CAMBADA: Information Sharing and Team Coordination

Nuno Lau, *Member, IEEE*, Luís Seabra Lopes, *Member, IEEE* and Gustavo A. Corrente

***Abstract*— This paper presents the architecture, information sharing and team coordination methodologies of the CAMBADA RoboCup middle-size league (MSL) team. An overview of the software architecture and individual decision capabilities of the agents is also presented. The information sharing and integration strategy is designed to both improve the accuracy of world models and to support the team coordination. Part of the coordination model is based on previous work in the Simulation League, which has been adapted to the MSL environment. With the described design, CAMBADA reached the 1st place in the Portuguese Robotics Open in 2007 and the 5th place in RoboCup 2007 world championship.**

I. Introduction

ROBOTIC soccer is currently one of the most popular research domains in the area of multi-robot systems. The RoboCup rules and regulations for different robotic soccer modalities are widely accepted and followed. Many robotic soccer projects use RoboCup competitions for testing and validation of the adopted approaches.

In the context of RoboCup, the so-called “middle-size league” (MSL) is one of the most challenging, since robotic players must be completely autonomous and must play in a field of 12 m 18 m [13]. In this modality, teams are composed of at most six wheeled robots with a maximum height of 80 cm and a maximum weight of 40 Kg. The rules of this modality establish several constraints to simplify perception and world modeling. In particular, the ball is orange, the field is green, the field lines are white, the players are black, etc. The duration of a game is 30 minutes, not including a half-time interval of 5 minutes. The referee orders are communicated to the teams using an application called “referee box”. The referee box sends the referee orders to the team through a wired LAN TCP link connected to the base station of each team. It is the team's responsibility to communicate these orders to the robots inside the field via standard wireless LAN. No human interference is allowed during the games except for removing malfunctioning robots and re-entering robots in the game.

Building a team for the MSL is a very challenging task, both at the hardware and software level. To be competitive, robots must be robust and fast and possess a comprehensive set of sensors. At the software level these robots must have an efficient set of low-level behaviors and must coordinate themselves to operate as a team. Coordination in the MSL league is usually achieved through the assignment of different roles to the robots. Typically there is, at least, an attacker, a defender, a supporter and a goalie [21][2]. As the maximum number of robots in each team increases (it is currently 6) and the field becomes larger, more sophisticated coordination techniques must be developed.

In the RoboCup simulation league teams have been using coordination schemes based on a coordination layer that includes Strategy, Tactics and Formations [17][20], coordination graphs [10] and reinforcement learning [19].

CAMBADA is the RoboCup middle-size league soccer team of the University of Aveiro (Fig. 1). This project started officially in October 2003 and was initially funded by the Portuguese research foundation (FCT). Since then, CAMBADA participated in several national and international competitions, including RoboCup world championships, the European "RoboLudens” and the annual Portuguese Open Robotics Festival.

The CAMBADA project aims at fostering the Aveiro university research at several levels of the MSL challenge. Research conducted within this project has led to developments at the hardware level [3], infrastructure level [1][15][16], vision system [14][5], multi-agent monitoring [9] and high-level decision and coordination [4]. This paper is focused on the last of these components.

This paper is organized as follows: Section II presents the hardware and software architectures of CAMBADA players and provides details on the main software components involved in individual decisions of the players, namely roles and behaviors. Section III describes how players share information with teammates and how they integrate shared information. Section IV describes the adopted coordination methodologies. Section V presents the latest results and concludes the paper.

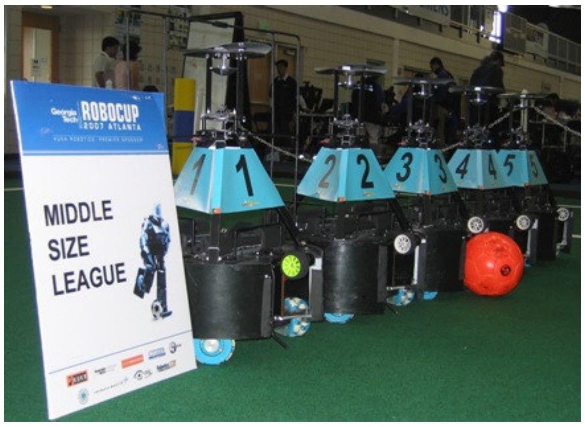

Fig. 1 CAMBADA robotic team

Nuno Lau and Luís Seabra Lopes are with the Transverse Activity on Intelligent Robotics of IEETA research unit as well as with the Department of Electronics, Telecommunications and Informatics, Universidade de Aveiro, Portugal. (e-mails: { nunolau, lsl }@ua.pt).

Gustavo Corrente was a researcher with the Transverse Activity on Intelligent Robotics of IEETA research unit, Universidade de Aveiro, Portugal, and is currently with Nokia Siemens Networks Portugal, Aveiro, Portugal. (e-mail: gustavo@ua.pt)

II. Player Architecture

A. Hardware Architecture

The CAMBADA robots (Fig. 1) were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter. The mechanical structure of the players is layered and modular. Each layer can easily be replaced by an equivalent one. The components in the lower layer, namely motors, wheels, batteries and an electromechanical kicker, are attached to an aluminum plate placed 8 cm above the floor. The second layer contains the control electronics. The third layer contains a laptop computer, at 22.5 cm from the floor, and an omni-directional vision system, close to the maximum height of 80cm. The players are capable of holonomic motion, based on three omni-directional roller wheels. The mentioned vision system allows detecting objects (ball, players, goals) and field lines on a radius of nearly 5m around each player. Besides vision, each player includes wheel encoders, battery status sensors and, for detecting if the ball is kickable, an infra-red presence sensor.

The robots computing system architecture follows the fine-grain distributed model [11] where most of the elementary functions, e.g. closed loop control of complex actuators, are encapsulated in small microcontroller based nodes, connected through a network. A laptop node is used to execute higher-level control functions and to facilitate the interconnection of off-the-shelf devices, e.g. cameras, through standard interfaces, e.g. USB or Firewire (Fig. 2). For this purpose, Controller Area Network (CAN), a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with a higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance, namely the FTT-CAN protocol (Flexible Time-Triggered communication over CAN) [7].

In the middle-size league, inter-robot communication and communication between the team's base station and the robots is extremely necessary for the team to maintain a coordinated behavior. The communication among robots and to the base station uses the standard wireless LAN protocol IEEE 802.11x profiting from large availability of complying equipment. The base station is connected to the referee box through a wired LAN TCP link.

B. Software Architecture

The software system in each player is distributed among the various computational units (Fig. 2). High-level functions run on the computer, a laptop PC running Linux operating system. Low-level functions run partly on dedicated microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1] has been adopted. The RTDB is a data structure where players share their world models. It is updated and replicated in all players in real-time.

Fig. 3 shows the class diagram of the CAMBADA WorldState class. This class supports the information storage of ball and players positions, roles, behaviors, etc.. A module called Integrator is used to update the world state information. This is done by filtering the raw information coming from sensors (i.e. vision, odometry, etc.) and determining the best estimate of the position and velocity of each object. The World State class includes several methods that test conditions on the current situation (ex: if the robot is facing the opposite goal).

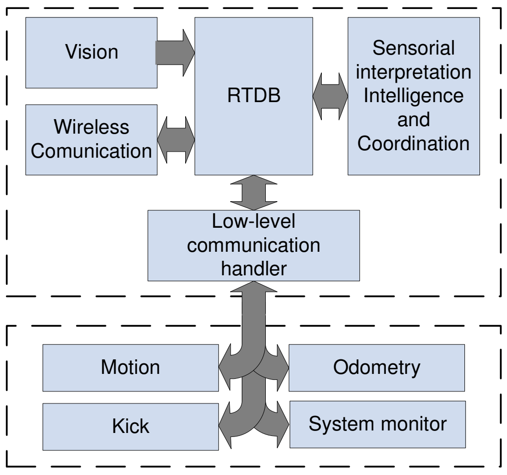
Vision
Wireless Comunication
RTDB
Sensorial interpretation Intelligence and Coordination
Low-level communication handler
Motion
Odometry
Kick
System monitor

Fig. 2. Layered software architecture of CAMBADA players, from [3]

A recent, and very important, development as been the integration into the sensor fusion module of a self-localization lines based engine, based on the one described in [12], that allows a high level of confidence in the robots estimated self position.

The high-level processing loop starts by integrating perception information gathered locally by the player. This includes information coming from the vision processes, which is stored in a Local Area of the RTDB, and odometry information coming from the holonomic base via FTT-CAN. After integration, part of the world state is written in the shared area of the RTDB to make it available to teammates. The next step is to integrate local information with information shared by teammates.

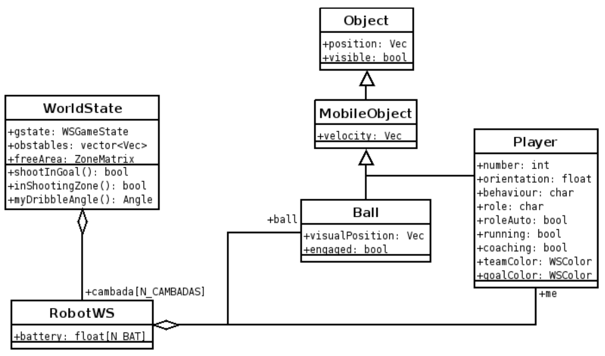

Fig. 3. *WorldState* class diagram

The software of the CAMBADA agent is composed of several different processes that have responsibility for different tasks: image acquisition, image analysis, integration/decision and communication with the low-level

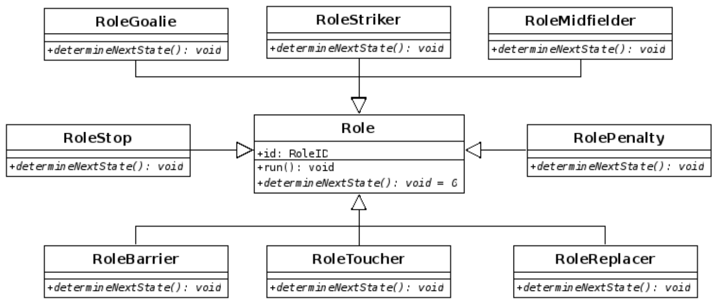
RoleGoalie
+determineNextState(): void
RoleStriker
+determineNextState(): void
RoleMidfielder
+determineNextState(): void
RoleStop
+determineNextState(): void
Role
+id: RoleID
+run(): void
+determineNextState(): void = 0
RolePenalty
+determineNextState(): void
RoleBarrier
+determineNextState(): void
RoleToucher
+determineNextState(): void
RoleReplacer
+determineNextState(): void

Fig. 4. *Role* class diagram

modules. The order and schedule of activation of these processes is performed by a processor manager library called Pman [16]. Pman stores in a database the characteristics of each process to activate and allows the activation of recurrent tasks, settling phase control (through the definition of temporal offsets), precedence restrictions, priorities, etc. The pman services allow changes in the temporal characteristics of the process schedule during run-time.

It is very important that all robots share the same play mode obtained by processing the referee orders given through the referee box. In CAMBADA, an application inside the team's base station checks the messages received from the "referee box", and converts the event triggered protocol of communication "referee box" - "base station" to a state oriented playmode information that is broadcasted to robots using the RTDB. This ensures that the delay between the reception of a referee event from the "referee box" and its awareness by all robots is minimized, enabling a synchronized collective behavior.

C. Roles and Behaviors

The CAMBADA agent decision module is based on the concepts of *role* and *behavior*. Behaviors are the basic sensorimotor skills of the robot, like moving to a specific position or kicking the ball, while roles select the active behavior at each time step.

All roles within a CAMBADA agent are derived from the *Role* abstract class (Fig. 4), whose most important element is the *determineNextState()* method. This method is responsible for the selection of the active behavior. To develop a new role, a *Role* derived class is created and the *determineNextState()* method is implemented. The *run()* method is implemented only in the base class and is responsible for the selection of the active behavior, using *determineNextState()*, and for its execution.

To change the active behavior, the method *changeBehaviour(Behaviour*)*, implemented in the *Role* base class, is used.

During play-on mode, the CAMBADA agents use only three roles: *RoleStriker*, *RoleMidfielder* and *RoleGoalie*. The *RoleGoalie* is activated for the goalkeeper.

RoleStriker is an "active player" role. It tries to catch the ball and score goals according to the finite-state machine shown in Fig. 5. The striker activates several behaviors that try to engage the ball (*MoveToBall*, *MoveOutsideBall*), get into the opponent's side avoiding obstacles (*Dribble*) and shoot to the goal (*Kick*). The *Kick* behavior can perform 180º turns while keeping possession of the ball. The *MoveOutsideBall* is used in situations where a direct catch would lead to the ball getting out of the field. In these situations the robot approaches the ball from the exterior side of the field thus pushing it inside.

RoleMidfielder is a "passive" player. It moves according to its determined strategic positioning [18]. The strategic position is determined for each positioning using a home position and then adjusting it using attractions to the ball current position. Using different home positions and attractions according to the positioning allows a simple definition of defensive, wing, midfielders and attack strategic movement models.

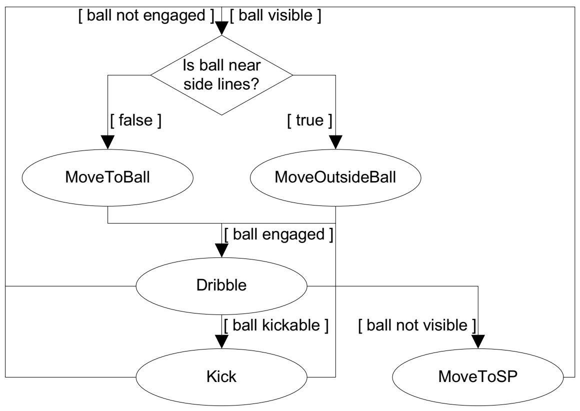

Fig. 5. Finite-state machine for decision-making in *RoleStriker*

Three more roles are used in set-pieces like kick-off, throw-in, goal-kick, corner-kick, free-kick and penalty. *RoleToucher* and *RoleReplacer* are used to overcome the indirect rule in the case of indirect set pieces. The purpose of *RoleToucher* is to touch the ball and leave it to the *RoleReplacer* player. The replacer handles the ball only after it has been touched by the toucher. This scheme allows the replacer to score a direct goal if the opportunity appears. *RoleBarrier* is used during the set-pieces against CAMBADA to protect the goal from a direct shoot. *RolePenalty* is used in penalty shootouts. It randomly chooses the goal side to which to kick and kicks the ball so that it enters the goal at 0.75m height.

The class diagram of behaviors is shown in Fig. 6. The abstract class *Behavior* is the base of all behaviors. It has three important methods. The first one is *calculate()*, an abstract method, whose implementation in derived classes determines which are the parameters of the command that this behavior intends to execute (*velX*, *velY*, *velA*, *grabberInfo* and *kickerInfo*) but does not execute them. The second is *execute()*, which sends previously computed linear, angular velocities and the kicking parameters to the low level computation modules. The separation of

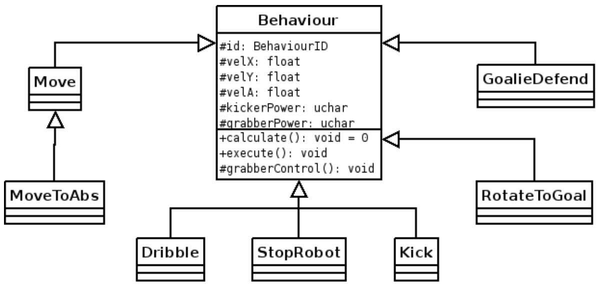

Fig. 6. *Behavior* class diagram

calculation and execution enables the agent to reason on the expected result of the commands while deciding which one to execute. Finally, *grabberControll()* controls the grabber mechanism automatically, without concerns for the behaviors developments.

The set of behaviors that are implemented in the CAMBADA agent are adapted to its catadioptric omnidirectional vision and holonomic driving systems. The combination of these technologies enhances the set of possible behaviors when compared to a differential drive robot or to an holonomic drive robot with a limited field of view.

The behavior *Move* uses two symbolic parameters: the target position where to move; and the position which the CAMBADA player should be facing in its path to the target. The symbols used are *OBall*, *TheirGoal* and *OurGoal*. The other moving behavior *MoveToAbs* allows the movement of the player to an absolute position in the game field. Those moving behaviors may activate the functions of avoiding obstacles and avoiding the ball (used during the game repositions to avoid collisions with the ball). The *Dribble* behavior is used to dribble the ball to a given relative player direction. *GoalieDefend* is the main behavior of the goalie. The *Kick* behavior is used to kick the ball accurately to one 3D position in opponent goal.

III. Information Sharing and Integration

Sharing perceptional information in a team can improve the accuracy of world models. Sharing internal state can improve the team coordination. Therefore, information sharing and integration is one of the key aspects in multi-robot teams.

In CAMBADA, each robot uses some of the perceptions of the other robots, obtained through the RTDB, to improve its knowledge about the current positions and velocities of the others robots and of the ball. It is very important for our coordination model to keep an accurate estimation of the absolute position of the ball by each robot. The role assignment algorithm is based on the absolute positions of the robot and its teammates. The teammates' positions are not obtained through the vision system and rely completely on the communicated estimated self positions of others.

Each agent communicates its own absolute position and velocity to all teammates as well as its ball information (position, velocity, visibility and engagement in robot), current role and current behavior is also shared.

The sharing of own absolute position and velocity is needed first of all because the vision system of the agents currently cannot detect the localization of the teammates. The vision system only detects obstacles but it doesn't try to detect individual robots within the detected obstacles nor does it try to determine if they are teammates or opponents. The absolute position of teammates is necessary to the strategy of our team, as the information is used to define our formation/strategy. So each robot trusts the estimated self position of teammates that is communicated through the RTDB.

Multi-robot ball position integration has been used in the middle-size league by several teams [21][6]. In CAMBADA, multi-robot ball position integration is used to maintain an updated estimate of the ball position, when the vision subsystem cannot detect the ball, and to validate robot's own ball position estimate, when the vision subsystem detects a ball.

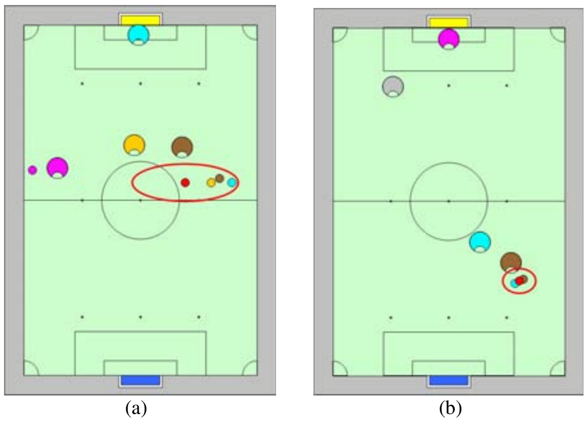

Fig. 7. Multi-robot ball position integration

Currently, a simple integration algorithm is used. When the agent doesn't see the ball, it analyzes the ball information of playing teammates. The analysis consists in the calculation of the mean and standard deviation of the ball positions, then discarding the values considered as outliers of ball position, and finally using the ball information of the teammate that has a shorter distance to ball. To determine if the agent sees a fake ball, i.e., to validate the robot's own perception, we use a similar algorithm.

Communication is also used to convey the coordination status of each robot allowing robots to detect uncoordinated behavior, for example, several robots with the same exclusive role, and to correct this situation reinforcing the reliability of coordination algorithms.

The communication between the base station and the robots informs the robots of the active playmode (decided by the referee). In some of the used setups, a coach agent, also possibly running in the base station, decides robot's roles and communicates its decisions to the robots using the RTDB. During development the base station can be used to control several robotic agent characteristics like fixed roles, fixed behaviors, manually activated self-positioning, etc, all managed through the RTDB.

IV. Team Coordination

Our coordination model is based on the definition of a strategy for a game, where each strategy may be composed of several tactics and each tactic defines a formation to be used at each situation in a similar way as SBSP strategies previously developed for the RoboCup Simulation League [18]. However several changes had to be introduced in order to adapt the coordination model to the specificities of the Middle-Size League. This model is merged with role based coordination and different priorities are assigned to the different roles and positionings. In specific situations, like kick ins, or corners, specific set-plays are activated where a coordinated and synchronized set of basic behaviors is performed by all robots in the team.

A. *Strategy of Role based strategic positioning*

Each tactic is a complete specification of the team coordinated behavior for all situations. A formation defines the movement model of the set of all robots which assigns to each positioning a home position and corresponding attractions to the ball. All these items are maintained in a strategy configuration file, to enable flexible alterations to the current strategy. To maintain a correct formation all robots should have estimations of the ball absolute position that are close to each other. Fig. 8 shows the formation of the team used in Robótica 2007 Tournament [8] for several ball positions.

The *Striker* is helped by other teammates as they maintain their strategic positioning and accompany the striker, without interfering with him, as it plays along the field. In case the ball is captured by the opponent the other mates are in good positions to become the new strikers.

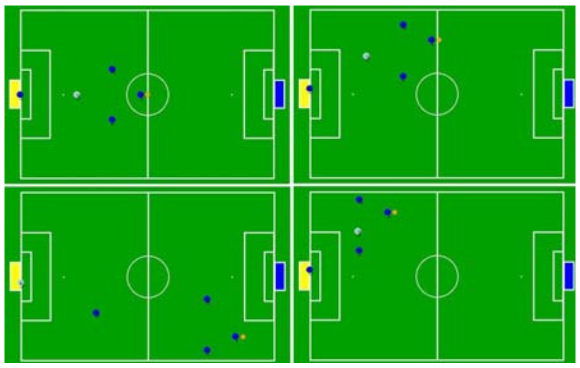

Fig. 8. Strategic positions for several different ball position

B. *Role/Positioning assignment algorithm*

So far, several different roles have been described but coordination must ensure a proper and safe role assignment algorithm. This algorithm should be able to function with a varying number of active players in the team, either because of hardware or software malfunctioning or because of referee orders. These are very common situations in the MSL.

The playon decision that assigns the *Striker* role and the positionings of the other robots in the formation is performed using an algorithm similar to DPRE [17], but with the innovation of considering different priorities for the different roles and positionings, so that the most important ones are always covered as the number of active players varies.

The algorithm is presented in Fig. 9. Considering a team of *R* field players (not counting the goal-keeper which has a different mechanical configuration and therefore a fixed role), to assign the role and positioning to each robot, the distances of all robots to all strategic positions are calculated. Then the Striker role is assigned to the robot that is closest to the highest priority strategic position, which is in turn the closest to the ball. From the remaining *R-1* robots the closest to the defensive positioning (second highest priority) is assigned to this positioning, then the closest to the third level priority positioning is assigned next and the algorithm continues until all active robots have positionings and roles assigned. This algorithm results in the Striker role having top priority, followed by the defensive positioning, followed by the other supporter positionings. The assignment algorithm may be performed by the coach agent in the base station, assuring a coordinated assignment result, or locally by each robot, in which case the inconsistencies of world models may lead to unsynchronized assignments.

```
MSL_DPRE(robotPositions, ballPosition,
         formation)

clear assignments
determine strategicPositions[N_POSITIONS]
determine distSP[N_POSITIONS][N_ROBOTS]
for each SPos sorted by priority
   determine closest free Agent to SPos
   assign SPos to Agent

Return assignments
```

Fig. 9. CAMBADA Positioning/Role assignment algorithm

C. *Set plays*

One other coordination methodology that is being used in CAMBADA is the use of predefined set plays. Currently set plays are only initiated when re-entering the play-on mode. Set plays define a sequence of behaviors for several robots in a coordinated way. Each of the tasks that compose a set play are implemented using a special role. These roles are activated at the specific situation: kick-off, kick in, corners, free kicks and goal kicks.

The assignment of the *Barrier*, *Replacer* and *Toucher* roles is executed by sorting the agents according to their perceived distances to the ball and selecting the closest ones, up to the maximum number of agents in each role. When selecting a role like the *Replacer*, which is exclusive, the agent looks at the other teammates role decisions and if it finds a *Replacer* with a lower uniform number it will never select the *Replacer* role. A similar approach is performed for the other roles. This assignment is always performed locally by each robot.

V. Conclusion

The data structures used for world state representation clearly separate the raw sensor information from the world model that results of integrating local and shared information. This architecture is easily adaptable to the addition of new sensors. Access to the world model is performed by using specific queries.

The adaptation of SBSP and DPRE [17][18] to the Middle-Size League environment resulted in a coordinated behavior of the team that contributed to its recent successes. The formation flexibility and adaptability was one of the components presented by CAMBADA in the 2nd Technical Challenge of RoboCup 2007 (based on Challenge 6 of [13]), Atlanta, where the team ranked in the 4th place. The robot malfunctions decrease the number of field players, but the positioning/role assignment algorithm maintains a competitive formation with fewer players in the field. Set plays were very efficient as several of the CAMBADA goals were the direct result of their activation.

The work described in this paper was used in two RoboCup competitions:

a. Portuguese Robotics Open 2007 (Portugal): 1st place, 6 wins, 0 draws, 0 looses, 16 goals scored and 3 goals suffered;

b. RoboCup2007 (USA): 5th place, 7 wins, 1 draws, 1 looses, 24 goals scored and 7 goals suffered.

Acknowledgment

The authors and the other CAMBADA team members thank Stefen Welker (Tribots team member) for the supplied material on the localization module, it was very useful to improve our team. Part of this work was funded through a research scholarship granted by IEETA.

References

[1] Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes, Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences -- ISCIS 2004: 19th International Symposium, Proceedings*, Aykanat, Cevdet; Dayar, Tugrul; Korpeoglu, Ibrahim, eds., Lecture Notes in Computer Science, Vol. 3280, 2004, pp. 876-886.

[2] M. Arbatzat, S. Freitag, M. Fricke, R. Hafner, C. Heermann, K. Hegelich, A. Krause, J. Krüger, M. Lauer, M. Lewandowski, A. Merke, H. Müller, M. Riedmiller, J. Schanko, M. Schulte-Hobein, M. Theile, S. Welker, D. Withopf: Creating a Robot Soccer Team from Scratch: the Brainstormers Tribots, *Proceedings of Robocup 2003, Padua*, Italy, 2003.

[3] Azevedo, J.L., M.B. Cunha, L. Almeida, Hierarchical Distributed Architectures for Autonomous Mobile Robots: a Case Study. Proc. *ETFA2007- 12th IEEE Conference on Emerging Technologies and Factory Automation*, Patras, Greece, 2007, pp. 973-980.

[4] Bartolomeu, P., L. Seabra Lopes, N. Lau, A. Pinho, L. Almeida, Integração de Informação na Equipa de Futebol Robótico CAMBADA, *Electrónica e Telecomunicações*, 4 (4), Universidade de Aveiro, Portugal, 2005, pp. 467-477.

[5] Cunha, B., J. Azevedo, N. Lau, L. Almeida, Obtaining the Inverse Distance Map from a Non-SVP Hyperbolic Catadioptric Robotic Vision System, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, editors, *RoboCup-2007: Robot Soccer World Cup XI*, LNAI, Springer Verlag, Berlin, 2008.

[6] Ferrein, A., L. Hermanns and G. Lakemeyer, Comparing Sensor Fusion Techniques for Ball Position Estimation, *RoboCup 2005: Robot Soccer World Cup IX*, A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, eds., Lecture Notes in Computer Science, 4020, Springer, 2006, pp. 154-165.

[7] Ferreira, J.; Pedreiras, P.; Almeida, L.; Fonseca, J.A. The FTT-CAN protocol for flexibility in safety-critical systems, *IEEE Micro*, 22 (4), 2002, pp. 46-55.

[8] Festival Nacional de Robótica'2007, Paderne, Portugal, http://www.ccvalg.pt/robotica2007/, 2007

[9] Figueiredo, J., Lau, N., Pereira, A. Multi-Agent Debugging and monitoring framework, *Proc. First IFAC Workshop on Multivehicle Systems (MVS'06)*, Brasil, October, 2006.

[10] Kok, J.; Spaan, M. and Vlassis, N., Non-communicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*, 50 (2-3), Elsevier Science, 2005, pp. 99-114.

[11] Kopetz, H., *Real-Time Systems Design Principles for Distributed Embedded Applications*, Kluwer, 1997.

[12] Lauer, M., S. Lange and M. Riedmiller, Calculating the perfect match: An efficient and accurate approach for robot self-localisation, *RoboCup 2005: Robot Soccer World Cup IX*, A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, eds., LNCS 4020, Springer, 2006.

[13] MSL Technical Committee 1997-2008, *Middle Size Robot League Rules and Regulations for 2008. Draft Version - 12.2 20071109*, November 9, 2007.

[14] Neves, A.; Corrente, G. and Pinho A., An omnidirectional vision system for soccer robots. *Progress in Artificial Intelligence*, Lecture Notes in Computer Science. Berlin, nº 4874, Springer, 2007, pp. 499-507.

[15] Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos, Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques, *RoboCup-2005: Robot Soccer World Cup IX*, I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, eds., Lecture Notes in Computer Science, 4020, Springer, Berlin, 2006, pp. 371-383.

[16] Pedreiras, P.; Almeida, L., Task Management for Soft Real-Time Applications Based on General Purpose Operating Systems, *Robotic Soccer*, edited by: Pedro Lima, Itech Education and Publishing, Vienna, Austria, 2007, pp. 598-607.

[17] Reis, L.P. and N. Lau, FC Portugal Team Description: RoboCup 2000 Simulation League Champion, *RoboCup-2000: Robot Soccer World Cup IV*, P. Stone, et al. eds., LNCS 2019, Springer, 2001, pp. 29-40.

[18] Reis, L.P. and N. Lau, and E.C. Oliveira, Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents, *Balancing Reactivity and Social Deliberation in Multiagent Sytems: From RoboCup to Real Word Applications*, M. Hannenbauer, J. Wendler, and E. Pagello eds., LNAI 2103, Springer-Verlag, 2001, pp. 175-197.

[19] Riedmiller, M. Gabel, T., On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup, *Proceedings of the 3rd IEEE Symposium on Computational Intelligence and Games (CIG 2007)*. IEEE Press, April 2007, pp. 17-23.

[20] Stone, P. and M. Veloso, Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic Teamwork, *Artificial Intelligence*, vol. 110 (2), 1999, pp. 241-273.

[21] Weigel, T. W. Auerbach, M. Dietl, B. Dümler, J.S. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski and M. Thiel, CS Freiburg: Doing the Right Thing in a Group, *RoboCup 2000: Robot Soccer World Cup IV*, P. Stone, G. Kraetzschmar, T. Balch, eds., Springer-Verlag, 2001, pp. 52-63.