Bochica 2011 - Team Description Paper

E. Gonzalez[1], F. De la Rosa[2], A. S. Miranda[1], C. F. Rodríguez[2], M. Manrique[1], C. Otalora[1],
J. Angel[2], Y. Arévalo [2], J. S. Figueredo[1] and R. Puerta[1]
[1]Pontificia Universidad Javeriana, Bogotá, Colombia
[2]Universidad de los Andes, Bogotá, Colombia

Abstract. In this paper we present an overview of the Bochica 2010 team, from universities Javeriana, Andes and Norte of Colombia. This team participates in the RoboCup Small Size League. The overview describes both the robot hardware and the general software architecture of our team. The main characteristics are on the one hand, concerning the hardware, an omni-directional three-wheeled based platform with dribbler and kicker, and on the other hand, concerning the software, a multi-agent layered architecture MRCC to control the multi-player system which depends on an external global vision system as sensor.

1 Introduction

BOCHICA is the first version team of Small Size F180 robot soccer league [1][2] developed as a joint effort of researchers at the universities Javeriana, Andes and Norte from Colombia.This paper will describe in section II the robot's electronic and mechanic aspects. Section III introduces the Multi-Resolution Cooperation Control arquitecture that supports the multi-agent platform to define the motions of the robots. Finally, section IV presents the conclusions of the work made by the join team.

Manuscript received September 10, 2010. This work was supported in part by the Colombia government under the COLCIENCIAS projects Agentes Coperativos.

E. Gonzalez, A.S. Miranda, M. Manrique, C. Otalora, J.S. Figueredo and R. Puerta are with the Pontificia Universidad Javeriana, Bogotá, Colombia (e-mail: {egonzal, alvaro.miranda, smanriq, camilo.otalora, jfigueredo, rpuerta}@javeriana.edu.co).

F. De la Rosa, C.F. Rodríguez, J. Angel and Y. Arévalo are with the Universidad de los Andes, Bogotá, Colombia (e-mail: {fde, crodrigu, jul-ange, yf.arevalo2026}@ uniandes.edu.co).

2 Robot Hardware

2.1 Robot's Electronic

The electronic components used in the project were entirely conceived and developed at Universidad Javeriana. The main requirements concern the robot movement control, the wireless communication component and the sensor system. To fulfill those requirements, the team decided to work with an integrated IDE and a fully tested compiler. The physical layout of the project's electronic is defined by a logical structure in which the processing module is separated from the power module (figure 1). The first module corresponds to the brain of the robot per se, where the microprocessor unit, a regulated voltage source, several communication ports, and some peripherals are located. The regulated voltage source provides the electronic card with the ability to be plugged directly to the batteries of the robot.

As the heart of the processing module, a dsPIC 30F6010A, has been selected. Some of the characteristics that made it a very suitable choice for the project are:

- The availability of an RTOS, which is well suited to implement an event-oriented real time control.
- An efficient C compiler is available, which reduces the development time.
- This device has enough peripherals and memory to satisfy the requirements of the project, and enough space in memory to handle future expansions.
- The compiler includes some libraries for signal processing, giving the ability to embed in the robot some sensor information processing.

The second electronic card incorporates the power module used to control the robot's motors. Each motor has a driver and is controlled by a PWM signal generated at the processing module of the robot. Both modules were designed with the same dimensions, to assemble them together, as seen in Figure 1. This arrangement allows placing in a small footprint all the electronics of the robot.

The embedded software is based on the Free RTOS operating system, which is the heart of the robot control. The considerations for the use of an RTOS, instead of a more simpler structural schema, responds to the idea of an scalable system that provides, in the future, the capability to expand the robot's abilities without complex adaptations of the embedded software. In addition, the current software was written in terms of data types and structures defined by an intermediate layer of the used RTOS.

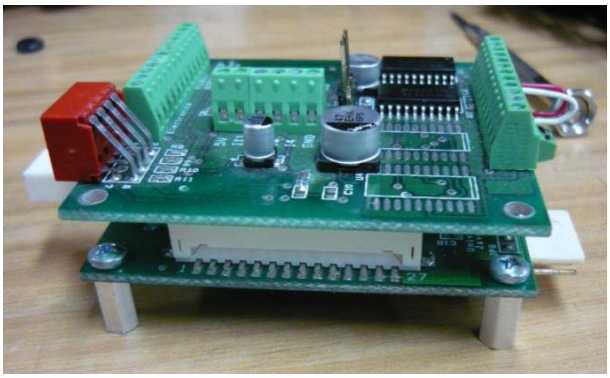

Fig. 1. Robot´s electronic control board assembled along the power board.

The hardware portability of the current software is increased by this approach, and makes it the most important advantage over other structural software approaches that were available for the project. Even though the learning curve of an RTOS is steeper than other options with simpler structural approaches to write embedded code, the initial time price paid in the training of an engineer is rapidly recovered by the adaptability that such approach generates towards the development of new tasks and software for the project.

2.2 Robot's Mechanical Structure

The robots are three wheeled holonomic platforms driven by DC motors. Energy is provided by AA batteries. They also have one dribbling mechanism based on a roller and a shooting mechanism based on a solenoid.
The base and the top of the structure is formed by two stainless steel sheets, separated by three columns (C folded SS sheets). This structure generates room for the motor gears, the dribbling, and the shooting devices. Each motor is attached to a column and has a single Swedish wheel. The motor gears (HSIANG HN-GH12-1634TR) give 190 rpm at 10 V, providing the robot with a maximum advance speed of 0.45 m/s.
The shooting mechanism [5] is essentially a solenoid driven by a voltage elevation circuit. The core of the solenoid is constructed in two materials: steel and aluminum. It is mounted below the top sheet at the center of the robot. The rod of the solenoid is extended by a L-shaped bar, needed for hit the ball. Figures 2 and 3 show a model of this mechanism and the CAD model shows the housing of the solenoid is showed in green.
Additionally, each robot has a dribbling mechanism. It is mounted in front of the robot, between two columns. It is composed by a housing frame for a small DC motor and a roller coupled by gears. This mechanism is activated when the ball is detected by a pair of infrared sensors located on each side of the frame. The roller is a segment of flexible hose made off rubber. The device provides a backspin to the ball at 3000 rpm.

Battery holders, each for four AA size batteries, are placed in the rear of the robot figure 2. The Batteries also provide weight balance to the robot. The robot weight is 1.8 kg, its total height is 135 mm and 180 mm of diameter.

Fig. 2. Test robots. Left, Dribbling system. Right Final robot

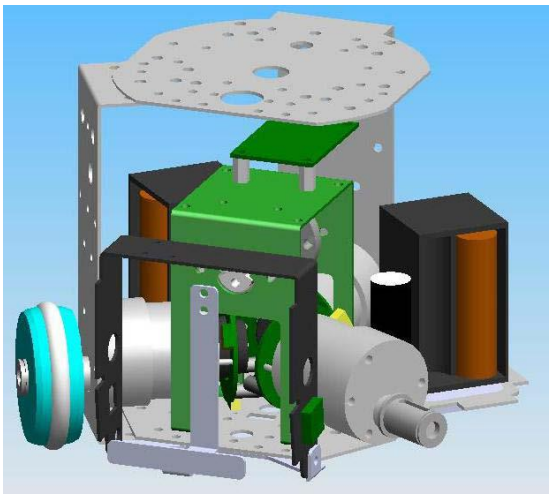

Fig. 3. CAD view of the model robot.

The electronic cards are placed under the top part of the metallic structure of the robot, in the green card beneath the top, figure 3.

3 Software Architecture

The team software architecture is based on a multi-agent layered architecture (*MRCC*) [3], implemented over the multi-agent platform BESA [4], where several cooperative actions are defined and compete to be executed. Each cooperative action

has an objective and its definition depends on different roles executed by the robot players.

3.1 MRCC Architecture

The conceptual model of Multi-Resolution Cooperation Control (MRCC) proposes that the cooperation control architecture that is formed by a hierarchy of layers, each layer deals with goals of different abstraction levels. For the case of robotic soccer, the MRCC that is composed by 4 layers appears as a complete solution. See Figure 4.

3.2 System Layer

It is responsible for the decisions of the highest degree of abstraction [3], where the global team goals are considered. It sends parametric influences to the lower layers; it is responsible for the general composition and the structure of the spatial regions of the formation level; it can also give more preference to some cooperative actions depending on the state of the team in relation to its goals. This layer can be seen as the coach of the team that takes strategic decisions.

3.3 Formation Layer

It manages a set of spatial regions in which actions take place; for each region a software agent aims to achieve a set of local goals. Robots can be assigned dynamically by a negotiation mechanism between regions as the goals of a region are not fulfilled. Inside a region, robots assume structural roles, which determine restrictions in the movement of a robot and also individual goals to accomplish.

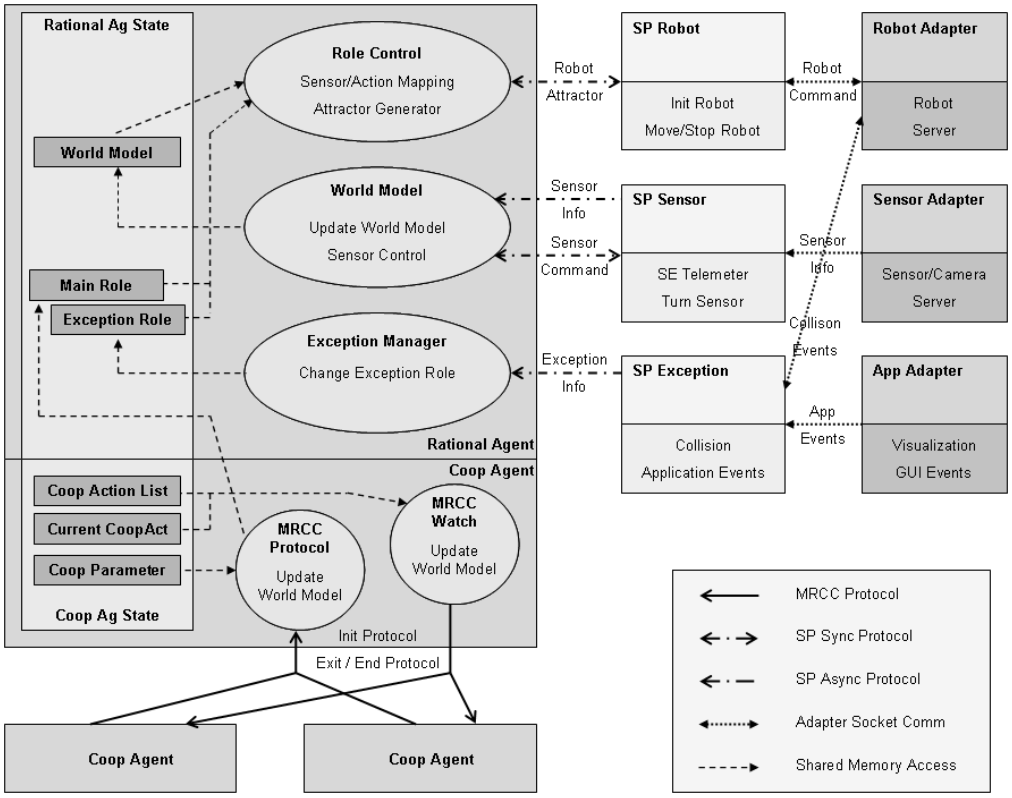

Fig. 4. Figure 4. MRCC Cooperative Agent Architecture

A structural role can be seen as a default role, assigned only when there are no opportunities to participate in a cooperative action.

3.4 Micro-social Layer

It is charged of the detection of opportunities to execute cooperative actions. Each robot includes a component that constantly monitors if the preconditions of any of the available cooperative actions are true. As an opportunity is detected, a negotiation protocol takes place between the concerned robots. If the negotiation succeeds, the action is executed by assigning cooperative roles to them. In figure 4, the internal architecture of a cooperative agent as proposed by the MRCC approach is shown.

3.5 Agent Layer

At this layer all the decisions taken at the formation and micro-social layers are transformed in actions performed by each individual agent according to the context

and role assigned. Thus, each cooperative action of the micro-social layer is carried out according to a number of roles that are assumed by a specific robot in a dynamic way. The appropriate role assumed by an agent not only depends on the opportunities, the location and its structural role within the system, but also depends on the characteristics and abilities of the robot. If an agent is not involved in the execution of a cooperative action, its actions are guided by the structural role assigned by the formation level.

3.6 Role Concept

A role is a set of goals, skills and resources that enable an agent to perform specific tasks within the framework of collaborative action [3]. Then, a role can be defined within our model as a set of elements (type of agent, responsibilities, skills, context and resources), assigned to a robot which allow to meet one or more individual goals, which are aimed to support the objectives of the team. In the 4-layers model, there are 2 different types of roles within the system, each one associated with a specific aspect of the model, among them are:

Structural Role: According to Kendal defined in [6] and [7], and over the formation level, a structuring role is a set of defined characteristics to meet specific needs within a system. Among these needs, in the case of the formation layer, compliance with responsibilities of a spatial region. For instance, a robot can get assigned the role of central defender within the defensive zone.

Cooperative Role: The cooperative role is defined as the set of guidelines to be assigned to the agent to accomplish with part of the goals involved in the achievement of a cooperative action. For instance, a robot can get assigned the role of receiver within a pass cooperative action.

3.7 Cooperative Actions

A cooperative action aims to fulfill a particular goal by a reduced group of agents by detecting when a cooperation opportunity is present.

Once an opportunity is detected, a cooperative strategy is applied, which involves specialized negotiation mechanisms and also action preemption control. In general, a cooperative strategy includes the following components: opportunity detection, negotiation protocol, preemption mechanism, monitoring and rating functions.

The general internal architecture that is used to implement MRCC based agents was described in a precedent paper [3]. The key elements of this architecture that allows implementing a cooperative action are the following.

Matching: it measures the similarity between the ideal and the current situations. A situation is defined from factors that should be considered in the cooperative action, such as object/agent positions, lengths, angles, etc.

Mapping: it makes all the decisions and calculations concerning the necessary actions that should be accomplished in order to reach the goal associated to a specific

role.

Parameters: these are a set of input values that allow specifying the characteristics of matching and mapping functions. Thus, these parameters allow reusing the code of a cooperative action to achieve different goals.

Role: in practice a role is composed of mapping and matching functions and their parameters. The conjunction of these elements is used by the opportunistic detection mechanism and the decision component associated to a role.

Cooperative action: it aims to define the sets of roles that must participate to accomplish an action that involves several agents; the role's matching is used to calculate if the cooperative action is suitable for the current situation or not.

Based on the above definitions, the process to create a new cooperative action includes the following steps:

- To define the system's goals, so all the cooperative actions that will be defined in the system has to contribute to reach these goals.
- To define the action's objective, this has to be aligned with the system's goals.
- To define all the roles needed in the action, as well as their matching and the mapping functions.
- To identify how to calculate the cooperative action's matching, using the matching of each of its associated roles.
- To try, if possible, to reuse precedent roles, matching and mapping functions that already exist.

3.8 Robot Soccer Actions

In our robot soccer team we have implemented a set of multi-agent and mono-agent cooperative actions under the MRCC architecture:

- *Direct pass to teammate* (multi-robot offensive action): the player controlling the ball makes a pass in direction of a teammate.
- *Indirect pass to goal* (multi-robot offensive action): the player controlling the ball makes a pass in direction of an estimated position where a teammate must arrive to shoot to the opponent's goal.
- *Delta formation* (multi-robot defensive action): the goalkeeper and two players define a triangle formation which follows the ball; the is to make obstruction to prevent shots to the goal.
- *Dribble* (mono-robot offensive action): the player controlling the ball moves it from one point to other.
- *Dribble and "self-pass"* (mono-robot offensive action): the player controlling the ball releases it after a predefined distance toward the near front; the idea is to try for regain possession of the ball in the next action.
- *Shoot to goal* (mono-robot offensive action): the player controlling the ball shoots it to the opponent's goal.
- *Approach to ball* (mono-robot defensive action): a player without the ball gets close to the ball.

4 Conclusion

As far as we know this is one of the few Robocup teams that work entirely following on a real multi-agent approach. Also, Bochica is the first F180 team that is completely built in Colombia, giving the participating universities the know-how to lead the process of developing more new Robocup teams.

5 Acknowledgment

This work is product of the Cooperative Agents projects: “Cooperación en Sistemas MultiAgentes Aplicada a Robótica Móvil” and “Robótica Cooperativa Basada en Agentes Heterogéneos Aplicada a Educación en Tecnología”. Projects financed by the government of Colombia through COLCIENCIAS with the participation of Universidad Javeriana, Universidad de los Andes, Maloka and Universidad del Norte. The authors thank the students and colleagues that have contributed to the development and testing of the MRCC architecture and Bochica platform.

6 References

[1] The Robocup Federation [Online], Available: http://www.robocup.org/

[2] Robocup Small Size Robot League [Online], Available: http://small-size.informatik.uni-bremen.de/

[3] Gonzalez, E.; Perez, A.; Cruz, J. & Bustacara, C. (2007, November). MRCC: A Multi-Resolution Cooperative Control Agent Architecture. *Proceedings of IEEE/WIC/ACM Intelligent Agent Technology (IAT)*, pp. 391 - 394, San Francisco – USA.

[4] González, E.; Avila, J. & Bustacara, C. (2003, June). BESA: Behavior-oriented, Event-driven and Social-based Agent Framework. *International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA*, pp. 1033-1039, 2003.

[5] Rojas Sandoval J., Diseño y construcción de los dispositivos de “Dribbling” y “Shooting” para un robot futbolista. Universidad de los Andes, Bogotá-Colombia, 2010.

[6] Kendall, E.A. (1998). Agent Roles and Aspects, In: Lecture Notes in Computer Science - Workshop on Aspect Oriented Programming – ECOOP 1998, Goos, G.; Hartmanis, J. & van Leeuwen, J., (Ed.), Vol. 1543, pp. 431-432, Springer, ISBN 978-3-540-65460-5.

[7] Kendall, E.A. (2000, April). Role Modeling for Agent System Analysis, Design, and Implementation. IEEE Concurrency, Vol. 8, No. 2, 2000, pp. 34-41, ISSN 1092-3063.