learning how to control the body within a given environment is a fundamental perceptual task. It requires self-explorative behavior and at the same time the ability to build a model of the body – be it a human being, animal, or robot. In biological systems, information of body posture is available in real-time by afferent proprioceptive sensory signals, but there is no such sensory signal which is directly informative about the body's size and shape. Although the need for a stored body model has been recognized for quite some time, only recently techniques have been introduced to systematically isolate and measure this model for one limb: in [1] the authors produced maps of the mental representation of people's hand size and shape.

Clearly, body models do not need to explicitly encode body shape, joint positions, lengths and mass distributions of the limbs. This would only be required if a robot is to be driven using inverse kinematics, like with industrial robots, where this is still the preferred method. A large amount of literature is dealing with optimal control of smooth motion trajectories, whilst circumnavigating singularities of the inverse model; for an overview see [2]. Body models can just as well be built up implicitly. One example how information about the full body size of a segmented artificial organism can emerge within each body segment using a local homeostatic learning rule can be found in [3].

In the paper at hand we introduce Quadric-Representing Neurons (QRENs) and show why they are very well-suited to model the plurality of body morphologies. The remaining sections are organized as follows. We first define quadrics, describe what they are able to represent geometrically, and introduce QRENs by using quadrics as kernel functions. Then, we describe the different body morphologies under investigation and give examples on how the body morphologies are mapped onto the QRENs. We discuss learning with small data sets, show how a robot can be controlled by using the QRENs, and finally comment on the QRENs' potential to allow for modular and hierarchical learning strategies.

II. Quadrics and QRENs

A quadric Q is a surface defined by an algebraic equation of degree two. Formally, we have

$$Q = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x} + c = 0\}, \qquad (1)$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetric, $\mathbf{b} \in \mathbb{R}^n$, and $c \in \mathbb{R}$. The vector notation is neat and advantageous if we need the eigenvalues and eigenvectors of $\mathbf{A}$. This is the case if we want to normalize the quadric, i.e., successively get rid of the mixed terms of order two (off-diagonal elements of $\mathbf{A}$ are all zero) and, if possible, also the linear terms ($\mathbf{b} = 0$) and the constant ($c = 0$). Using the normalized standard form, quadrics can be categorized, and for $n = 3$ also be visualized. Some examples are shown in figure 1. A parabolic cylinder, a hyperboloid of one sheet, and two parallel planes are respectively defined by

$$\begin{aligned} x_1^2 - x_2 &= 0, \\ x_1^2 + x_2^2 - x_3^2 - 1 &= 0, \\ x_1^2 - 1 &= 0. \end{aligned}$$

Quadrics play an important role in algebraic geometry. We can link quadrics to robot morphologies which are situated in an environment by identifying the variables with sensor values. This will briefly be addressed in the following section.

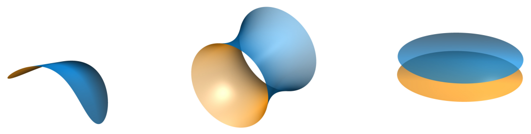


Figure 1. Three examples of different quadrics in $\mathbb{R}^3$, namely a parabolic cylinder, a hyperboloid of one sheet, and two parallel planes (from left to right).

A. Why Second Order is Sufficient

For a large range of robots quadrics are able to describe invariants, i.e., when choosing the appropriate coefficients, the quadric stays zero for a specific subset of the robot's configurations. In other words, quadrics exist, which are invariant under some type of the robot's behavior. This is inter alia the case for robot arms with a series of revolute joints [4].

Why is it sufficient to use a second order polynomial to get a behavior-invariant constant expression? This depends on the type of sensor values used. Let us assume we have a robot arm with two revolute joints in series and arbitrary angular ranges and segment lengths of the arm. We are interested in all configurations where the robot touches a plane. If we use angular sensor values, then we have the trivial case, that the sum of the angles remains constant. If we use cartesian coordinates (e.g., if the sensor values are derived from an image of the scene), then we have

$$x_1 = r\cos\varphi,\ x_2 = r\sin\varphi, \tag{2}$$

for a single joint. Obviously, we get a constant expression by squaring and summing. If we use a completely different type of sensor value, namely acceleration forces of sensors mounted on the robot arm, then again the squared sum of all values will be constant and represent the static gravitational force. This is of course only exactly true for moderately slow motions of angular joints, but approximately still holds otherwise. All in all, there is good reason why quadrics are sufficiently accurate to describe behavioral relationships between a robot's body and the environment.

B. Using Quadrics as Kernel Functions

The main idea of a QREN is to use quadrics as kernel functions and let the QREN indicate the presence of a specific behavioral mode of a robot within a given environmental context.

Since we are not interested in normalizing and categorizing the quadric but want to learn the quadric in real-time, it is advantageous to switch from matrix notation to the following notation:

$$Q = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{w}^T \mathbf{f_n}(\mathbf{x}) = 0\}, \tag{3}$$

where $\mathbf{f_n} : \mathbb{R}^n \to \mathbb{R}^m$ is the expansion of $\mathbf{x}$ including all quadratic terms and the constant 1, $\mathbf{w} \in \mathbb{R}^m$ is a weight vector, and $m = \frac{n(n+3)}{2} + 1$.

We are now ready to define the output of a QREN as follows:

$$q_{\mathbf{w}}(\mathbf{x}) = \mathrm{e}^{-\left(\mathbf{w}^T \mathbf{f_n}(\mathbf{x})\right)^2}. \tag{4}$$

Clearly, $q_{\mathbf{w}}(\mathbf{x})$ acts as an indicator neuron, as we have

$$q_{\mathbf{w}}(\mathbf{x}) = 1 \iff \mathbf{x} \in Q. \tag{5}$$

The more $\mathbf{x}$ is distant from Q, the more $q_{\mathbf{w}}$ is close to zero. Please note, that $\mathbf{w}$ is only defined uniquely up to a multiplicative factor. We therefore have to normalize $\mathbf{w}$ in some way, if we want to compare the output signals of different QRENs to find out which quadric $\mathbf{x}$ is closer to. A straightforward approach would be to demand $\|\mathbf{x}\| = 1$, but we already succeeded with the even easier trick to always have the quadric's constant term equal one (we get more explicit below).

In other fields, there exists quite some literature on how to fit the parameters of a quadric to a given data set, mostly for data stemming from stereo images or laser range measurements, e.g., see [5], [6], or, very recently, [7]. But also more general approaches have already been addressed [8]. There exist comparative surveys on the quality of different methods, as in [9] and [10], as well as reports on stability [11]. Of specific interest for our QRENs is the comparative survey of neural learning rules in [12], most standard methods of which can be applied here. Using a competitive neural network with a simple delta rule will already work well when learning several QRENs simultaneously. In this paper, we will not go into detail concerning the selection of an appropriate learning rule, but focus on the peculiarities of robot learning, namely, that sensory data from real behaviors will mostly only cover a small part of a full quadric. Therefore, we are interested in the QRENs' extrapolating abilities.

For what follows we use $n = 3$, so the parameter vector to be learned is of dimension $m = 10$. We have

$$\begin{aligned}
\mathbf{x} &= \left(x_1\ x_2\ x_3\right)^T, \\
\mathbf{f_3}(\mathbf{x}) &= \left(x_1^2\ x_2^2\ x_3^2\ x_1x_2\ x_2x_3\ x_1x_3\ x_1\ x_2\ x_3\ 1\right)^T, \\
\mathbf{w} &= \left(w_1\ w_2\ \ldots\ w_{10}\right)^T,
\end{aligned}$$

where we demand $w_{10} = 1$ during the least squares fitting of $\mathbf{w}$. Coefficients shown in the section on experiments will always be in the order $w_1, w_2, \ldots, w_9$. Since $w_{10} = 1$ by construction, this coefficient will be omitted.

III. Different Body Morphologies

For the investigation of the QRENs' properties, the two robots SEMNI and Myon have been used, which significantly differ in size, mass, actuation, and overall morphology. They will briefly be described within the following subsections.

Despite their different properties, they are equipped with the same data interface, so data acquisition and experimental setting could be identical for both hardware platforms. Thus, it can be excluded that resulting body models are distorted systematically due to platform-dependent data quality (e.g. noise, resolution, sampling rate). Sensorimotor loops have always been guaranteed to run tightly at a rate of 100 Hz.

A. The Self-Exploring Robot SEMNI

The acronym SEMNI stands for *Self-Exploring Multi-Neural Individual* – and that is exactly the purpose the robot has been built for. It possesses only two degrees of freedom: one revolute joint at the hip, and another one at the knee, as can be seen in figure 2.

Proprioceptive sensors continuously measure the current and temperature of each actuator, the angular positions of both joints and the acceleration forces within the robot's midsagittal plane, relative to the center of the printed circuit board in the head. Thus, there are two motor values and eight sensor values per 10 ms time slot.

The actuators can be controlled in various ways. For the experiments reported here, we choose a constant velocity paradigm to always keep the robot in motion at a moderate speed. This way, we can omit the robot's velocity vector and restrict ourselves to the analysis of the manifold

$$M_r = \{(\varphi_h, \varphi_k, \varphi_b) \in \mathbb{R}^3 \mid \chi_r(\varphi_h, \varphi_k, \varphi_b) = 1\}, \quad (6)$$

where φ_h, φ_k and φ_b are the angular positions of the hip joint, knee joint and the robot's body, the latter being measured relative to the horizontal ground surface. Due to the moderately slow motion, the horizontal and vertical acceleration forces F_h and F_v are dominated by the earth's gravitational force, so the body position can be calculated as $\varphi_b = \arctan(F_v/F_h)$.

The characteristic function χ_r equals one, iff its argument represents a body posture which the robot can reach and hold on its own. Thus, the manifold M_r is defined by all reachable body positions on a flat ground, without using dynamic motions and excluding transient motions when the robot falls over.

Obviously, M_r completely depends on the robot's body shape, moving abilities, and environment. It can thus be regarded as an implicit body model of the robot, situated in a fixed environment. Whenever the robot is at rest, its body posture and position relative to the ground correspond to a point $\mathbf{p} \in M_r$. So, at low velocities, we stay within M_r, whereas dynamic motions temporarily leave M_r, be they induced by the robot's motors or by tumbling accidentally.

Having noted this, we are now able to inspect M_r for SEMNI on a flat ground. Please refer to figure 3 for the following explanation. The manifold has been cut into four parts. We start with the bottom left image, which corresponds to the situations where the robot is lying on its front side (i.e., 90 degrees counter-clockwise from the position shown in figure 2). The xy-position corresponds to the posture of the leg, with the horizontal position representing the joint angle of the hip, and the vertical position that of the knee.

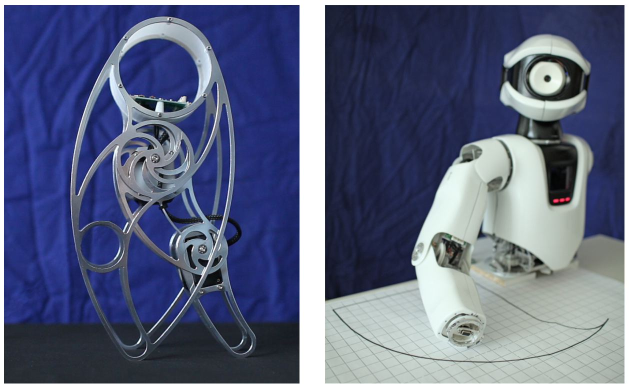


Figure 2. Left: The 30 cm tall robot SEMNI standing in an upright position. Proprioceptive sensors are located within the actuators (hip, knee) and on the printed circuit board in the head. The robot is facing to the left, with the leg standing on the back side. Right: Head, torso, and right arm of the modular humanoid research robot *Myon*. The black solid line indicates the area which can be reached by the arm. The hand is not attached in this experiment.

The border of the missing corner (top right white part of the image) represents all leg postures where the foot touches the back part of the head. For each xy-position (i.e., leg posture) the color encodes the body's angle relative to the ground. The darker the blue, the more the head is near the ground, whereas the darker the red, the more the head is in the air. The pale greyish regions indicate leg postures where the robot's body is lying on the ground horizontally.

As can be seen, the robot is tilted a bit in one or the other direction, depending on where the barycenter of the leg is, relative to the hip joint. Now, the robot is starting to sit up, if the leg is following a specific trajectory, which is shown in the top left image. Coming from the bottom left and continuing to the top right (in the image), the robot is quickly raising its head and then falling over – either onto its back or back to the front again. This can be seen in the two bottom images by the isolated blue dots which correspond to impacts of the robot's body. Obviously, the manifold has borders which correspond either to the joints' end positions, or to unstable body positions. In the former case we are just stuck, whereas in the latter case, we fall off the manifold and back onto it to another place.

The two images on the right side are analogous to the left ones, but describe the robot starting from lying on its back. The darkest blue positions correspond to the robot performing a headstand. Due to the leg length and slow motion, the robot is not able to do a backward somersault on the ground. When speeding up the motion, this is indeed possible.

Summing up, the manifold M_r, which describes SEMNI's capabilities during moderately slow motions, is already complex enough to be of interest for building up an implicit body model. As we will see later, only few QRENs are needed to accurately capture M_r.

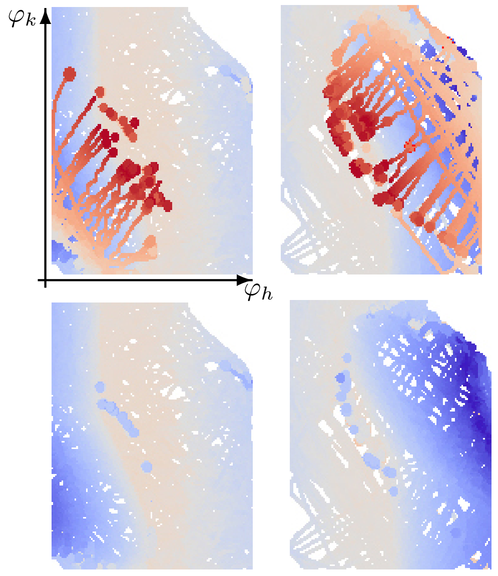


Figure 3. A sampled version of the manifold M_r of the robot SEMNI, situated on a flat ground. It indicates the body posture (xy-coordinates) and body position relative to the ground (color-coded). See text for an explanation.

B. The Humanoid Robot Myon

Since the shape and configuration of SEMNI is rather uncommon, we choose a more standard scenario for additional tests of the QRENs. The humanoid robot *Myon* is a modular research robot, the body parts of which can be detached and reattached during run-time, since they all possess their own processing power and energy supply. Figure 2 shows Myon composed of three body parts, namely the torso, the head, and the right arm. This is the experimental setting we used to record sensory data when the arm was moving, while the end of the arm was touching the table.

IV. EXPERIMENTS AND RESULTS

In order to put the QRENs to test, we first recorded sensory data of the robot SEMNI while it was touching the ground with both feet. This is a somehow artificial situation, since only part of the poses are stable and we had to hold the robot still in place to get the rest of the data. The result can be seen in figure 4. This is obviously an elliptic hyperboloid of the kind shown in middle of figure 1.

We then attached a QREN to the behavioral primitive *standing-on-the-ground-with-both-feet* by fitting the corresponding weights using a least squares approach. The weights are shown by the crosses in figure 5. To check the quality of the fit, we calculated the QREN's kernel function for the original raw sensory data, sorted the results in descending order, and plotted the result in figure 6. As can be seen, the QREN quite nicely fits the original data – the error remains small, even without having filtered the raw sensor values beforehand.

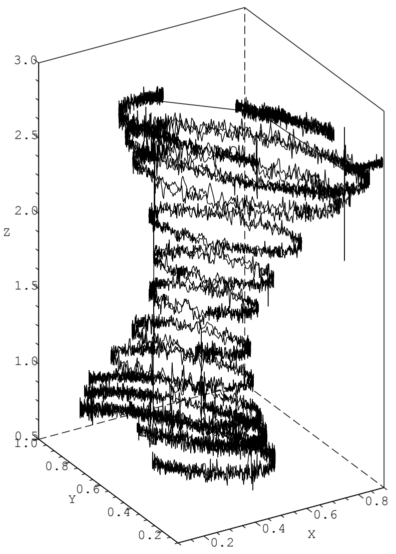


Figure 4. Sensory data of the robot SEMNI which has been recorded while the robot was touching the ground with both feet, like shown in figure 2. The xy-plane encodes the angular position of the joints in the same way as in figure 3. The z-axis shows the body's angle relative to the ground, the standing position being at $z = \pi/2$.

A. Using Massively Reduced Data Sets

The weight vector contains nine free parameters, so only nine out of the over 30.000 data values are sufficient to fully specify the QREN. In order to find out how much the weight vector varies depending on the data sample, we randomly picked nine data values and calculated the weights. This has been repeated 50 times and plotted together, as shown in figure 7. Obviously, there is not much variance, since the sampled values are far apart by chance. The subset-depending weight vector variance is of more behavioral relevance, if the subset is not spread over the whole raw values, but localized. This is equivalent to sampling the stream of sensor values over a short time period, where the pose of the robot does not change too much. We used the lower corner of the hyperbolic ellipsoid, as the robot passes through this part during exploration (see figure 3). The used subset is shown in figure 8 and the result can be found in figure 5 by comparing the crosses (full data set) with the solid dots (localized subset of the data). The differences are almost unnoticeable, which illustrates the QRENs' excellent extrapolation characteristics. This in turn allows for feeding the very first quadric estimates back to the control of exploration – the QRENs can very soon help to steer the direction of exploration.

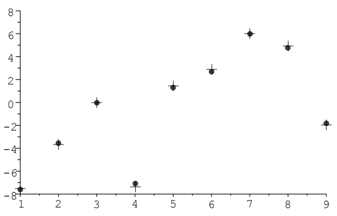


Figure 5. Crosses: weight vector of the elliptic hyperboloid which optimally fits the robot's sensory data. Solid dots: weight vector found by the least squares approach when using only a small subset of the data.

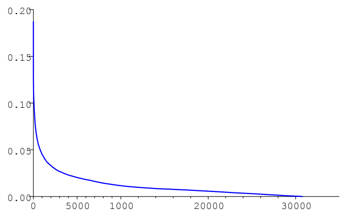


Figure 6. Values of the QREN's kernel function for all original sensor values, sorted in descending order. The error remains small, even without having filtered the data.

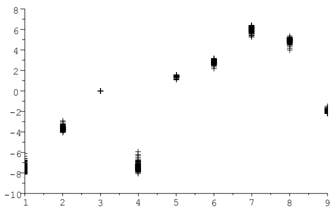


Figure 7. The weight vector does not change very much when only nine random samples from the whole data set are drawn for parameter calculation. The process has been repeated 50 times.

B. *Fitting the Quadric of a Moving Arm*

Next, we used the angular data which we recorded using the arm of the humanoid robot Myon, as shown in figure 2. The arm was moving in all directions, while the end of the arm was touching the table all the time.

The raw sensory data is shown in figure 9 and reveals part of an ellipsoid. The corresponding weight vector can be found in figure 10. Although the raw data forms less than one eights of a full ellipsoid, again, the quadric which has been fitted by the QREN is as close to the sensory data as in the former case of the hyperboloid. Due to the limited space, we do not explicitly show the model and the quality of fit here. Interestingly, we also got an ellipsoid when used the acceleration sensors instead of the angular ones.

C. *Quadric-Based Movements*

Since QRENs represent quadrics that encode an implicit body model of a robot given in a specific environment, they are of behavioral relevance. As it turns out, the weight vector of a QREN can be used in a straightforward manner to purposefully control a robot's motion.

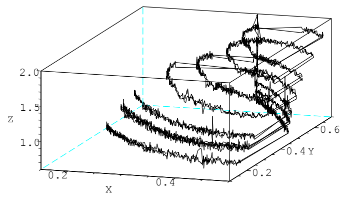


Figure 8. Localized subset of the full sensory data which has been used to test the QRENs' extrapolation abilities.

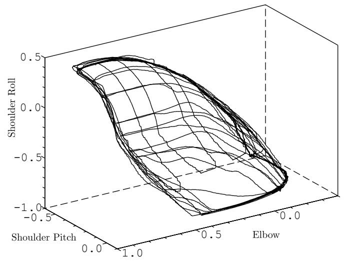


Figure 9. Raw angular sensor data of the three joints of Myon's right arm, as shown in figure 2. Obviously, the data can be modeled by an ellipsoid.

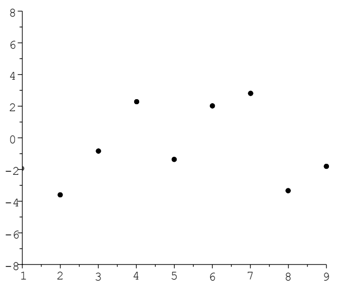


Figure 10. Weight vector of the ellipsoid which best fits the sensor data of Myon's moving arm.

The following example will illustrate this. Say we want the robot SEMNI enable to sit-up from arbitrary starting postures. There are many target positions for the robot to stand upright and they all lie on the hyperboloid shown in figure 4. To be more precise, they satisfy the two conditions:

$$\mathbf{w}^T \mathbf{f_3}\left((\varphi_h\ \varphi_k\ \varphi_b)^T\right) = 0, \tag{7}$$

$$\varphi_b = \pi/2. \tag{8}$$

Using the well-known gradient descent, we control the voltage of the hip actuator (U_h) and of the knee actuator (U_k) as follows:

$$U_h = -\mu \mathbf{w}^T \mathbf{f_3}(\hat{\mathbf{x}}) \frac{d\mathbf{w}^T \mathbf{f_3}}{d\varphi_h}(\hat{\mathbf{x}}), \tag{9}$$

$$U_k = -\mu \mathbf{w}^T \mathbf{f_3}(\hat{\mathbf{x}}) \frac{d\mathbf{w}^T \mathbf{f_3}}{d\varphi_k}(\hat{\mathbf{x}}), \tag{10}$$

where μ is a fixed motor constant and

$$\hat{\mathbf{x}} = (\varphi_h\ \varphi_k\ \frac{\pi}{2})^T \tag{11}$$

is the current posture, but with φ_b clamped to the desired target value. Of course this approach can also be used to define arbitrary targets, e.g., stretching the leg ($\varphi_k = 0$).

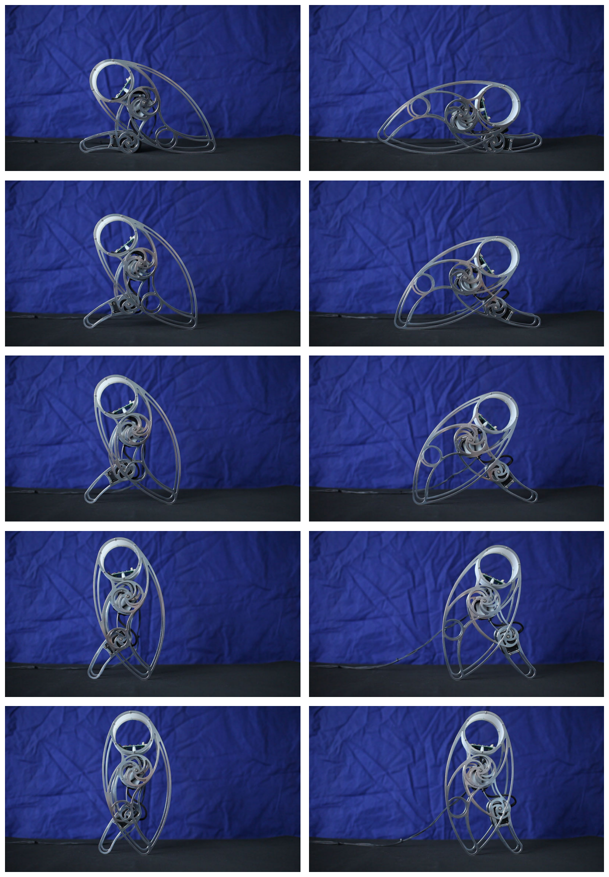


Figure 11. Series of snapshots (top to down) illustrating two different sit-up motions of the robot SEMNI. Left column: Starting with the robot's front side facing the ground. Right column: Starting with the robot's front side looking away from the ground.

For the experimental setup with the robot Myon this would include lifting the arm to arbitrary heights above the table. Of course sensory signals from the vision system need to be recruited during the learning process in first place, but after that the robot is able to reach a target position blindly.

Figure 11 shows the results after implementation of the aforementioned motion control loop. The robot successfully sits up from different starting postures and, as a consequence, also stops in an upright position with different leg postures. The motion trajectories are highly efficient in the sense that as less movements as possible are being executed.

D. *Speeding Up the Learning Process*

As already mentioned above, a standard competitive neural network, where only the winning QREN is updated by the delta rule, will learn the manifold of a robot, like the one shown in figure 3. Using the sensory information in a smart way, one can considerably speed up the learning process. Figure 12 shows the impacts of SEMNI's body. They are derived from the acceleration data. Since we used a moderately slow motion for the exploration of the robot's behavioral manifold, those impacts can only be due to jumps off and back onto the manifold (also see the isolated dots in figure 3). This information can be used to introduce new QRENs in early stages of the learning process.

Another strategy uses the current sensors of each actuator: Whenever a stall current situation is detected, a specific QREN is learned. This QREN will soon anticipate self-harm of the robot and most likely be representing a quadric with two parallel planes, as shown in figure 1, right. Finally, referring to the QREN's excellent extrapolation characteristics, and using the quadric-based motion control we introduced, it is a good idea to explore the manifold QREN by QREN. This reduces the amount of impacts which could potentially harm the robot, and, at the same time, improves the accuracy of the QREN which is currently active.

V. Conclusion and Outlook

We have formally introduced QRENs and given experimental evidence that they are able to implicitly model different body morphologies and motion capabilities. QRENs can be learned using standard methods, but we also outlined how sensory information can be used to speed up the learning process. For a single QREN first results concerning the dependence on the data set have been presented. They show the QREN's robustness and extrapolating capabilities.

There are two very promising directions, we would finally like to comment on. First, when modeling behavioral manifold by multiple quadrics the latter will intersect. So-called QSICs (Quadric Surfaces Intersection Curves), see [13] for a classification, show up. Those QSICs are behaviorally of special interest, because a robot who will spend most of its time on QSICs will have better options of quickly doing one or the other.

Second, the scalar output of a quadric can serve as a virtual sensor. We intend to use this approach for the modular humanoid robot Myon. If a QREN is able to encode the behavioral condition, that a leg or arm stands upright (like shown with the robot SEMNI), the scalar output of another QREN can encode the height of the limb relative to the ground (like with the arm of Myon). This way, QRENs inside the torso can make use of this virtual sensor information to model *crawling*, *walking*, and the like.

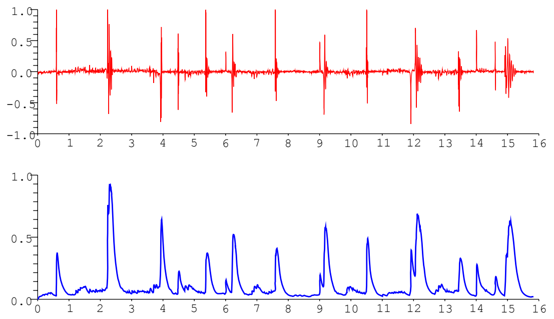


Figure 12. Sensory information of the type shown here can be used to speed up the learning process. Top: Squared sum of the three perpendicular acceleration forces minus the squared static value of the earth's gravitation. Bottom: Smoothed and normalized absolute value of the above curve.

References

[1] M. R. Longo and P. Haggard, "An implicit body representation underlying human position sense," *Proceedings of the National Academy of Sciences of the United States of America*, 2010.

[2] J. Lenarcic and P. Wenger, Eds., *Advances in Robot Kinematics: Analysis and Design*. Springer, Netherlands, 2008.

[3] M. Hild and F. Pasemann, "Self-adjusting ring modules (sarms) for flexible gait pattern generation," in *IJCAI*, 2007, pp. 848–852.

[4] J. M. Selig, *Geometric Fundamentals of Robotics*. Springer, New York, 2005.

[5] S. Lee and H. Hahn, "An optimal sensing strategy for recognition and localization of 3D natural quadric objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, October 1991.

[6] G. Cross and A. Zisserman, "Quadric reconstruction from dual-space geometry," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, 1998, p. 25.

[7] M. R. Rahayem and J. A. P. Kjellander, "Quadric segmentation and fitting of data captured by a laser profile scanner mounted on an industrial robot," *The International Journal of Advanced Manufacturing Technology*, 2010.

[8] G. J. Agin, "Fitting ellipses and general second-order curves," Robotics Institute, Carnegie-Mellon University, Tech. Rep. CMU-RI-TR-81-5, 1981.

[9] M. Dai and T. S. Newman, "Hyperbolic and parabolic quadric surface fitting algorithms – comparison between the least squares approach and the parameter optimization approach," Computer Science Department, University of Alabama, Huntsville, Tech. Rep. TR-UAH-CS-1998-02, 1998.

[10] S. Petitjean, "A survey of methods for recovering quadrics in triangle meshes," *ACM Computing Surveys*, vol. 34, no. 2, pp. 211–262, June 2002.

[11] R. Halir and J. Flusser, "Numerically stable direct least squares fitting of ellipses," 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.7559

[12] S. Haykin, *Neural Networks and Learning Machines*. Pearson Prentice Hall, New Jersey, 2009.

[13] C. Tu, W. Wang, B. Mourrain, and J. Wang, "Using signature sequences to classify intersection curves of two quadrics," *Computer Aided Geometric Design*, vol. 26, pp. 317–335, 2009.

Multi-Modal Local Sensing and Communication for Collective Underwater Systems

Serge Kernbach, Tobias Dipper, Donny Sutantyo

Abstract— **This paper is devoted to local sensing and communication for collective underwater systems used in networked and swarm modes. It is demonstrated that a specific combination of modal and sub-modal communication, used simultaneously for robot-robot and robot-object detection, can create a dedicated cooperation between multiple AUVs. These technologies, platforms and experiments are shortly described, and allow us to make a conclusion about useful combinations of different signaling approaches for collective underwater systems.**

I. Introduction

Underwater exploration represents a very important economic, technologic and scientific challenge. This is closely related to Arctic and Antarctic offshore resources, pollution monitoring, general oceanographic data collection and, recently, to underwater actuation [1]. Due to very large underwater areas and high damping properties of water, application of multiple Autonomous Underwater Vehicles (AUVs) in cooperative missions seems very promising [2].

For application of AUVs in networked or swarm mode, there is a number of crucial issues: underwater sensing and communication (S&C), cooperation and mission control, design of AUV platforms, autonomous behavior and several collective aspects of running multiple AUVs. In this work we concentrate on local S&C, and related coordination strategies, being motivated by the following reasons [3].

In several past and running projects devoted to underwater swarms, such as AquaJelly [4], Angels [5], CoCoRo [6], a number of AUV platforms and sensing technologies has been developed. These works indicated two important issues: a successful AUV platform needs a dedicated combination of different S&C technologies, moreover capabilities of underwater cooperation depends on the level of embodiment of on-board S&C systems. In several cases, even a simple multi-modal signal system leads to advanced cooperation [7].

Since swarm approaches rely primarily on local interactions between AUVs [8], the paper is devoted to local S&C systems (unmodulated and modulated IR/blue light, RF and electric field), which can be used for robot-robot/robot-object detection and provide sub-modal information, such as direction and distances, as well as can be used for analog and digital communication. These systems, developed for each of the platforms is described in Sec. III, whereas Sec. II provides general overview over different S&C technologies. In Sec. IV we shortly sketch a few behavioral experiments with these systems and finally in Sec. V conclude about useful combinations of S&C and their embodiment for collective underwater systems.

Institute of Parallel and Distributed Systems, University of Stuttgart, Universitätstr. 38, 70569 Stuttgart, Germany, emails {Serge.Kernbach, Tobias.Dipper, Donny.Sutantyo}@ipvs.uni-stuttgart.de

II. Comparison of different S&C systems

In this section we give a short overview over different state-of-the-art S&C systems in an underwater environment, see e.g. [9]-[10]. Four systems will be compared:

- *Sonar*: Sonic waves travel very well under water and the energy and build-space required for generating and receiving them is very low. This approach is used in so-called acoustic modems [11]. Drawbacks of this approach are, firstly, the relatively low sound travel speed of roughly 1500 m/s (much slower than any other S&C system), and, secondly, multiple reflections causing essential distortions in the signal.
- *Radio*: Electromagnetic waves are a standard communication method in air; its application under water creates several problems [12]. Due to water connectivity, the attenuation of radio waves depends on the used frequency, which in turn results in the size of antenna. High frequencies ($>$ 100 MHz) only need a small antenna (0.1 m) while their range is restricted to 2.5 m. Lower frequencies (100 kHz) have a long range (100 m), but need a large antenna (100 m).
- *Optical*: Using light as a communication channel can provide a compact size of transmitting equipment and acceptable range [10]. Due to the color dependent attenuation of light in water, the communication range varies between a few centimeters in IR spectra and increases to over a meter by using blue or green light.
- *Electric Field*: This is a new communication approach. It bases upon generating and measuring electric fields. The build-size and energy required for this system is very small. Unfortunately the attenuation of electric fields is very high, liming the range of this communication channel to less than 1 m. We will discuss this approach in Sec. IV-B.

Table I shows an overview of the discussed approaches. Since the used AUVs operate in a swarm mode (large number of AUVs, full decentralization, utilization of swarm approaches for coordination, see more in e.g. [13]), in this paper we concentrate on a local S&C approaches. The S&C is defined as local, when the communication range R_c (i.e. communication volume $V_c = 4/3\pi R_c^3$) does not overstep the second-next-neighbors at average swarm density $D_{sw} = N/V_{sw}$, where V_{sw} is the volume occupied by AUVs and

Channel	Attenuation	Antenna Size	Range
Sonar (30 kHz)	0.3 dB/m	0.1 m	300 m
Radio (100 kHz)	1 dB/m	100 m	100 m
Radio (1 MHz)	4 dB/m	10 m	25 m
Radio (100 MHz)	40 dB/m	0.1 m	2.5 m
Optical unmodul. (IR 800 nm)	10 dB/m	0.1 m	0.25 m
Opt. modul. (PCM IR 800 nm)	10 dB/m	0.1 m	0.5 m
Optical modul. (blue 460 nm)	1 dB/m	0.1 m	1.2 m
Electric Field (2.5 kHz)	100 dB/m	0.1 m	1 m

TABLE I
COMPARISON OF DIFFERENT COMMUNICATION CHANNELS

N is their number. Local communication range R_c can be approximated by

$$R_c =^3 \sqrt{\frac{V_{sw}}{N4/3\pi}}, \qquad (1)$$

where for $V_{sw} = 5m^3$ and $N = 20$, R_c is about 0,4m. For the platform size 10-15cm, this results in 3 to 4 times the robot length. Generalizing the AUV size up to 50cm, we assume that R_c^l within 0,5-1,2m are local, whereas R_c^g capable to cover the whole V_{sw}, i.e. 3-4m, are global. In the following sections we consider the developed optical and electric field S&C approaches which are related to R_c^l.

III. LOCAL S&R APPROACHES

A. Multi-modal Optical System

As the first developed approach for combined S&C within R_c^l, we describe a specific bi-modal directional optical system, which underlies cooperative behavior of AquaJelly robots [4]. AquaJelly was a project between Festo AG & Co. KG (coordinator and founder), Effekt-Technik GmbH, and University of Stuttgart intended to create a swarm ($N = 20-30$ robots) of autonomous underwater robots, capable of multi-modal interactions and underwater recharging. Robots have been developed and manufactured within a very short time of 8 months in 2007-2008.

Fig. 1. AquaJelly robots.

Technical requirements define robot-robot, robot-docking-station, and collisions recognition; cooperative collision avoidance; several types of cooperative behavior around docking station, and vertical movement of robots (robots possess only vertical DoF with a balancing mechanism; this allows an inclined vertical movement). Due to visual effects, which are one of the main developmental goals of this platform, it was decided to use unmodulated blue light. Since several communication approaches should remain invisible for human observers and to make the system more stable to different illumination conditions, it was decided to use additionally 36kHz modulated IR light. The platform possesses also very sensitive pressure and temperature sensors, 3D accelerometer and energy sensor (additional RF system was used for a backup communication with the host). The energy part consisted of 4A/h LiPo accumulator with a power management circuitry and Hot-Swap controller for underwater recharging. Due to low energy consumption, the autonomy lies between several hours and with autonomous recharging is theoretically unlimited.

3D omni-directional communication and sensing was one of the main technical requirements. Blue light and IR sensors are used in different ways. Since IR light is more dumped in water, IR channels are very useful for a short range directional communication. Blue light channels are used as unidirectional system, which was mainly used for navigation approaches based on optical pheromone. To enable directional communication, the original platform has 11 IR emitters and receivers with integrated PCM decoder, see Fig. 2.

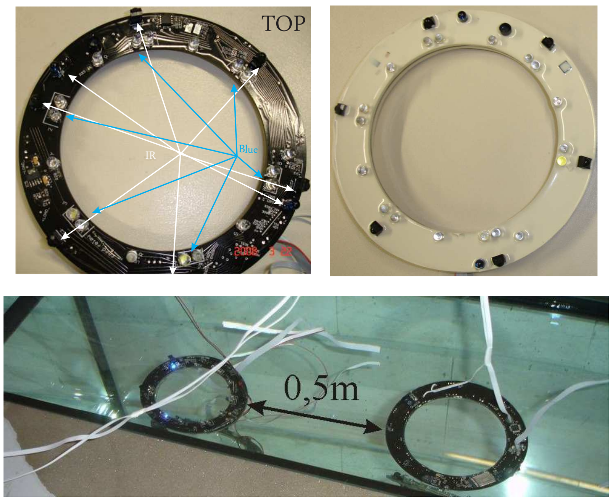


Fig. 2. **(Top-left)** Placement of IR and blue light sensors in the top of the ring; **(Top-righ)** Molded ring in white polyurethane; **(Bottom)** Experimental measurement for IR communication/sensing.

Spatial displacement has an important role, so sidewise IR emitters and receivers are set up each 60 degrees and separated through thick a black-colored PCB on TOP and BOTTOM sides. Three IR sensors are positioned down-side and two up-side. Six blue light LEDs are installed on the top of the platform and through mate cover created almost a homogeneous ”light ball” around the robot. All 17 communication channels are independent from each other through an analog multiplexor. To make the platform waterproof, the ring with all sensors was molded in polyurethane. Blue light system has been used in analog mode, whereas IR used a digital PCM-modulated signal. All sensors are directly connected to I/O pins of the Atmel MCU, which can provide around 8-10mA current at 3V, opening angle of all LEDs is about 10 degrees. Communication range of modulated IR is about 0,5m, see Fig. 2. Due to passive PCM filter, the communication range was fixed on this distance and used in 4kbps communication mode. Average range of the analog blue light system is about 1-1,3m and can be varied by regulating intensity and number of LEDs.

The main idea for using two different optical systems was a split between analog gradient-based interactions between robots (visible for human observer) and digital channels used for robot-objects interactions and for communication, which synchronizes internal states of robots and docking station (invisible for observers). Thus, a combination between analog omni-directional ”long-range” and digital directional ”short-range” optical systems used in different modifications AquaJelly robots allowed a wide range of different sensing and communication approaches, which result in interesting cooperative behavior of these platforms, see Sec. IV.

B. *Modulated and Encoded Blue Light*

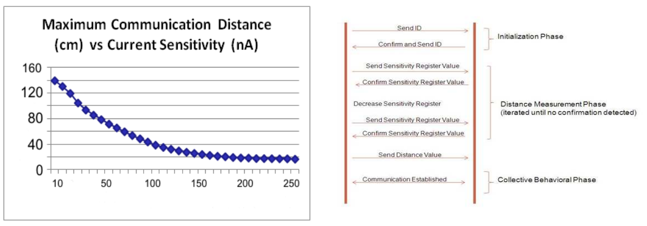
As a further development of the S&C system, described in the previous section, we intend to use only one light system with modulated blue light for both robot-robot/object detection, distance measurement and digital communication. Digital optical communication is widely used due to its high bandwidth. However, the absence of gradient-based optical guide for sensing and localization makes the digital system less suitable for navigation purposes. Therefore specific protocols are required to extract sub-modal information about distances and orientation from the digital channel. The table II shows our underwater measurement results that compare the common modulated IR and blue light communication in many modulation types at the bandwidth of 119kbps.

Modulation	Transducer	Maximum Communication/Sensing
direct	Infra-red	- / -
IrDA	Infra-red	7 cm / 0-5 cm
TV Remote	Infra-red	5 cm / 0-5 cm
QAM	Infra-red	12 cm / 0-5 cm
direct	Blue LED	20 cm / - / -
IrDA	Blue LED	60 cm / 0-5 cm
TV Remote	Blue LED	45 cm / 3-8 cm
QAM	Blue LED	120 cm / 7-12 cm

TABLE II
RANGE OF UNDERWATER OPTICAL COMMUNICATION FOR 119KBPS.

As a digital communication transceiver, the blue light system needs modulator, amplifier, signal conditioner, and protocol encoder/decoder. The one chip solution can be solved by using a CS 8130 IrDA chip from Cirrus Logic. Since the blue light system has a directional S&C, two channels are not sufficient for the swarm robot to communicate in every direction. The half duplex behavior of each channel makes one channel unable to be applied for sensing, because the sensing mechanism requires to transmit and to receive the sensing signal in one time. Therefore, the position of the transmitter and receiver are swapped with neighboring channels for sensing application.

The system must be configured and calibrated for finding the best modulation type for underwater communication and sensing. According to the measurement results, both for communication and sensing, the Quadrature Amplitude Modulation (QAM) seems to be the best modulation for underwater application. Fig. 3(a) shows the relation between


Maximum Communication Distance (cm) vs Current Sensitivity (nA)
160
120
80
40
0
10
50
100
150
200
250
Send ID
Confirm and Send ID
Send Sensitivity Register Value
Confirm Sensitivity Register Value
Decrease Sensitivity Register
Send Sensitivity Register Value
Confirm Sensitivity Register Value
Send Distance Value
Communication Established
Initialization Phase
Distance Measurement Phase (iterated until no confirmation detected)
Collective Behavioral Phase

(a) (b)

Fig. 3. **(a)** Distance measurement with QAM blue light; **(b)** Active Sensing Algorithm.

current sensitivity and communication distance. By using this curve, an active sensing algorithm can be added to the inter-robot communication algorithm by varying the amplification and the sensitivity of the programmable amplifier via software, see Fig. 3(b). The robot can approximate the distance with other robots by gradually decreasing the current sensitivity while communicating each other. The developed inter-robot communication algorithm has three phases. First, when two robots are in the communication range, they begin to establish the communication by sending their IDs to each other. Second, the communicating robots are approximating their distance by gradually decreasing the current sensitivity within the programmable gain amplifier. Therefore, after knowing their own position, behavioral or cooperation phased can be performed. A robot will continuously iterate the first phase if there is no other robots in the communication range. Hence an obstacle might reflect the transmitted signal and the robot would receive back the first phase communication packet that contains its own ID.

C. *Electric Sense*

After experimenting with optical S&C systems, we implemented another approach, which is inspired by weakly electric fish. These animals are capable of producing an electric field which they can use for localisation and communication [5]. Here we try to use this bio-inspired approach for analog communication and navigation in robot swarms.

Electric fields. Electric charges generate electrical fields in their vicinity. Electric fields are vector fields. For a point charge Q the field intensity $\vec{E}$ can be calculated at each point $\vec{r}$ as

$$\vec{E} = \frac{Q}{4\pi\epsilon_0\epsilon_r} \cdot \frac{\vec{r}}{r^3} \tag{2}$$

with the permittivities ϵ_0 (vacuum) and ϵ_r (relativ) [14].

The field vectors of multiple point charges follow the superposition principle. In our robot the electric field is generated by a dipole. The field intensity is proportional to the charge in the electrodes, which themselves are proportional to the applied voltage to the electrodes:

$$Q = C \cdot U \tag{3}$$

with the voltage U and the capacity C.

Communication. The intensity of an electric field can then be detected by measuring the differential potential between two electrodes in the field. This potential is proportional to the electric field intensity, which itself is proportional to the output voltage of the sender. By modulating the output voltage of the sender, information can be transmitted.

Localization. By using multiple pairs of electrodes in the receiver it is possible to calculate the bearing and distance to the sender. This is achieved by utilizing the drop in field intensity with relation to the distance. A sinus wave is impressed on the sender's electrodes which creates an oscillating electrical field. The field intensity depends mainly on the amplitude and frequency of the output voltage and some environmental conditions. The amplitude in the field intensity at a specific point $\vec{r}$ is proportional to the amplitude of the output voltage:

$$u(t) = a_o \cdot \sin(\omega t) \sim E(t) \sim a_i \cdot \sin(\omega t) \cdot \frac{\vec{r}}{r^3} \tag{4}$$

with the amplitude of the output signal a_o and measured input a_i, the frequency ω and the time t.

If sender and receiver are approximately in the same plane and the electrodes have the same orientation (compare Fig. 4 left) (4) can be simplified to:

$$a_o \cdot \sin(\omega t) = a_i \cdot \sin(\omega t) \cdot \frac{F(\omega)}{r^2} \tag{5}$$

with the frequency dependent proportionality factor $F(\omega)$ and the distance r between sender and receiver. Measuring the sinus amplitude (a_1, a_2, a_3, a_4) at four points with a specific geometrical pattern (Fig. 4 right) leads to the following equations:

$$\begin{array}{llll} a_1 = & \frac{A_o}{F(\omega)} \cdot \frac{1}{(r-s\cdot\cos\alpha)^2}, & a_2 = & \frac{A_o}{F(\omega)} \cdot \frac{1}{(r-s\cdot\sin\alpha)^2} \\ a_3 = & \frac{A_o}{F(\omega)} \cdot \frac{1}{(r+s\cdot\cos\alpha)^2}, & a_4 = & \frac{A_o}{F(\omega)} \cdot \frac{1}{(r+s\cdot\sin\alpha)^2} \end{array} \tag{6}$$

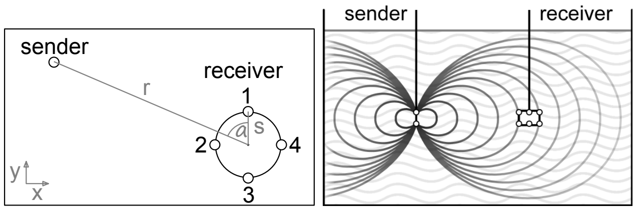


Fig. 4. Position and orientation of sender and receiver electrodes, top-(left) and side-view (right)

In setting up these equations it is assumed that $r >> s$ so that the error in the angle α and distance r between the different sensors is minimal. In (6) the proportional factor and output amplitude can be eliminated, under the condition of $r > s$ leading to:

$$r = s \cdot \cos\alpha \underbrace{\frac{\sqrt{a_1/a_3}+1}{\sqrt{a_1/a_3}-1}}_{u_1}, r = s \cdot \sin\alpha \underbrace{\frac{\sqrt{a_2/a_4}+1}{\sqrt{a_2/a_4}-1}}_{u_2} \tag{7}$$

and

$$\alpha = \arctan\frac{u_1}{u_2} \tag{8}$$

Design limitations. This approach has two design limitations: it requires the sender and receiver electrodes to have the same orientation (i.e. vertical) and to be roughly in the same plane (horizontal to the orientation):

- The first limitation holds no practical difficulties. Our robot maintains a specific orientation, caused by its center of gravity. By placing one of the sender and receiver electrodes on top and one on the bottom of the robot the orientation is always vertical.
- The derivation above is only correct if sender and receiver are on the same plane, which is horizontal to the orientation of their electrodes. In a three dimensional environment this is not always true, but usually the working space is wider than it is high, even in a 3D environment.

Even so, we are working on overcoming these limitations. We are confident that the second can be eliminated by rearranging the receiver electrodes. To overcome the first limitation additional electrodes may be needed.

IV. Experiments

As described in the previous sections, different local S&C systems utilize the same hardware components for sensing and communication. Moreover, they use modal and submodal approaches, which provide not only message transmission, but also deliver spatial information about position and distances of robots and objects. In this section we describe several behavioral experiments, performed with these systems.

A. Experiments with Bi-modal Optical System

One of the implemented scenarios with AquaJelly robots had the following form, see Fig. 5. In water, a robot sends sequentially in all IR channels its own ID. Listening and sending times are selected as approx. 95% listening and 5% sending, so that all robots most of time silently observe the environment. Receiving another-than-own ID means meeting another robot, whereas non-ID IR light means own reflection from passive objects. Granularity of IR channels is enough for rough collision avoidance with objects, e.g. walls of the aquarium. Collision avoidance based on digital channels are impossible for more than two robots (or robot and object). In opposite, blue light channels emit almost all time.

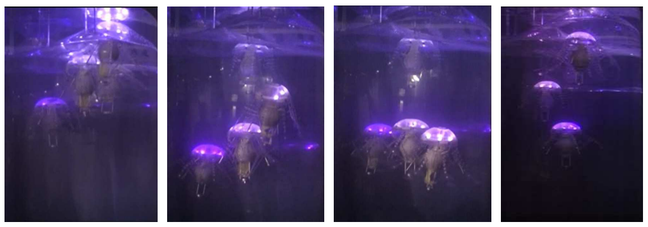


Fig. 5. Experiment with collective decision making during the docking approach.

Since light has additive properties, when two robots meet each other, intensity of light in the point of light-spheres-intersection creates a light gradient and can be locally sensed by both robots. Especially interesting is the light gradient when several robots meet each other; they create complex gradients, which can be used for precise multi-robot navigation. Unfortunately, blue-light sensors continuously receive signals from their own light sphere so that no efficient communication is possible in this mode.

Initially all robots are fully charged. When a robot has a low energy value, it swims up and recharges. With the progress of experiment, more and more robots swim up for recharging. In this way, several robots meet in the upper part of the aquarium and compete for the docking station, see Fig. 5(from left to right). Since only a robot with lowest energy value should recharge, all robots bilaterally exchange values of their own energy level. The robot with the lowest energy value can swim up. This local behavior leads to the following interesting collective behavior. Due to light gradient created by many robots, all robots exert “optical pressure” on each other, and collectively swim down, whereas only one most ”hungry” robot swims up and recharges, see Fig. 5(right).

B. *Experiment with Encoded Blue Light*

In order to investigate the modulated blue light S&C system, we used underwater submarine toy as a mechanical platform with new electronic components for locomotion, computational and S&C capabilities. This submarine has three degrees of freedom and three actuators for moving forward/backward, turning left/right, and diving up to 1 meter. The necessary modifications of the submarine including the replacement of the original electronic parts with the new designed electronic boards, and drilling some new holes on the robot’s body for communication/sensing transducers placements. Cortex3 LM3S316 microcontroller with 25MHz of clock frequency, 16kb of internal flash ROM, and 16kb of RAM has been used in the platform. Two motor drivers and two navigational sensors are placed on the main board of the electronic platform. The combination of the available PWM output from Cortex3 and motor driver perform the ability to control the swimming velocity via software. A digital compass and pressure sensor are added as a three-dimensional orientation sensor (an low-frequency RF part is foreseen for a backup communication with host).

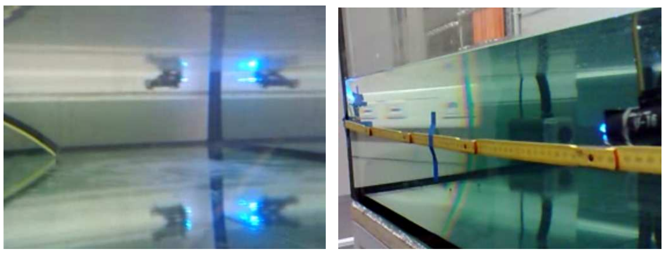


Fig. 6. **(Left)** Autonomously swimming AUV recognizes obstacles with the digital S&C system (active sensing); **(Right)** Experiments with digital communication.

During the experiment, robot is deployed into the aquarium fulfilled with several obstacles. The available obstacles and aquarium’s walls are used to examine the sensing capability, see Fig. 6(left). Both types of obstacles have different type of optical characteristic, which create different reflection behavior for the blue light. Therefore, white papers can be put outside the aquarium walls to increase the reflection capability of the optical sensor.

For testing the communication and active sensing capabilities, one robot is deployed underwater and a static encoded blue light transceiver is installed on the aquarium wall as a measurement reference point, see Fig. 6(right). The static transceiver can illuminate several type of light signals if it receives a specific blue light packet data from the swimming robot. Different types of blinking signals are used to examine the functionality of the active sensing capability. This approach underlies several other experiments, where a few passive robots are identified by one active AUV as foraging targets.

C. *Experiment with Electric Field*

The circuit for electric field communication is very simple. It consists mainly of a digital-analog-converter (DAC) for the sender and four amplifiers (OP) with analog-digital-converters (ADC) for the receiver (Fig. 7).

- *Sender:* The output of the 14-bit DAC is directly tied to one of the sender electrodes while the other is connected to VCC/2. The electrodes are in direct contact with the surrounding water. The output of the DAC can be set to a voltage between GND and VCC. This setup allows control of the field intensity and polarity.
- *Receiver:* The receiver has four pairs of electrodes in the water to measure the difference in the potential of the electric field in four places. The electrodes use capacitors as highpass-filters to filter DC signals. The signals are amplified by differential OPs (magnitude 1000) and digitized by 14-bit ADCs.

For the experiments the sender and receiver are put under water (Fig. 4 right). The receiver has a sampling rate of 10 kHz. The measured data is transmitted to a PC, where the bearing and distance are calculated.

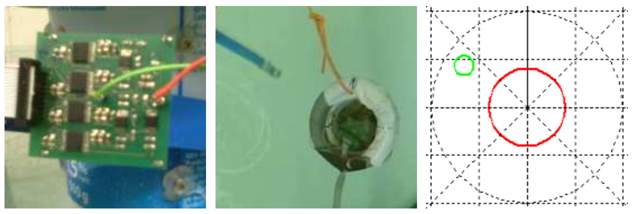


Fig. 7. Experiment electric field: test circuit, video stream, calculated bearing

In the experiment the sender was moved around the receiver and the received data was recorded together with a video tape of the experiment for comparison (Fig. 7). The true bearing was extracted from the video stream and compared with the from the electrical sensor data calculated bearing. Fig. 8 shows the result.

For 50% of the measuring points the error is less than 5° and it exceeds never more than 15°. This might be further improved by increasing the magnitude of the amplifiers and reducing the noise through optimized circuits and digital filters.

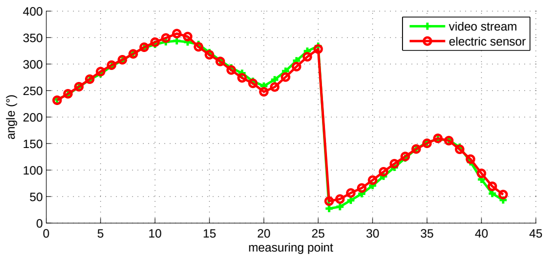


Fig. 8. Calculated bearing from data provided by (a) video stream and (b) electrical sensor

V. Conclusion

In this work we considered several optical and electric-field-based approaches for sensing and communication within R_c^l. Together with S&C approaches for R_c^g, such as acoustic and low-frequency RF, they represent the available spectra of S&C technologies for underwater networked and swarm robotics. As indicated in the Sec. IV and from other performed experiments, these approaches combine communication with localization, distance measurement and object detection. In several cases, such a sub-modal information is available even during communication and can be used for very efficient behavioral strategies.

Each of the considered S&C system has its own benefits and weaknesses. It seems that no current single system is capable of achieving all the requirements on R_c^l/R_c^g, sensing, minimal build space, energy consumption and complexity. Therefore the best approach lies in combining several of the available systems, for example in the way shown in Fig 9. The optical system provides split wave-length dependent channels. It can be used in analog and digital mode with existing control circuits for IR systems (e.g. IrDA or different modulations e.g. PCM/QAM), which with small modifications can be used for green, cyan and blue light. The range and bandwidth are sufficient for local communication. The channel is directional which can be of benefit for swarm-based coordination approaches. Additionally the reflection in analog mode can be used for navigation and detection tasks.

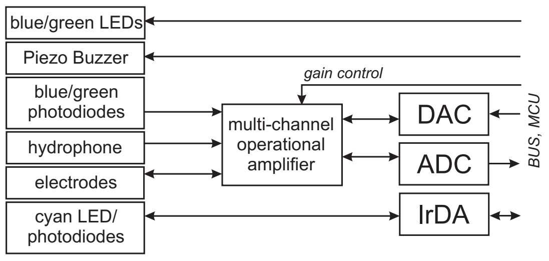


Fig. 9. Combination of different S&C approaches.

The electrical sensor is a good supplementary element to directional optics. Electric fields-based channels are omni-directional; hardware required for generation and detection of electric fields utilizes off-the-shelf components and is compact and energy efficient. It can be used to calculate the bearing between sender and receiver, i.e. for self-localization. The range is small but sufficient for R_c^l.

It is also necessary to supplement these S&C systems by acoustic or ultra-low-frequency RF to provide global communication. Sonar requires a bit larger hardware equipment than the optical system. With additional components, it can be used for measuring distances to obstacles. RF systems represent a trade-off between the frequency (i.e. communication distances) and the size of integrated antennas (i.e. the size of platform). The control circuits are more complex than those for optic or acoustic approaches. Since bandwidth for low-frequency RF is not sufficient for application of standard protocols (e.g. ZigBee), global RF communication represents some open problems. If more than one receiver is used, the bearing between sender and receiver can be calculated. However this would make the hole system more complex and expensive. Comparing acoustic and low-frequency RF approaches for R_c^g, acoustic one is more favorable due to more less complex hardware. Usage of global communication for networked and swarm systems should be reduced to absolute minimum.

Acknowledgment

The ANGELS project is funded by the European Commission within the work programm "Future and Emergent Technologies" under the grant agreement no. 231845. We want to thank all members of the project for fruitful discussions.

References

[1] K9keystrokes. Tiny robot fish may help save sea life from BP oil leak. *http://hubpages.com/hub/Tiny-Robot-Fish-may-help-Save-sea-life-from-BP-Oil-Leak*, 2010.
[2] David Bingham, Tony Drake, Andrew Hill, and Roger Lott. The application of autonomous underwater vehicle (AUV) technology in the oil industry vision and experiences. In *FIG XXII International Congress Washington, D.C. USA, April 19-26 2002*, pages 1–13, 2002.
[3] Jim Partan and Jim Kurose Brian Neil Levine. A survey of practical issues in underwater networks. In *In Proc. ACM WUWNet*, pages 17–24, 2006.
[4] FESTO. *An artificial jellyfish with electric drive unit: an autonomously controlled jellyfish.* Festo AG & Co. KG, 2008.
[5] ANGELS. *ANGuilliform robot with ELectric Sense, EU-project 231845, 2009-2011.* European Communities, 2009.
[6] CoCoRo. *Collective Cognitive Robotics, EU-project 270382, 2010-2013.* European Communities, 2010.
[7] S. Kornienko, O. Kornienko, and P. Levi. Collective AI: context awareness via communication. In *Proc. of the IJCAI 2005, Edinburgh, UK*, 2005.
[8] S. Kernbach. *Structural Self-organization in Multi-Agents and Multi-Robotic Systems*. Logos Verlag, Berlin, 2008.
[9] Liu Lanbo, Zhou Shengli, and Cui Jun-Hong. Prospects and problems of wireless communication for underwater sensor networks. *Wirel. Commun. Mob. Comput.*, 8(8):977–994, 2008.
[10] F. Schill, U. Zimmer, and J. Trumpf. Visible spectrum optical communications and distance sensing for underwater applications. In *In Proc. Australasian Conf. Robotics and Automation, Canberra*, 2004.
[11] Ian F. Akyildiz, Dario Pompili, and Tommaso Melodia. Challenges for efficient communication in underwater acoustic sensor networks. *ACM SIGBED Review*, 2004.
[12] Marvin Siegel and Ronold King. Electromagnetic propagation between antennas submerged in the ocean. *IEEE Transactions on Antenas and Propagation*, AP-21(4):507–513, 1973.
[13] S. Kernbach. From robot swarm to artificial organisms: Self-organization of structures, adaptivity and self-development. In P. Levi and S. Kernbach, editors, *Symbiotic Multi-Robot Organisms Reliability, Adaptability, Evolution*. Springer, Berlin, 2010.
[14] Charls Oatley. *Electric and magnetic fields*. Cambridge University Press, 1976.

Intelligent Wheelchair Simulation: Requirements and Architectural Issues

Marcelo Petry, Antonio Paulo Moreira
Robotics and Intelligent Systems - INESCPorto
Faculty of Engineering, University of Porto – FEUP/UP
Porto, Portugal
marcelo.petry@fe.up.pt, amoreira@fe.up.pt

Luis Paulo Reis, Rosaldo Rossetti
Artificial Intelligence and Computer Science Laboratory
Faculty of Engineering, University of Porto – FEUP/UP
Porto, Portugal
lpreis@fe.up.pt, rossetti@fe.up.pt

***Abstract*—Using robotic simulators, researchers are able to improve the algorithms of their robotics platforms before testing them in real environments. In fact, the safe environment provided by simulation is important, especially for robots that are constantly in contact with human beings, such as assistive robots and intelligent wheelchairs. Here, we propose to take advantage of an available general robotics simulator to model the IntellWheel's intelligent wheelchair prototype and its environment, enabling patient's drills and creating a test bed to refine and to experiment new methodologies of autonomous navigation, obstacle avoidance and human-machine interaction. As a result of the evaluation of four general simulators, the USARSim simulator has demonstrated to be more suitable to serve as basis for the IntellWheels simulation prototype. The development of a rough model of the intelligent wheelchair and of an appropriate test environment proved that with some modifications USARSim is able to provide a very realistic simulation platform for Intelligent Wheelchairs.**

Keywords- Intelligent Wheelchair, Simulation, Mixed Reality

I. INTRODUCTION

In the attempt of assisting people with mobility problems, many research projects of intelligent wheelchairs (IWs) have been created over the last years [1]. According to the general concept, an intelligent wheelchair can be defined as a robotic device built from an electric powered wheelchair, provided with a sensorial system, actuators and processing capabilities. At the same time, it is assumed that IW may present at least some skills such as autonomous navigation, autonomous planning, extended human-machine interaction, semi-autonomous behavior with obstacle avoidance, cooperative and collaborative behaviors. Thus, IW may be a good solution to assist severely handicapped people who are unable to operate classic electric wheelchairs by themselves in their daily activities. The aim of this paper is to take advantage of an available robotics simulator to model an intelligent wheelchair and its behavior, in order to create a test bed for new methodologies of autonomous navigation, obstacle avoidance and human-machine interaction, for instance.

Up to a recent past the use of simulations for simulating IWs (as any robot in general) was quite restricted due to the lack of general simulators. Usually, the existing simulators were developed to deal with some quite specific situations and environments. The development of a new tool for the simulation of IWs is time and resource consuming, and frequently is out of the project's scope. However, this reality started changing due to the release of general simulators and to the advantages of using robotics simulators.

Simulations have a great potential for low cost analysis, since it is able to give researchers access to cost-prohibitive sensors and robotic platforms. In addition, simulators provide the ability to compress time, and so, to evaluate the results of time-consuming experiments much faster. They are pedagogically proven technique for training [2], so they can be used to drill people in safe environments. They allow testing under repeatable and controllable conditions, simplifying debugging (e.g. the same scenario can be precisely generated to trigger a known error). Unlike real testing environments, which may not be accessible, or may only be accessible at certain times, simulated environments have unlimited availability [3]. For example, experiments that require special natural illumination (i.e. sun light) may be accessible for just some hours a day, and experiments requiring special weather conditions (like fog, rain, etc.) may be accessible just a few times a year. Simulations also provide researchers virtual access to different testing environments, making these virtual testing very cost effictive. Actually, with the right modelling, the behavior of the robot can be tested in any environment (from a reconstruction of a laboratory up to urban environments, desserts, catastrophes, lakes, oceans, others planets, etc). Finally, the extensive use of simulators allows researchers to safely refine their algorithms before testing the robot behaviour in real environments. Although researchers are no more required to develop a simulator from scratch, the simulation results just reflect the reality when the simulation requirements are considered and when the appropriate models are introduced.

A. *Requirements for the simulation of robotic systems*

The requirements for simulating mobile robots may differ according to the purpose of the simulation. For testing motion control, a higher level of detail in multi-body may be important. On the other hand, for testing sensor data processing, a higher fidelity in the sensors measures is desirible. If the simulation aims to evaluate higher level of abstractions, like global localization, ground truth data should be provided. If machine vision is used by the robot, a good rendering is required.

Essentially, simulation requirements can be classified into physical fidelity and functional fidelity. The former concerns with how the simulation looks, sounds and feels. In other words, it is the ability of the simulator to render high resolution textures, shaders, lighting and reflection. The second concerns with the simulation of most of the forces acting on robots and on its actuators, including not only but gravity, dragging, accelerations and collisions [4].

B. *Characteristics of the intelligent wheelchair*

The IntellWheels Project is focused on creating a platform to develop intelligent wheelchairs, to help people with severe disabilities to live a more normal life. It is mainly concerned with the research and design of a multi-agent system to enable an easy integration of distinct sensors, actuators, user input devices, navigation methodologies, intelligent planning techniques and cooperation methodologies. This platform aims to facilitate the development and testing of new methodologies and techniques, and to integrate it with minor modifications into most of the commercially available electric wheelchairs. We believe that this platform can bring real capabilities to the wheelchairs, such as intelligent planning, autonomous and semi-autonomous navigation, thus making it possible to execute the desired displacement through a high-level command language.

As depicted in the Fig. 1, the hardware architecture of the IntellWheels prototype is composed by a set of encoders, ultrasounds and infrared sensors, input controls and by an ordinary powered wheelchair. The software platform, as illustrated in Fig. 2, relies on a multi-agent system (MAS) architecture composed basically by four agents, currently under development in Object Pascal [5], [6].

In order to achieve that, in Section 2, we will analyse the existing robotics simulators to model the IntellWheels intelligent wheelchair prototype. Then, throughout Section 3, we will discuss about the architecture and the main features of selected simulator. In Section 4 we describe preliminary results of the simulation, and finally, in Section 5, we will present the conclusions and suggested future improvements for this work

II. Related Work

Currently, an extensive number of simulators are available for robotics research. In [4], Craighead *et al.* identify the weakness and strengths of 14 commercial and open source simulators.

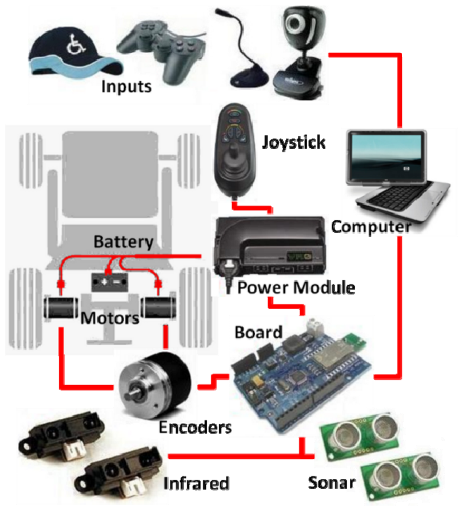


Figure 1. IntellWheels hardware architecture

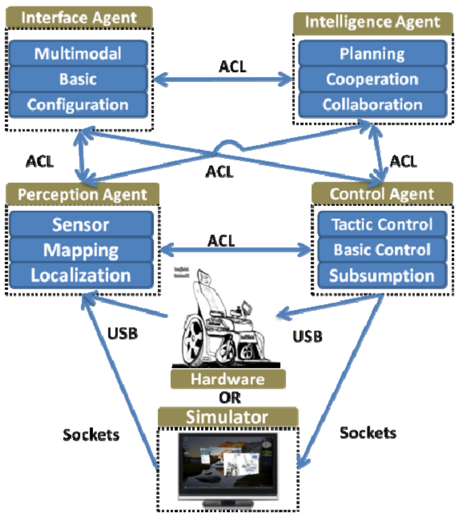


Figure 2. IntellWheels software architecture

However, in our case, specificities of the IntellWheels may be taken into account when choosing a tool to simulate intelligent wheelchairs. We also agree with Carpin et al. [7], when they claim that the simulation of robotic platforms should not consider a robot as an isolated entity, but as an entity which interact and is affected by the environment where it is situated. Therefore, we have restricted our analysis on four simulators that are able to offer these possibilities: Unified System for Automation and Robot Simulation (USARSim) [8], Microsoft Robotics Developer Studio (RDS) [9], Webots [10] and Gazebo [11].

A. *Criteria for evaluating robotics simulators*

In order select the available robotic simulator that better model the IntellWheels prototype, we propose the evaluation of the simulator using a set of seven criteria:

1) *Import 3D models* – we define this criteria as the ability of a simulator to import three-dimensional models of objects from typical Computer Aided Design (CAD) programs (such as Solidworks, Autocad, Pro-engineer, etc). We believe that this ability can facilitate the development of a more realistic model, thus improving the simulation. The evaluation of this criterion receives "yes" when the simulator supports importing objects and "no" when it does not.

2) *Programming language* – in this criterion we identify which programming languages are supported by the simulator to create the program that controls the robot. A wide support in the programming langue criteria is desired. In addition, we look specifically for a simulator that supports object pascal, once the IntellWheels platform is currently under development in that language. The evaluation of this criterion receives the list of the supported languages.

3) *External agent support* – concerns the ability to run the agent(s) that control the robot from outside of the simulator. This characteristic is desired because we want to be able to distribute the agents that control an IntellWheels prototype and the agents that provide additional services in more than one computer. This way, it is possible to increase the robustness of the system, since an agent can assume the tasks of other agents that for any reason are not answering. The evaluation of this criterion receives "yes" when the simulator supports external agents and “no” whenever it does not.

4) *Multi-thread support* – is the ability of the simulator to run more than one simulation task simultaneously. This ability is important to improve the simulation efficiency. The evaluation of this criterion receives “yes” when the simulator supports multi-thread and “no” when it does not support.

5) *Physics Engine* - concerns the identification of the libraries used for computing physics simulation. The main task of all physics engines is to solve the motion of the system given the forces acting on it. Therefore, they play a very important role in the simulation of dynamic systems because they are directly responsible for its functional fidelity. On the other hand, physics engines have a indirect responsibility also in the physical fidelity of the simulation. Particularly, the way that a simulation looks is closely dependent on the type of features the physic engine is able to simulate. For example, simulations with deformable objects demonstrate a greater realism over those which consider objects as rigid bodies, the simulation of fluids, like fog, may be important for machine vision and for video feedback, and so on. In subsection 2.2 the weaknesses and strengths of each library will be discussed in more detail. The evaluation of this criterion receives the name of the library used in each simulator.

6) *License* – corresponds to the monetary cost for the developer and for the end user. The evaluation of this criterion can receive the value “Open Source” for those simulators that are released with their source code, “free” for simulators that are available without any monetary compensation and without their source code, and “commercial” for those simulators that require monetary compensation.

7) *Sensors* – in this last criterion we identify which sensors are released with the simulators and if the simulator allows developers to create new sensors.

B. Evaluation results

Each simulator was evaluated through its User Manual or equivalent documentation, and results can be summarized in Table 1. With the exception of Gazebo, all simulators evaluated can import 3D models from typical CAD tools. However, when comparing the programming language, only USARSim and Gazebo can cope with a wider support. It is possible because these simulators rely on a client/server architecture with communication through UDP protocol, which also provides the support to external agents. Regarding multi-thread support, only USARSim and Microsoft Robotics Studio are able to benefit from the simultaneous task processing. Despite several libraries for physics computation available (PhysX, Bullet, JigLib, Newton, ODE, Tokamak, True Axis) [4], only PhysX and ODE were used by the four robotics simulators chosen for comparison.

ODE (Open Dynamics Engine) is an open-source library that is designed for simulations of rigid bodies and articulated bodies dynamics. For this reason, this library is not able to support the simulation of deformable objects, particles and fluids. ODE is platform independent with an easy to use C/C++ API. The kind of applications ODE was developed for also explains some of its characteristics, since ODE was developed to support speed, the physics accuracy tends to be compromised. On the other hand, PhysX is a proprietary solution widely used in Epic games. It provides support to the main platforms for games and graphics (such as PS3, XBOX, PC, etc.). Its main advantage consists in supporting not only rigid and articulated bodies, but also fluids (such as water, blood, smoke, gas, etc.) and particles (such as sparks, scattered glass fragments, dust, etc.). PhysX has a faster physics integration algorithm, and provides a more stable simulation when dealing with the collision of several objects [12]. In addition to the physics library, nVidia has also developed a special hardware device: the Physics Processing Unit (PPU).

With respect to the license, Gazebo and USARSim are open source simulators. At this point, it may be noticed that despite USARSim is open source, it relies on a proprietary engine and so has a small monetary cost corresponding to the Unreal Tournament 3. In its latest version Microsoft has combined the previous Express, Standard and Academic licenses into one license (RDS 2008 R3) free of charge, while the Webots has a commercial license with versions that costs from €250,00 Eur. (EDU version) up to €2600,00 Eur. (PRO version). Finally, the analyses of the sensors criteria revealed that all four simulators present the basic sensors used in the IntellWheels. The only severe limitation was observed in the RDS, which does not allow researchers to develop new sensors. For these reasons, USARSim was selected to simulate the IntellWheels prototypes. We have considered the lack of support for Object Pascal of the RDS and Webots, the limitation in the development of new sensors of the RDS, the cost of Webots, and the lack to support multi-task processing and to import 3D models as the main problems of the other simulators. USARSim, on the other hand, presents a superior physics engine, has the validation of several sensors and actuators and is probably the most used robotics simulator within the scientific community.

III. USARSIM SIMULATOR

As elegantly described by Carpin *et al.* [7], “USARsim is a general-purpose multi-robot simulator that can be extended to model arbitrary application scenarios”. It was designed to create physically accurate simulations of robots for research in fields like the human-robot interaction and multi-robot coordination. The simulator is built upon a commercial game engine thanks to the architecture of the Unreal Tournament 3, which separates the game logic and rules from simulation dynamics and environmental data. This way the game core code was reused and applied to a more comprehensive simulation, providing USARsim with high realistic visual rendering and high performance physics simulation. A further advantage relies in the fact that every improvement driven by the gaming industry translates directly into simulation advantages, which is particularly true for hyper realistic rendering and physical simulation [13].

The simulator is open source under the GPL licensing, and platform independent, running under operating systems like Windows, Linux an MacOS. USARSim is highly configurable and extensible, allowing users to develop new sensors, to model new robots and to create and re-create virtually any desired environment. As a consequence of its advantages, USARSim has become quite widespread within the scientific community, which has released a number of improvements. Simultaneously, researchers have published several papers with quantitative evaluations that demonstrate a very close similarity between the real world and USARSim system and sensors [8].

TABLE I. MAIN CHARACTERISTICS OF THE MOST USED 3D SIMULATORS IN THE ROBOTICS

Features		USARSim	RDS	Webots	Gazebo
Import 3D models		yes	yes	yes	no
Programming language		Any (UDP)	C#, VB, JScript, IronPython	C,C++,Java, Python, MATLAB	Any (TCP/UDP)
External agent support		yes	no	no	yes
Multi-thread support		yes	yes	no	no
Physics Engine		UT3 with PhysX	PhysX	ODE	ODE
License		Open Source*	Free	Commercial	Open Source
Sensors	Camera	yes	yes	yes	yes
	Touch sensors	no	yes	yes	yes
	Sonar Sensors	yes	yes	yes	yes
	Infra-red	yes	yes	yes	yes
	Sound sensor	yes	no	no	no
	GPS	yes	yes	yes	yes
	RFID	yes	no	no	yes
	Laser Range Finders	yes	yes	yes	yes
	Create new sensor	yes	no	yes	yes

The Unreal Engine is responsible for the sound, physics engine (collision detection and collision response), scripting, animation, threading, streaming, memory management and for rendering 3D graphics. On the initialization of the simulator, the Unreal Engine loads the set of geometrical models that describes all the objects in the simulation environment. For each object it is possible to specify its shape, colour and texture (among other properties). In addition, the Unreal Engine also loads a set of classes of compiled scripts that govern the behaviour of loaded models [14].

However, once the Unreal Engine is proprietary, it is not possible to establish a straight communication between the clients and the server. Instead, all the information exchange (in both directions with the engine) may occur through the network by means of a middleware application called Gamebots. The client side includes the Unreal client and the controller or the user side applications. Unreal clients are responsible for providing video feedback, rendering the simulated environment. The whole system architecture is depicted in Fig. 3.

A. Communication and control

All communication between the controllers and the Unreal Server is made through Gamebots. This middleware opens a TCP/IP socket for communication, allowing up to 16 connections (by default). The communication follows the Gamebots protocol, which is divided into Messages and Commands, following the structure:

data_type {segment1} {segment2}…

Where: 'data_type' specifies the type of the data and 'segment' specify the list of name/value pairs.

In Gamebots, Messages are a specific type of communication that contains information about the robot state (data_type = STA) or about the sensor data collected (data_type = SEN). On the other hand, commands contain instructions to control the world (data_type = INI), the robot (data_type = DRIVE), the camera (data_type = CAMERA) or robot's sensors (data_type = SENSOR). The main commands used for controlling a simulated intelligent wheelchair are briefly described above:

1) *Init* - This command adds a robot to the simulation, and is instantiated as:

INIT {ClassName robot_name} {Location x,y,z}

Where: {ClassName robot_name} 'robot_name' is the class name of the robot and {Location x,y,z} 'x,y,z' is the stat position of the robot in Unreal Unit.

2) *Drive* - This command is used to control the left and right side wheels, on a percentage of maxValue:

DRIVE {Left float} {Right float} {Light bool} {Flip bool}

Where: {Left float} 'float' is the spin speed for the left side wheel and {Right float} 'float' is spin speed for the right side wheel. Their range is –1.0 ~1.0 (move backward and move forward respectively). {Light bool} 'bool' is the Boolean value for turning on or turning off the headlight. {Flip bool} 'bool' is the Boolean value for flipping the robot.

3) *Camera* – This command controls the robot camera orientation and focus.

CAMERA {Rotation pitch,yaw,roll} {Zoom int}

Where: {Rotation pitch,yaw,roll} 'pitch,yaw,roll' is the relative value or absolute rotation angle of the camera. {Zoom int} 'int' is the zoom value. Positive values means zoom in, while negative values means zoom out.

B. Sensors

In USARSim, each virtual sensor is treated as an instance of a sensor class. Despite all the objects of a sensor class have the same sensing capability, it is possible to configure each sensor individually with different parameters (e.g. noise, distortion), allowing them to be simulated as close as possible of their counterparts in real systems. In addition, it is also possible to create a new type of sensors derived from a pre-existing sensor class.

Currently, just a few classes of sensors were already ported from the previous version of USARSim to the new UT3 version. Among those, we can find three classes of sensors that may be used in the simulation of the IntellWheels prototype: encoder, ground through and Range Finder sensors. Fig. 4 depicts the main classes of sensor present in USARSim.

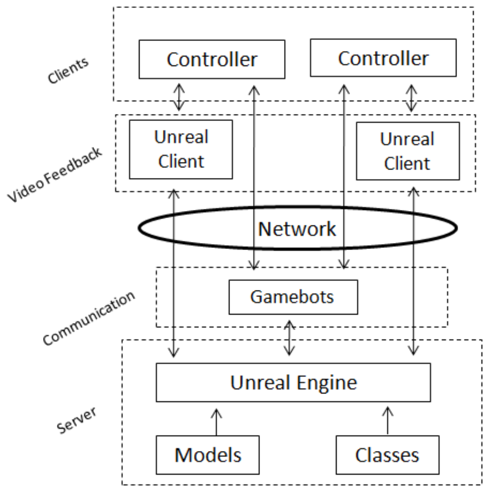


Figure 3. Architecture of the USARSim simulator (adapted from [8])

Classes of sensors drawn in blue were already ported from the previous version of USARSim and are distributed with the beta release [8]. Classes in green and purple were not ported from UT2004. However, in order to simulate the full set of sensors present in IntellWheels prototype, two subclasses of sensor (drawn in purple) were specially implemented in this project. A brief description of each class and its main characteristics will be presented bellow:

1) Encoder - This sensor measures a part's spin angle around the sensor's axis. The returned value is a tick count which is the real angle divided by the sensor's resolution. There are three parameters that can be set up in this sensor:

- Resolution: minimum spin angle the sensor can recognize [radians]
  Noise: it is the maximum amplitude of the noise [% of the truth measure]. The returned value containing the number of ticks (NTicks) with noise is then computed through the following equation:

$$NTicks = (1 + rand(-noise, noise)) * NTicks. \quad (1)$$

- Wheel: attach the encoder to its respective wheel. To perform such set up, one may use 'W' followed by the wheel number as the name of the sensor (e.g. EncoderW1).

The output of this sensor is a Message containing the type and name of the sensor and the number of ticks:

SEN {Type Encoder} {Name EncoderW1} {Tick NumTicks}

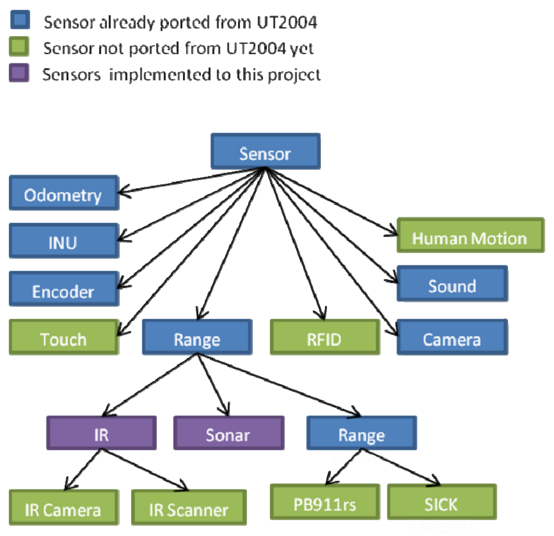
2) Ground truth sensor - This sensor returns accurate measures of the robots global position and orientation. Since it does not introduce any noise in its measures, the output data may be used to verify the performance of the robots localization algorithms, as well as for debugging and for testing high level algorithms (e.g. decision, planning, collaboration). The output of the Ground truth sensor is a Message with the type and the name of the sensor, the position and orientation of the robot at a given time stamp (Timestamp):

SEN {Time Timestamp} {Type GroundTruth} {Name GndTruth} {Location x,y,z} {Orientation roll,pitch,yaw }

3) Infra-red sensor - The IR sensor class implements the simulation of Infra-red sensors. This kind of sensor is used to detect the distance to the closest point (object) that lies on the line from the sensors position with the direction of the sensor. A full set up of this sensor includes the following parameters:

- HiddenSensor: Boolean variable used to indicate whether the sensor will be visually.
- MaxRange: maximum distance in which the sensor can detect objects [m].
- ScanInterval: time difference between two consecutive readings.
- Noise: it is the maximum amplitude of the noise [% of the truth measure]. The returned value containing the range distance (d) with noise is then computed through the following equation:

$$d = \big(1 + rand(-noise, noise)\big)d. \quad (2)$$



🟦 Sensor already ported from UT2004
🟩 Sensor not ported from UT2004 yet
🟪 Sensors implemented to this project
Sensor
Odometry
INU
Encoder
Touch
Range
RFID
Camera
Sound
Human Motion
IR
Sonar
Range
IR Camera
IR Scanner
PB911rs
SICK

Figure 4. Classes of sensors in USARSim

- bWithTimeStamp: Boolean variable used to indicate whether the time stamp in the output message is included.

A typical IR Sensor output Message includes information about the time stamp (Timestamp), the type of the sensor (Range), its name (IR1) and the measured distance (DistanceValue):

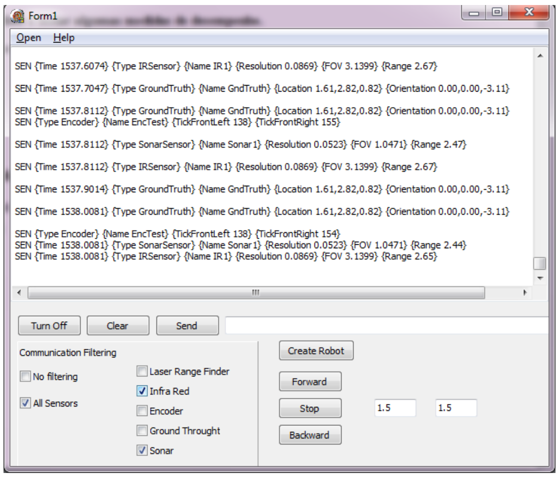
SEN {Time Timestamp} {Type Range} {Name IR1 Range DistanceValue}

4) Sonar - Sonar sensor class was designed to implement the simulation of ultrasound sensors, following the same concept used to implement it in previous versions of USARSim (as a series of IR sensors). Thus, data is obtained by rotating the sensor step by step (resolution) from the start to the end direction (field of view). Finally, the lowest from the data gathered by the sensor is then returned in the output Message. This presents the folowing parameters:

- HiddenSensor: Boolean variable used to indicate whether the sensor will be visible.
- MaxRange: maximum distance in which the sensor can detect objects [m].
- ScanInterval: time difference between two consecutive readings [s].
- Resolution: number of radians of each step [rad].
- Noise: it is the maximum amplitude of the noise [% of the truth value measured]. The returned value containing the range distance (d) with noise is then computed through (2).
- ScanFov: sensor's field of view [rad].
- bWithTimeStamp: Boolean variable used to whether include or not the time stamp in the output message.

The output message of sonar sensors contains information about the time stamp (Timestamp), the type of the sensor (Range), its name (Sonar1) and the measured distance (DistanceValue):

SEN {Time Timestamp} {Type Range} {Name Sonar1 Range DistanceValue}



```
SEN {Time 1537.6074} {Type IRSensor} {Name IR1} {Resolution 0.0869} {FOV 3.1399} {Range 2.67}

SEN {Time 1537.7047} {Type GroundTruth} {Name GndTruth} {Location 1.61,2.82,0.82} {Orientation 0.00,0.00,-3.11}

SEN {Time 1537.8112} {Type GroundTruth} {Name GndTruth} {Location 1.61,2.82,0.82} {Orientation 0.00,0.00,-3.11}
SEN {Type Encoder} {Name EncTest} {TickFrontLeft 138} {TickFrontRight 155}

SEN {Time 1537.8112} {Type SonarSensor} {Name Sonar1} {Resolution 0.0523} {FOV 1.0471} {Range 2.47}

SEN {Time 1537.8112} {Type IRSensor} {Name IR1} {Resolution 0.0869} {FOV 3.1399} {Range 2.67}

SEN {Time 1537.9014} {Type GroundTruth} {Name GndTruth} {Location 1.61,2.82,0.82} {Orientation 0.00,0.00,-3.11}

SEN {Time 1538.0081} {Type GroundTruth} {Name GndTruth} {Location 1.61,2.82,0.82} {Orientation 0.00,0.00,-3.11}

SEN {Type Encoder} {Name EncTest} {TickFrontLeft 138} {TickFrontRight 154}
SEN {Time 1538.0081} {Type SonarSensor} {Name Sonar1} {Resolution 0.0523} {FOV 1.0471} {Range 2.44}
SEN {Time 1538.0081} {Type IRSensor} {Name IR1} {Resolution 0.0869} {FOV 3.1399} {Range 2.65}
```

Figure 5. GUI of the robot controller

IV. PRELIMINAR EXPERIMENTS

In order to run initial experiments of communication between the control agent and the simulator we have used a robotic platform released with the USARSim simulator. P3AT is a skid-steer robot with 50cm x 49cm x 26cm of body size and four 21.5cm diameter wheels from ActivMedia Robotic. With proper configuration, the robot was provided with a camera and a set of eight sonars and eight infrared sensors assembled in a semi-circular ring around its body. In order receive the feedback from the simulator, a simple Graphic User Interface (GUI) was developed (Fig. 5). Through that interface, it is possible to create the robot in USARSim, parse the message containing the measures of each sensor and steer the vehicle in the simulated world.

V. CONCLUSIONS

In this paper, we have discussed the benefits of simulation in robotics and presented the requirements and characteristics for the simulation of intelligent wheelchairs – more specifically to the prototype developed in IntellWheels project. A set of seven criteria were proposed to assist in the evaluation of robotics simulators.

The results of the evaluation have demonstrated that both RDS and Webots lack on supporting Object Pascal. Also, RDS has a severe limitation in the development of new sensors, and Webots has high monetary cost associated. USARSim, on the other hand, has demonstrated a superior physics engine and a validation of several sensors and actuators. Thus, USARSim was selected for simulating the IntellWheels prototype. Once some important classes of sensors had not been ported from the previous version of the USARSim simulator, we have implemented one class for simulating the ultra-sound sensors and one class for simulating the infra-red sensors. Preliminary results were achieved using a P3AT platform configured with eight sonars and infrared sensors. A robot controller was developed in order to create and steer the robot in the simulated world and to parse the messages received from simulator.

As future work, we intend to design a realistic model of the wheelchair and of a cluttered environment. In addition, with the integration of the agents that control the IntellWheels prototype, we intend to create mixed reality environments. Drills of patients with real wheelchairs in virtual scenarios could be performed with increased realism, eliminating the risk of injuries and the stress of steering the wheelchair in the real environment. Furthermore, mixed reality experiments would make it possible to test the real IW in several scenarios (e.g. narrow corridors, crowded places, moving objects) in a safe (free of collisions with real objects, reducing the risk damaging the equipment) and inexpensive way (reduced time demanded to create scenarios, and minimum infra-structure cost).

ACKNOLEDGMENT

The first author would like to acknowledge FCT (grant SFRH/BD/60727/2009) for his PhD scholarship funding. This work was supported by the Fundação para a Ciência e Tecnologia through the project "INTELLWHEELS - Intelligent Wheelchair with Flexible Multimodal Interface" (grant ref. FCT/RIPD/ADA/109636/2009).

REFERENCES

[1] R. C. Simpson, "Smart wheelchairs: A literature review," *Journal of Rehabilitation Research and Development,* vol. 42, pp. 423-435, 2005. ISSN:0748-7711 .

[2] M. Friedmann, K. Petersen and O. von Stryk, "Simulation of multi-robot teams with flexible level of detail," *Simulation, modeling, and programming for autonomous robots*. vol. 5325: Springer, 2008, pp. 29-40. ISSN:0302-9743 .

[3] C. Pepper, S. Balakirsky and C. Scrapper, "Robot simulation physics validation," Workshop on Performance Metrics for Intelligent Systems, Washington DC,USA, pp.97-104, 2007.

[4] J. Craighead, R. Murphy, J. Burke, B. Goldiez, "A survey of commercial & open source unmanned vehicle simulators," in *IEEE Int. Conf. Robotics and Automation*, Rome, Italy, pp. 852-857, 10-14 April 2007. ISBN: 1-4244-0601-3.

[5] R. A. M. Braga, M. Petry, A. P. Moreira and L. P. Reis, "Concept and design of the intellwheels development platform for intelligent wheelchairs," in *Informatics in control, automation and robotics*. vol. 37, Springer-Verlag, 2009, pp. 191-203. ISSN:978-3-642-00270-0.

[6] R. A. M. Braga, M. Petry, A. P. Moreira and L. P. Reis, "Intellwheels - a development platform for intelligent wheelchairs for disabled people," in *Int. Conf. Informatics in Control, Automation and Robotics*, Funchal, PORTUGAL, pp. 115-121, May 11-15 2008. ISBN: 978-989-8111-31-9.

[7] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, C. Scrapper, "Usarsim: A robot simulator for research and education," *IEEE Int. Conf. Robotics and Automation - ICRA*, Rome, Italy, pp. 1400-1405, April 2007. ISBN: 1-4244-0601-3

[8] B. Balaguer, S. Balakirsky, S. Carpin, M. Lewis and C. Scrapper, "Usarsim: A validated simulator for research in robotics and automation," Workshop on "Robot simulators: available software, scientific applications and future trends", at IEEE/RSJ IROS 2008

[9] Microsoft. *Microsoft robotics developer studio 2008,* http://www.Microsoft.Com/robotics/.(Consulted on January 2011)

[10] O. Michel, "Webotstm: Professional mobile robot simulation," *International Journal of Advanced Robotics Systems,* vol. 1, pp. 39-42, 2004

[11] Gazebo. *Gazebo user manual,*available at http://playerstage.Sourceforge.Net/index.Php?Src=doc. (Consulted on January 2011)

[12] M. Ma, M. McNeill, S. McDonough, J. Crosbie and L. Oliver, "Physics fidelity of virtual reality in motor rehabilitation," the Physics Fidelity of Virtual Reality in Motor Rehabilitation, Laval, France, pp. 35-41, April 2006

[13] M. Lewis, J. Wang and S. Hughes, "Usarsim: Simulation for the study of human-robot interaction," *Journal of Cognitive Engineering and Decision Making,* vol. 1, pp. 98-120, 2007

[14] J. Wang. (2005, october). *Usarsim v2.0.2 - a game based simulation of the nist reference arenas,* http://sourceforge.Net/projects/usarsim.(Consulted on February 2011)