RAC ROBOTIC SOCCER SMALL-SIZE TEAM: CONTROL ARCHITECTURE AND GLOBAL VISION

José Rui Simões * **Rui Rocha** * **Jorge Lobo** *
Jorge Dias *

* *Dep. of Electrical and Computer Engineering,*
Faculty of Sciences and Technology, University of Coimbra,
3030-290 Coimbra, PORTUGAL
{*jrmart*}*@alumni.deec.uc.pt,*
{*rprocha, jlobo, jorge*}*@deec.uc.pt*

Abstract: The main goal of the RAC project is to develop a robotic soccer team for participating in the RoboCup's small-size league competitions. This paper is focused on two main issues: distributed control of a team of robotic soccer robots; and computer vision for tracking and localization. A distributed control architecture tailored for robotic soccer is presented, which aims at fostering cooperation in the absence of centralized control. It is also described a global vision system based on two overhead cameras and color segmentation, which provides the RAC team with real-time information about the game.

Keywords: robotic soccer, distributed architectures, computer vision, color segmentation.

1. INTRODUCTION

The RAC project is an endeavor of the University of Coimbra, which aims at stimulating teaching and research about mobile robotics among researchers and graduate students, through competitions inside and outside the school. The project's main goal is to develop a robotic soccer team for the RoboCup's small-size league (SSL).

The RoboCup (Kitano *et al.* 1998) is an international initiative aiming at fostering intelligent robotics research. It provides standard problems requiring the integration of important technologies, such as autonomous robots, multi-agent cooperation, real-time reasoning, sensor-fusion, computer vision, mechatronics, artificial intelligence, etc. The RoboCup comprises essentially two test beds: robotic soccer and rescue. Both represent robotic applications without human intervention, occurring in adversarial, highly dynamic and uncertain environments, and requiring cooperation among robotic agents. RoboCup rescue aims at specifically promoting research in socially significant issues. Robotic soccer is an exciting domain for intelligent multi-agent robotics, requiring real-time sensing, action and decision making in a harsh and dynamic environment, where the teammates of a multi-robot team must cooperate to defeat the adversarial team. Because the decision making involves simultaneously cooperation and competition, robotic soccer is already considered a benchmark for the progress of robotics and artificial intelligence (Buss *et al.* 2003). Due to its connection with the very popular soccer game, it has been growing very fast for the past few years, with increasing number of competitions and participant teams from different countries all over the world.

1.1 Teamwork and control paradigms

Manuela Veloso and Peter Stone have been involved in robotic soccer since 1996, using it to study multi-agent learning (Veloso and Stone 1998). They participated on the development of CMUnited architecture, which includes a command server for strategic collaboration between teammates, through reactive control and formations (Stone and Veloso 1999). In (Behnke and Rojas 2001), it is described a hierarchy of reactive behaviors for robotic soccer, where deliberation is not implemented and each control layer is of type sense-think-act. Tews et al. developed the Multi-Agent Planning System (MAPS) (Tews and Wyeth 2000), where agents share a similar perceived model of the world, deriving individual coordinated actions without extensive negotiations, using the superposition of potential fields concept. Aparicio developed a functional architecture for a team of mobile robots (Aparício 2000), comprising three levels: individual, relational and organizational. These levels use, respectively, the concepts of primitive tasks, behaviors (using sense-think-act paradigm) and joint intentions (Cohen and Levesque 1991), where a joint intention can be roughly defined as a property that holds a group together in a shared activity. Tucker Balch has developed the social potentials technique for the formation control of multi-robot teams (Balch and Hybinette 2000). Such formations are implemented as a reactive navigation problem based on a collection of motor schemas with different objectives and restrictions. This technique suggests that sometimes effective cooperative teams can be composed of agents using simple individual agent behaviors with limited or no communication. Extensive work has been done in the field of Multi-Agent Systems (MAS) for the last two decades (Stone and Veloso 2000). The cooperation and coordination of activities in MAS is not easily scalable due to the required huge amount of explicit communication and to the time required to take extensive negotiations, which are not appropriate in real-time decision making. Thus, current multi-robot architectures result in strategies mainly reactive and less deliberative, where the attained cooperation is emergent.

One scientific goal of the RAC project is to develop a new real-time distributed, cooperative architecture for a team of soccer robots, comprising both reactive and deliberative control. The main idea is to combine these two control paradigms, as a means to take advantage from the responsiveness of reactive control and from the ability of deliberative control to perform complex and intelligent behaviors, while coping with the hard real-time requirements imposed by the soccer game. Section 2 presents the distributed architecture that we are implementing within the RAC project.

1.2 Intelligent sensors

Another scientific goal of the RAC project is to develop new intelligent sensors, based on the integration of artificial vision and inertial sensors. A small vision system is being implemented, which integrates inertial sensors on the robots, improving their autonomy. From our previous experience on inertial and vision sensor fusion, we intend to explore more dynamic situations, such as having an inertial-sensor-enhanced visual tracker. The high velocities of the small league robots enable the use of low cost micro-machined inertial sensors for short range inertial navigation to be used together with visual landmark and odometry data. The inertial sensors will be used to detect and assess robot collisions.

At the current stage of the project, robots have still low sensory power. Thus, most of the information about the game is dependent on a global vision system, providing real-time information about the game, namely the position of the players and the ball. Section 3 describes our global vision system, based on two overhead cameras and color segmentation, which provides the RAC robotic team with real-time information about the game.

2. DISTRIBUTED CONTROL ARCHITECTURE

The RAC project's current goal is to develop robotic soccer teams for the small-size league (SSL) of the RoboCup. In SSL, the robots have small size (18cm diameter) and have usually very limited computational and sensory capabilities, being the emphasis usually placed on the robots' dynamics and speed. For this reason, the SSL games are very fast and impose the hardest real-time requirements amongst the robotic soccer modalities. The SSL teams are usually controlled in a centralized way from a remote PC (e.g. (Veloso and Stone 1998)), being the robots used mainly as actuators, providing fast motion and other additional basic skills, such as kicking or dribbling the ball. Because of the lack of on-board sensory power, using global vision system, comprising one or more cameras positioned over the playing field, is currently allowed by the competition rules.

Although a centralized control approach fits well with SSL, given that a system completely dependent from a global vision can be assumed, the RAC project's long-term goals go beyond current SSL requirements. In fact, the long-term goal of

the project is to evolve towards a middle-size league team deployment, which requires on-board sensory and computational power and high control autonomy. For this reason, the team's control architecture has been developed in such a way that it can be instantiated along different levels of robot's autonomy, ranging from a centralized system completely dependent from global vision to a completely distributed system, where each robot's individual controller runs locally in the robot's own embedded computer and the robot possesses higher sensory power (e.g. on-board vision). More specifically, the RAC's control architecture main features are: three different abstraction levels of robot's control, with a suitable balance between reactivity and deliberation; distribution, modularity and flexibility; and centralized world model based on sensor fusion.

The main idea behind the RAC's control architecture is to combine both reactive and deliberative control paradigms, as a means to take advantage from the responsiveness of reactive control and from the ability of deliberative control to perform complex and intelligent behaviors, while coping with the hard real-time requirements imposed by the soccer game (especially the SSL games). Reactive control allow robots to exhibit fast response to new events (e.g. intercepting the ball), fast basic behaviors (e.g. moving while controlling the ball) and taking chance opportunities (e.g. shooting to goal). The deliberative control makes possible to define intelligent playing strategies based on cooperation among robots, using explicit communication. For reactive behaviors, cooperation is mainly emergent and does not use communication with teammates. Deliberative behaviors are devoted to more complex and time consuming decision making activities, implementing explicit cooperation through coordination and negotiation protocols between the robotic agents, which usually require explicit communication.

2.1 General description

Figure 1 depicts a general diagram of the RAC's distributed control architecture. It is composed of a centralized world model (WM), processes that directly access WM, one controller for each robot in our team (robotic agent) and an interface to global vision (GLBSENS).

The world model (WM) is a centralized repository of the information about the game, owned by the team. It stores fundamental data about the game, such as pose and velocity of all players and the ball, game's score, game's state, etc. This information is used by all the robots to take decisions under a consistent and common world representation. Modules UPDMOD and RFRINT are inter-

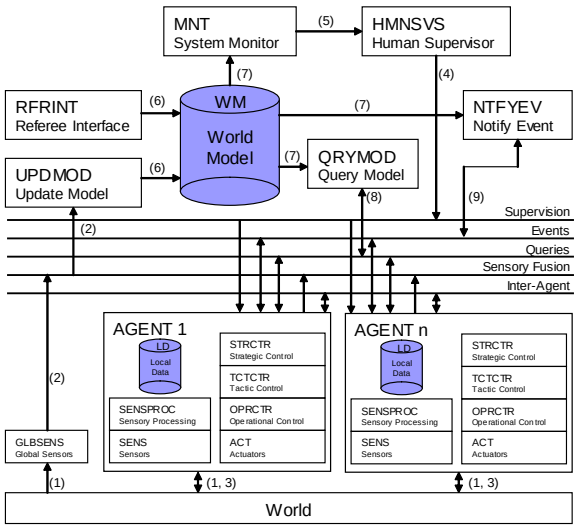

Fig. 1. General diagram of the RAC's control architecture: it comprises a robotic agent for each robot, representing each robot's controller, a shared information repository (WM – World Model), modules for getting information from WM (NTFYEV, QRYMOD), modules for updating WM (GLBSENS, UPDMOD, RFRINT) and modules used for supervising the team (MNT, HMNSVS).

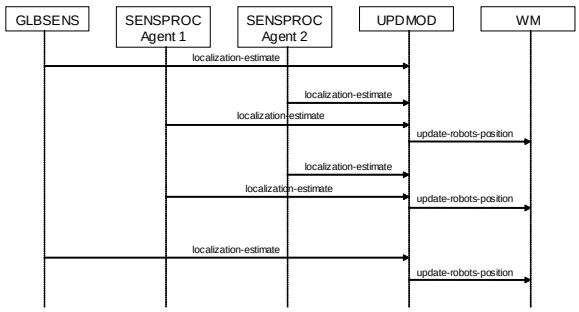

Fig. 2. Example of a sequence diagram related with updating the world model (WM).

faces with the sensory modules (including global vision) and the referee, respectively, which update and maintain WM consistent with current game's situation. The module UPDMOD implements sensor fusion from sensory information gathered by robots' local sensors and global sensors. Figure 2 shows an example of a sequence that updates WM as long as new data comes from sensors.

The module MNT implements a graphical user interface that enables the team's supervisor to monitor and log the game. The module HMNSVS enables the team's supervisor to tune the strategy followed by the team. The modules QRYMOD and NTFYEV makes WM available to all robots in the RAC team. While the former process works upon specific robot's queries about the world model (on demand information), the latter process is a way to synchronize each robot with the occurrence of

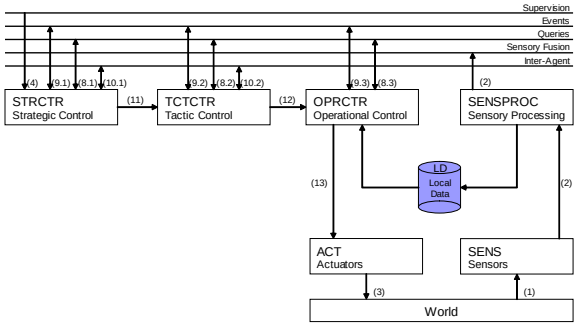

Fig. 3. Diagram of the robot's controller, comprising three control levels: operational, tactic and strategic. The different types of information are represented through different information channels.

those events for which the robot requested to be notified.

2.2 Robot's own controller

Each robot is represented in figure 1 by a robotic agent. Figure 3 shows a detailed block diagram of each robot's own controller. Both figures 1 and 3 show the different types of information conveyed between modules. There are three modules taking care with the robot's control at three different control levels: strategic (STRCTR), tactical (TCTCTR) and operational (OPRCTR). At the strategic level, the robot's behaviors are typically deliberative. At the tactical level, the robot uses both reactivity and deliberation. The operational level is mainly reactive and implements behaviors with the hardest real-time requirements. The module SENSPROC processes information gathered by the robot's own sensors (if any). Modules SENS and ACT interface directly with the robot's sensors and actuators.

The module STRCTR cooperates with peer processes of other teammates in order to select the best team's strategy under the current world state. The process TCTCTRL does the same as STRCTR but at the tactical level. It cooperates with peer processes of other teammates in order to map the chosen strategy to tactics, doing role assignment and coordination (synchronization). Figure 4 shows an example of a sequence diagram at tactical level under a defensive strategy.

The module OPRCTR implements a pool of sense-think-act control sequences (finite state machines) that are instantiated by the tactic level to perform primitive actions (e.g. dribble-ball-to-position). Each primitive action is a closed loop control sequence through the robot's sensors and actuators, although global information can also be used (see figure 5 for an example). Each robot has a local data repository, which is a sub-model of WM built upon data gathered by the robot's

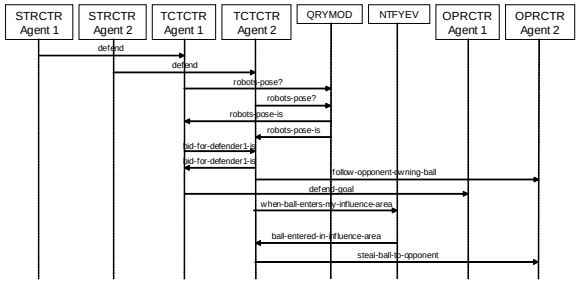

Fig. 4. Example of a sequence diagram at the tactical control level under a defensive strategy.

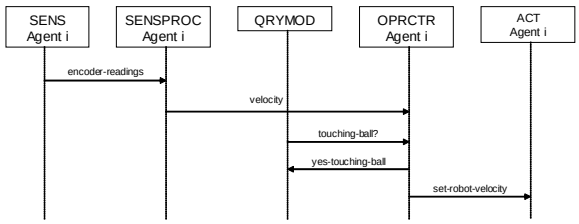

Fig. 5. Example of a sequence diagram of the execution of a primitive action at the operational control level.

1. Sensory infomation
encoders' pulses
kicker's state
image

2. Sensory data
encoder-readings
kicker-ready (flag)
localization-estimate

3. Actuators actions
wheels' movement
kicker shot

4. Supervision actions
offensive-team
defensive-team

5. Monitor data
game's score
game's statistics

6. Write access to WM
update-robots-position
update-ball-position
start-game
stop-game
free-kick
begin-first-half

7. Read access to WM
read-player-position
read-ball-postition
read-player velocity
read-game-state

8. Queries
8.1 Strategic level
robots-pose?
is-our-ball?
close-to-end-of-game?
8.2 Tactic level
position-of-opponent-that-owns-ball?
ball-position?
ball-in-my-influence-area?
8.3 Operational level
distance-to-ball?
my-pose-global-estimate?
touching-ball?

9. Notification of events
9.1 Strategic level
when-we-score-a-goal
when-we-win-ball-possession
9.2 Tactic level
when-ball-enters-my-influence-area
when-ball-in-my-line-of-view
9.3 Operational level
when-close-to-position
when-opponent-in-front

10. Inter-Agent messages
10.1 Strategic level
my-bid-for-being-captain-is
new-captain-is
new-strategy-is
10.2 Tactic level
my-bid-for-role-is
intercepted-ball
prepared-to-receive-pass

11. Strategies
defend strongly
defend
attack
counter-attack

12. Primitive actions
dribble-ball-to-position
follow-opponent-owning-ball
catch-ball-and-kick
steal-ball-to-opponent
defend-goal

13. Commands
set-robot-velocity
release-kicker
turn-on-dribbler

Fig. 6. Examples of the kind of information conveyed through the different information flows.

own sensors, suitable for the real-time operational control performed locally by the robot.

Each information flow between modules is numbered uniquely in figures 1 and 3. Figure 6 shows examples concerning the kind of conveyed among modules (not an exhaustive enumeration).

2.3 Instantiations having different levels of robot's autonomy

The architecture presented above can be easily adapted to different levels of robot's autonomy. We are particularly interested in two instances: small-size league (SSL) and middle-size league (MSL) of RoboCup. In both cases, the assignment of some modules to computational resources is fixed: modules SENS, ACT always run locally on the robots; modules UPDMOD, RFRINT, MNT, HMNSVS, QRYMOD and NTFYEV always run on a remote PC. If communication resources usage or the robots' computational power requirements have to be reduced as much as possible, all robotic agent's modules (except SENS and ACT) can run in the remote PC. Conversely, if the robot's controller should run totally in the robot itself, all robotic agent's modules processes run locally on the robot's own embedded computer.

For SSL, there are only very primitive local sensors on the robots and it is available a very powerful global sensor – an overhead camera – which is interfaced and integrated in the world model through GLOBSENS. As most of the sensory information comes from the global sensor, the implementation of module UPDMOD is very simple because sensor fusion is very minimal. If the robot's computational power is restrictive, all modules, except SENS and ACT, might run on the remote PC.

For MSL, global sensors are completely forbidden, therefore the module GLOBSENS doesn't exist. As sensory capacity is almost evenly distributed among robots, the module UPDMOD is much more complex than in the SSL instantiation. If robots have reasonable computational power, the most distributed and higher autonomy architecture instance can be implemented.

We are about to deploy our first team for robotic soccer, which uses global vision and implements a reduced version of the presented control architecture, with all robots' individual controllers running on the same PC. Figure 7 shows some photos of our first prototype. In order to increase the robots' autonomy and work without global vision, we intend to equip the robots with intelligent sensors, including vision and inertial sensors, and to implement a fully distributed version of the control architecture, by putting each robot's controller running locally on the robot's own embedded computer. For this purpose, the RAC robot's embedded computer is a PC104-based embedded computer, having the required enough computational power. Our long-term goal is to evolve gradually to the RoboCup middle-size league.

Fig. 7. RAC robot (*RACbot*) – first working prototype.

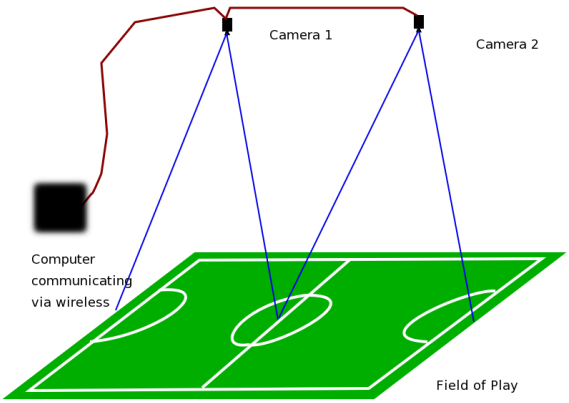

Fig. 8. Global Vision System implementation

3. GLOBAL VISION SYSTEM

3.1 Introduction

Our implementation of the global vision uses two cameras, one for each half of the field, which then report the acquired information to a central process. This process provides our soccer robots with information concerning the position of robots and the ball. Our implementation is shown in figure 8.

3.2 Camera calibration

Camera calibration (Wei 1994) is the process of computing the camera's physical parameters, such as the image center, focal length, position orientation, etc. In our case, we achieved camera calibration using the Camera Calibration Toolbox for MatLab© (Bouguet 2005), since it was readily available and met our requirements. Using a target chessboard pattern, we can calculate all the camera parameters from a set of images, including radial distortion parameters.

3.3 Coping with camera's radial distortion

Lens distortion is a phenomena that is very common, especially in wide-angle lenses. This kind

of phenomena is of relative importance for image quality, but it is of major importance when the image geometry must be maintained. In most cases, radial distortion is the most relevant type of distortion that lenses cause, and we can usually neglect all other kinds of distortion.

As we can see on the left side of figure 10, in our case, the barrel distortion is evident and, since we need the coordinates of robots and the ball with a high degree of confidence, this image is unacceptable. Thus, using the distortion parameters calculated with Camera Calibration Toolbox for MatLab© (Bouguet 2005), and applying the undistort routine supplied by the toolbox, a rectified version of the image is readily obtained (see the resulting image in the right side of figure 10). In our software, this functionality will be achieved using a similar routine included in the OpenCV distribution (OpenCV 2005).

3.4 Color segmentation

Vision systems employing region segmentation by color are crucial in real-time mobile robot applications, such as RoboCup (Kitano *et al.* 1997), or other domains where interaction with humans or a dynamic world is required. An important first step in many color vision tasks is to classify each pixel of the image into one of a discrete number of color classes (Bruce *et al.* 2000). To achieve this objective, there are four main approaches: linear color thresholding, nearest neighbor classification, color space thresholding and probabilistic methods. We have applied a method that falls mainly on the third approach, that is, we use two threshold levels for each channel to uniquely define each color.

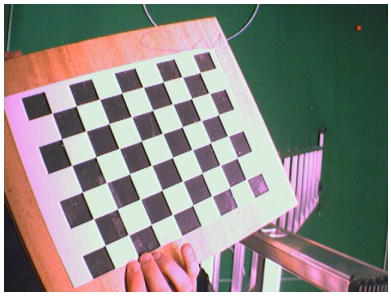

Fig. 9. Sample image used for camera calibration.

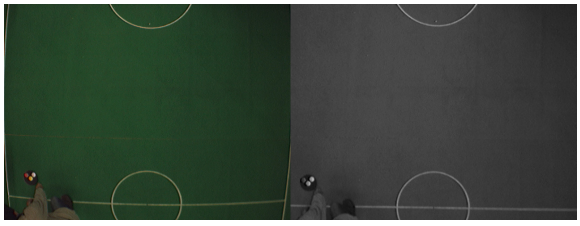

Fig. 10. Compound of two images: on the left, the original image having significant radial distortion; on the right, the radial distortion is corrected.

```
Do Forever   Acquire a new frame
  For all the pixels
    check if the color of the pixel
    is one of the colors of interest

  Extract contours from the regions created
    If contour is of right size
       Save contour information
       Position and colour
  Build the robots and
  get the ball position

  Send positions
```

Table 1. Pseudo-code of our algorithm.

3.4.1. Color Space There are several color spaces used in computer vision (RGB, YUV, YCrCb,HSV, etc.). Since we are using OpenCV for capturing images, the image is required to use the OpenCV's standard color space BGR when acquiring images. This color space is equivalent to RGB, but the color channels appear in reverse order. Consequently, it has the same problems than RGB color space; for instance, the luminance is encoded in all of the three channels. That's why we convert the acquired image to the YCrCb color space, which is a variant of YUV. This color space is much more interesting, because it separates luminance in the channel Y and the chrominance is coded in the other two channels. Although we are introducing some overhead when doing this conversion, the reliability on detecting the different markers is greatly improved, because the color detection is much less influenced by different lightning conditions.

3.4.2. Implementation Our implementation was based on the algorithm described in (Bruce *et al.* 2000). We begin by converting the color space to YCrCb and then we apply the thresholds for each pixel of the frame, which results in a binary image with all the regions of interest. The result of this can be seen in figure 11. This binary image is further processed for finding contours (shown in figure 12) and looking for regions of interest. This decision is based on the area of the region. Finally, we extract the position, heading and identification of the robot from the information gathered in the previous steps, which is marked in figure 13. If the region represents the ball, only the position is acquired, since there is no other relevant information about the ball in the image. Regarding the opposite team we can only acquire their position, because we don't know *a priori* the type of markers that they use for heading and identification.

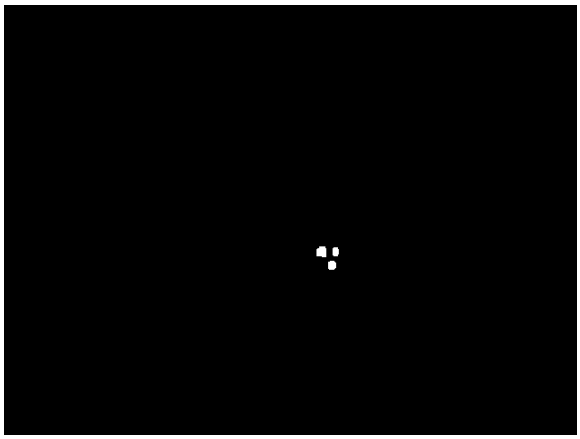

Fig. 11. Result of color segmentation.

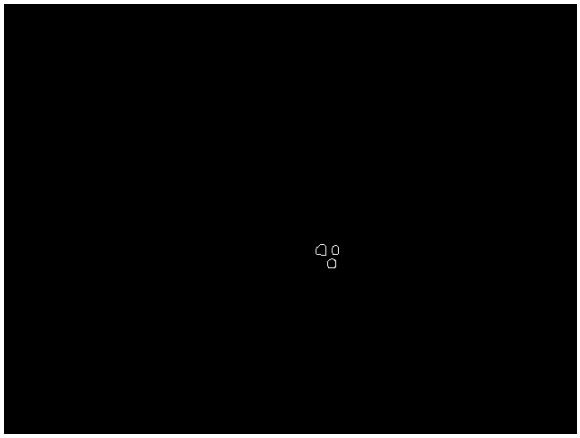

Fig. 12. Contours retrieved from color segmentation.

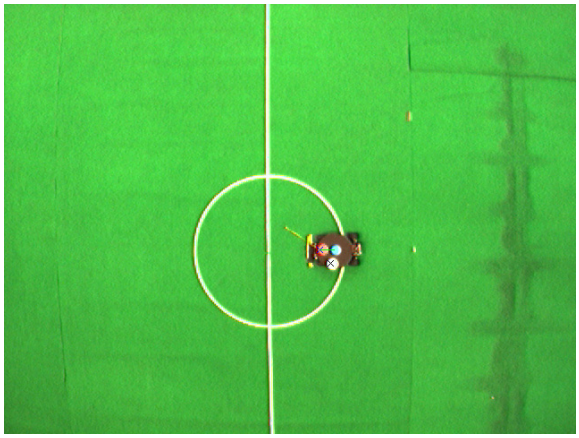

Fig. 13. Result of a positive detection of one of the robots.

3.5 Experiments and results

3.5.1. Framerate The framerate that we achieve is relatively low. Our cameras limit ourselves to 15 frames per second, but adding the processing of each frame, we have at most 5 frames per second. This was the highest frame rate observed since this value is highly dependent from the amount of noise and false positives in each frame.

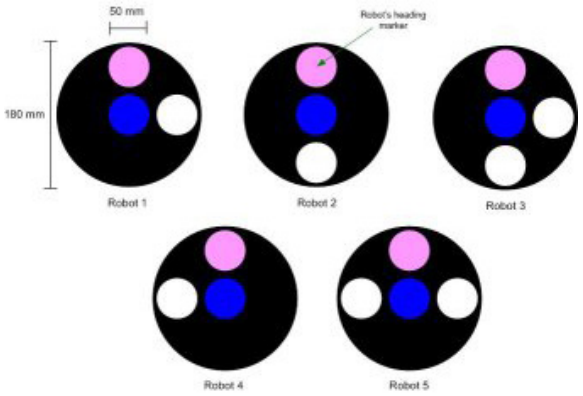

Fig. 14. Colored markers used on the top of the robots to localize and identify them through color segmentation.

	Ball		Robot		
	X (cm)	Y (cm)	X (cm)	Y (cm)	θ (^{o})
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	-176.424
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	-176.424
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	43.5183	0.9472	-175.914
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	-176.424
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	43.5183	0.9472	180
	-61.3931	50.1168	43.5183	0.9472	180
	-61.3931	50.1168	44.1069	0.9134	-176.424
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	44.1069	0.9134	-176.424
	-61.3931	50.1168	44.1069	0.9134	180
	-61.3931	50.1168	43.5183	0.9472	-175.914
Mean	-61.3931	50.1168	44.0301	0.9178	178.8791
σ	0	0	0.2357	0.0135	1.7164

Table 2. Results of static detection.

3.5.2. Results The markers that we are currently using are shown in figure 14. Since the prototype is still being developed, the tests are not very extensive. In table 2 we see the results for static detection of the ball and robot. As we can see, the measurements are very accurate, having a minimal standard deviation σ. This is in part due to using two cameras, since we have bigger regions to estimate the objects' position. The second set of tests, whose results are presented in table 3, were conducted with the robot going trough the center line, i.e. with x = 0. As we can see, the standard deviation is relatively bigger but still acceptable.

4. CONCLUSION

This article focused on two main issues of the RAC project. Firstly, it was presented a distributed, cooperative control architecture for a robotic soccer team. It combines both reactive and deliberative

	Position in cm		Heading
	X (cm)	Y (cm)	θ (^{o})
	-2.29885	129.337	83.991
	1.72414	109.75	85.9144
	2.29885	99.9736	90
	1.14943	78.7224	87.1376
	8.34465	33.333	81.8699
	1.14943	11.481	85.9144
	14.9425	-40.4016	83.6598
	13.7931	-50.7332	78.1113
	11.4943	-56.454	80.5377
	12.6437	-64.5131	80.5377
	13.7931	-77.17	83.6598
	19.5402	-101.949	93.3665
	16.092	-102.484	94.3987
Mean	7.6765		85.3357
σ	7.1087		4.7235

Table 3. Results of detection with a moving robot.

control and provides each robot with as much control autonomy as possible, while coping with the real-time requirements imposed by the soccer game. Secondly, as our current robots have not onboard vision, the team's global vision system was presented, which is based on two overhead cameras and real-time color segmentation algorithms.

We are about to deploy our first team for robotic soccer, which uses global vision and implements a reduced version of the presented control architecture, with all robots' individual controllers running on the same PC. In order to increase the robots' autonomy and work without global vision, we intend to equip the robots with intelligent sensors, including vision and inertial sensors, and to implement a fully distributed version of the control architecture, by putting each robot's controller running locally on the robot's own embedded computer. Our long-term goal is to evolve gradually to the RoboCup middle-size league.

ACKNOWLEDGMENTS

This work was funded by FCT Fundação para a Ciência e a Tecnologia, with project grant POSI/ROBO/43890/2002. This work was possible thanks to the contribution from all RAC team members.

REFERENCES

Aparício, Pedro (2000). Design and implementation of a population of cooperative autonomous robots. Master's thesis. Instituto Superior Técnico. Lisboa, Portugal. Supervised by Pedro U. Lima.

Balch, T. and M. Hybinette (2000). Social potentials for scalable multi-robot formations. In: *Proc. of IEEE International Conference on Robotics and Automation (ICRA'00)*. pp. 73–80.

Behnke, S. and R. Rojas (2001). A hierarchy of reactive behaviors handles complexity. In: *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*. pp. 125–136. Springer-Verlag.

Bouguet, Jean-Yves (2005). Camera Calibration Toolbox for Matlab. URL http://www.vision.caltech.edu/bouguetj/calib_doc.

Bruce, J., T. Balch and Veloso M. (2000). Fast and inexpensive color image segmentation for interactive robots. Technical report. School of Computer Science, Carnegie Mellon University, USA.

Buss, M., M. Hardt, J. Kiener, M. Sobotka, M. Stelzer, O. Stryk and D. Wollherr (2003). Towards an autonomous, humanoid, and dynamically walking robot: modelling, optimal trajectory planning, hardware architecture, and experiments. In: *Proc. of IEEE/RAS Int. Conf. on Humanoid Robots, Karlsruhe, Germany*.

Cohen, P. and H. Levesque (1991). Teamwork. *Nous, Special Issue on Cognitive Science and Artificial Intelligence* **25**(4), 487–512.

Kitano, H., I. Kuniyoshi, M. Asada, H. Matsubara and Osawa E. (1997). Robocup: A challenge problem for ai. *AI Magazine* **18**(1), 73–85.

Kitano, H., M. Asada, I. Noda and H. Matsubara (1998). RoboCup: Robot world cup. *IEEE Robotics and Automation Magazine* **5**(3), 30–36.

OpenCV (2005). OpenCV-Open Computer Vision Library. URL http://sourceforge.net/projects/opencvlibrary.

Stone, P. and M. Veloso (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence* **110**(2), 241–273.

Stone, P. and M. Veloso (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots, Special Issue on Heterogeneous Multi-Robot Systems* **8**(3), 345–383.

Tews, A. and Wyeth (2000). MAPS: a system for multi-agent coordination. Technical report. Computer Science and Electrical Engineering, University of Queensland, Australia.

Veloso, M. and P. Stone (1998). Towards collaborative and adversarial learning: a case study in robotic soccer. *Int. Journal of Human Computer Studies*.

Wei, Guo-Qing (1994). Implicit and explicit camera calibration: Theory and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(5), 469–480.