The Development of a Hybrid, Distributed Architecture for Multiagent Systems and Its Application in Robot Soccer

Claudio J. Biazus, Mauro Roisenberg

Abstract— **Several issues still need to be unraveled in the development of multiagent systems equipped with global vision, as in robot soccer leagues. Here, we underscore three of them (1) real-time constraints on recognition of scene objects; (2) acquisition of environment knowledge; and (3) distribution and allocation of control competencies shared between the repertoire of the agent's reactive behavior, and the central control entity's strategic and deliberative behavior. The objective of this article is to describe the implementation of a distributed and hybrid reactive-deliberative control architecture for a multiagent system, equipped with global vision camera and agent local sensor and cameras. This multiple agent system was developed for application in robot soccer. We present the digital image processing techniques applied, as well as the proposed control architecture aimed at satisfying the constraints of this kind of application.**

Index Terms— **hybrid and distributed control architecture, image processing, robot soccer.**

I. Introduction

During the 1990s, two scientific organizations were established to promote and develop Artificial Intelligence (AI) research applied to mobile intelligent robotics, namely, the RoboCup (Robot Soccer World Cup [7]), and the FIRA (Federation of International Robosoccer Association; [2]). In order to achieve their goals, both organizations chose the game of soccer as their primary application. The two associations organized world competitions of robot soccer, whose participating teams developed several new technologies, such as: autonomous agents project principles, intelligent control, image processing, multiagent collaboration, real-time reasoning and mobility, and acquisition and implementation of game strategies.

Most of these technologies are strongly coupled with a sensory apparatus [12]. In the case of a global vision device, such as a camera positioned with an overview shot of the field of play, the implementation strategies allow for little robot autonomy. In these cases, a technician operates a centralized control and deliberative system, which is coupled with the global vision device. The device, in turn, becomes the main source of input for the implementation of game strategies and control. Based on that input, the technician is in charge of controlling robot tasks. If global vision is not the system of choice, the sensory apparatus is embedded on the robots themselves. This is a local apparatus, which is made up of local cameras, and proximity and touch sensors. In this case, the robots are fully autonomous and team performance does not have to rely on a technician. The behaviors and strategies implemented are usually strongly reactive, and implemented directly in the robots; at any given moment, a new behavior may emerge without external interference [3]. Each implementation has advantages and disadvantages in terms of following time and response constraints, of processing sensory signals, and of generating strategies [8] as discussed below.

On the one hand, in traditional architectures of centralized robot soccer control, all information processing is concentrated in a central entity. In these centralized models, robot action is guided by information sent by this system. This type of architecture has the advantage of prompt recognition of the environment and definition of strategies. However, it does not allow for real-time reactive manifestations. On the other hand, models of local, or embedded, architecture implement autonomous control, behavior, and action in the system installed in the robot itself. This implementation is, thus, very efficient for avoiding obstacles and carrying the soccer ball. However, it has limitations in memory capacity, and in incomplete processing and view of the environment. In local architectures, the only way to obtain collaborative information is from cooperation between robots [8].

A recent trend in robotics is to try to make the best of what each strategy has to offer. This has lead to the development of hybrid (reactive x deliberative) and distributed architectures. Hybrid architectures allow for deliberative tasks, such as the generation of strategies, creation and manipulation of the environmental model, and control of robot global navigation, to be carried out by a central control entity. In turn, reactive tasks, which require rapid decisions, such as in avoiding obstacles, carrying the soccer ball, and deciding when to shoot, are made locally and autonomously by each robot in the team [6].

One of the biggest difficulties that presents with the development of a hybrid architecture is the precise definition of which competencies should be performed by the central control, and which should be performed by the local agents. This definition has to account for three

Departamento de Informática e Estatística (INE), Universidade Federal de Santa Catarina (UFSC), P.O. Box 476- 88040-900, Florianópolis, SC, Brazil, e-mail:cbiazus@inf.ufsc.br

Departamento de Informática e Estatística (INE), Universidade Federal de Santa Catarina (UFSC), P.O. Box 476- 88040-900, Florianópolis, SC, Brazil, e-mail:mauro@inf.ufsc.br

978-1-4244-1739-1/08/$25.00 © 2008 IEEE

objectives: (1) to minimize the size and amount of communication between the central entity and agents; (2) to ensure that agents can still perform satisfactorily in the environment, and perform in the absence of a central entity; and (3) to respond to present constraints in real robotic applications, such as real-time processing of information, noise, and incomplete or imprecise information extracted from a dynamic and non-uniform environment [4] and [5].

In this sense, the objective of this paper is to describe the development of a reactive x deliberative hybrid architecture that is distributed to autonomous, intelligent agents, and applied to robot soccer. The architecture aims at providing the necessary elements for agent recognition, for immediate reaction capabilities, and for strategic decision making.

The environment used for testing the hybrid architecture follows the Small Size Robot League (F-180) rules [13]. The following are the specifications for the SoccerBot Plus (EyeBot) used: MC68332 Motorola 32-bit microcontroller, equipped with 25MHz, 1 MB RAM, 512 Kb flash-ROM, LCD 128x64 pixels for low-resolution image presentation, serial and parallel ports, and digital and analog plug and socket system.

This paper is divided as follows: section 2 describes the hybrid architecture for multiagent systems; section 3, the experiment; section 4, results; and section 5 presents the conclusions and suggestions for future work.

II. Hybrid Architecture for a Multiagent System

Based on different architecture models put forth by different researchers [8], [9], [10] and [11], we propose a model that integrates centralized and embedded architectures.

In order to follow the specifications for hybrid and distributed architectures, it is necessary to achieve optimal exchange of messages between the central entity and the robots; in other words, messages should be short and transmitted only when necessary. At the same time, the system has to tolerate error in message communications. Robots should be able to continue to operate despite a failure in the communications system, and despite any delay between the receiving and transmitting of messages by the central entity and the ongoing gameplay–that is, the system has to adapt to a delay in processing the images captured by the global system and the transmission of a command message to the robot with instructions that pertain to gameplay action.

That the use of form independent, so much of the architecture embedded how much of the architecture centralized does not offer to better solution for the problem. The utilization of a responsible central entity by the taking of the strategic decisions although offer a solution for the problem, does not offer to better solution, due to the time of delay elapsed of the trial of capture of the images and the necessary time for act in the environment. Of that form, is necessary that have cooperation and negotiation between the systems, deliberative and reactive. In this sense, individual agents can ignore coming commands of the deliberative system when the system reactivate offers a better solution for the problem, by example, when the system of local vision indicate the position in that the ball is found. On the other hand, the agents also can ignore commands of the system reactive when the system of local vision do not indicate the presence of the ball and at the same time the deliberative system indicate that the sense of displacement of the agent should be an opposite sense.

The advantage of a hybrid architecture is optimal usage of available resources. Figure 1 illustrates the hybrid architecture for multiagent systems.

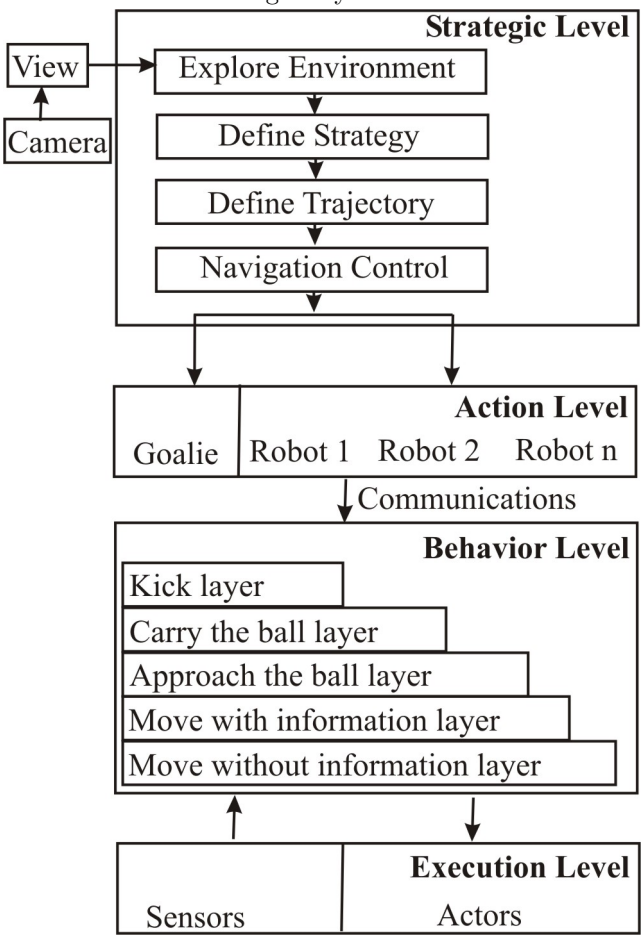

Fig. 1. The hybrid architecture for multiagent systems

As we be able to observe the proposed architecture is divided in two part. The first part of the architecture is composed by the module that is implemented in the central entity. This module possessed the levels: strategic and of action. On the other hand, to second part of the architecture is composed by a module that is implemented in the robots EyeBot (embedded). This module has the levels of behavior and of execution. Beyond that, have a module of communication that is responsible by link the two part of the architecture.

A. Central Entity Module

The central entity module includes a camera and the strategic and action levels. The strategic level includes four modules: explore environment, define strategy, define trajectory, and navigation control. The action level, in turn, includes actions attributed to the robots separately. The central entity and EyeBot module both follow a hierarchy, within which it is required that enough

pieces of information be made available for the explore environment, define strategy, define trajectory, and navigation control layers so that agent action can be determined.

B. Communications Module

The communications module enables the exchange of information between the central entity and the robots. The EyeBots are equipped with wireless communication units for sending and receiving messages. The units allow for communication in a pre-set frequency, for either sending EyeBot commands, or for exchanging information between the central entity and the EyeBots. The EyeBot operation system itself manages the communications, by means of a token ring virtual network. This communication system transmits at 9,600 bps and tolerates communication errors. The virtual network, called EyeNet, can connect to several EyeBots using the same communication station.

C. EyeBot Module

For the local agents, we decided to implement a strictly reactive architecture, based on Subsumption Architecture proposed by Brooks in the 1980s [1]. The EyeBot module includes a CCD camera, three sensors and five behaviors separated by layers. This reactive model has properties which are appropriate for gameplay. One such property is the ability of rapid execution, which also eliminates the need for planning behavior.

Behaviors were setup based on Subsumption. The position of behaviors in the layers was defined according to relevance for the architecture and, also, according to the necessity of a specific behavior in relation to the supply of information for the layers above it. The proposed architecture has five layers, namely move without information, move with information, approach the ball, carry the ball, and kick (Figure 1). EyeBot layers form a hierarchical structure, in which the upper layer follows instructions from lower layers, when a specific configuration of the sensors indicates appropriate action to be taken according to a determined situation. In this sense, the upper layers suppress lower-layer exits.

III. Implementation

A. Implementation of the central entity

The central entity system is responsible for exploring the environment and defining the robot team strategy. The central entity explores, plans, defines, and deliberates on strategies that will be executed by the robots. A strategy is determined when the system explores the environment and, together with the global view system, establishes the exact position of the ball and of the robots in the game. This particular system was implemented using Dev-C++ development tools, and OpenCV library.

Global View system: the system includes a camera installed with a birds-eye view of the soccer field. It is based on the information from the images captured by this camera that the central entity makes strategic and planning decisions. These decisions are aimed at assisting each agent's navigation system in executing the team's strategy. At the same time, the global view system provides the only data input for the image processing algorithm that is hosted by the central entity.

Explore Environment system: this system locates each object and its position in the environment. To locate an object, the system receives RGB format images, which it converts to HLS format. Next, the system divides the image into three channels: ChannelH, ChannelL, and ChannelS, where ChannelH represents the image tone, ChannelL, the luminosity level, and ChannelS, the saturation level. After the image is divided into three, cutoff limits are applied. These limits determine the intervals for the process of classification of a specific object. This process of determining the cutoff limits is carried out at the beginning of the game, with the process of calibrating the camera. The camera is calibrated by mouse-clicks on the color of the object being classified.

Define Strategy system: strategic behavior is determined by the process of planning. This behavior corresponds to a set of actions that may be carried out by each member of the team. The behaviors are divided into goalie behavior, which is preset for one robot specific robot in the team, and the defense and offense behaviors, which are alternated between the other robots in the team.

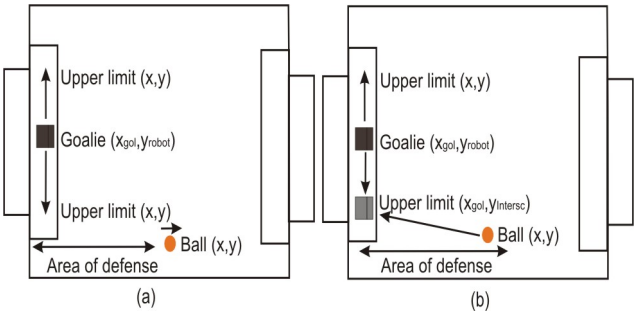

Fig. 2. Goalie strategy

The goalie behavior corresponds to positioning the robot at a strategic location, which is the center of the goal it defends. Figure 2 illustrates the strategic behavior for the goalie.

The algorithm implemented for goalie behavior consists of assessing the distance between the ball and the goal. If the distance between the goal and the ball is increasing (Figure 2a), the strategy is to return to or remain in the center of the goal. However, if the distance between the ball and the goal is decreasing (Figure 2b), the strategy is to position the goalie at the x-y coordinate which corresponds to the intersection point for the trajectory of the ball towards the goal. The objective is to stop the ball from crossing the intersection point. The behavior of the other robot players depend on the position of the robot in the environment. Figure 3 illustrates the offense and defense strategies.

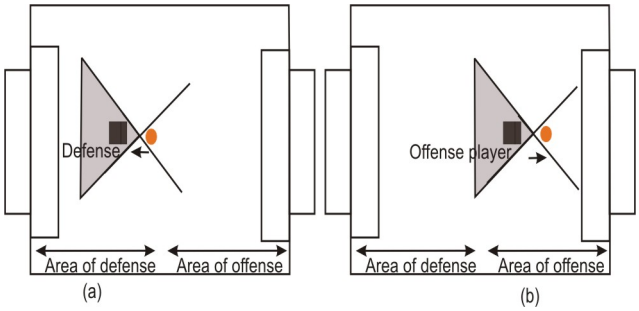

Fig. 3. Defense and Offense strategies

The algorithm for defensive behavior (Figure 3a) consists, basically, of stopping the robot from the opposing team to carry the soccer ball towards our goal. The defensive strategy is not dependent on the location of the robot. In other words, even if the robot is in the middle of the field, for example, it is able to take on defensive or offensive behavior. The defensive behavior takes into consideration the direction that the ball is moving.

The offensive behavior (Figure 3b), in turn, includes two actions: attack and take position. The attack action is implemented when the robot is behind the ball; in this sense, it consists simply of adjusting the robot's trajectory to carry the ball up to a point that is close to the opposing team's goal, at which point the kick mechanism is activated. The take position action is implemented whenever the robot is not located behind the ball. Thus, the robot must first position itself behind the ball and, then, the attack action can be applied. The definition of offensive behavior is based on the location of the ball and of the opposing team's goal.

Define Trajectory system: the implementation of this specific module depends on the behavior of the robot, that is, goalie, defense, and offense behaviors. It depends on the set of actions that are valid for each behavior, and on the target location, which is determined based on the location to where the robot needs to go relative to its present location.

This algorithm–considering that the information necessary for movement of the robots is collected by a set of sensors, and then processed by the layer Move Without Information in the EyeBot itself–we applied a method that aims solely at defining actions to be taken. The algorithm was implemented in this manner based on the characteristics of the architecture, which are to centralize the strategic and behavior attribution (action) levels in the central entity, and to allow for reactive behaviors by the robots themselves.

In this sense, the set of valid behaviors for the goalie are: stand still, move to a location to the right, and move to a location to the left. Hence, to position a robot at a specific point in the field, all that is necessary is a command message with the pertinent information for moving to that location. The set of valid behaviors for the robot players, in their turn, are: stop, move forward, turn right, and turn left. The trajectory of the players is determined as follows: define action and send information to the navigation system; the navigation system is responsible for the process of converting the action into a velocity vector, which is to be executed by the robots.

The Navigation Control system: this system is responsible for controlling the command messages with velocity vector information, which are sent to EyeBot agents. The navigation control system strategy applies local view system information to adjust the trajectory of the robots according to the dynamic position of the ball; that is, adjusts to changes in the position of the ball in relation to birds-eye global view information. The trajectory determined by the global view system determines the direction of the robots in the environment. The layer Move with Information, in turn, is responsible for fine-tuning the robot navigation control.

B. Implementation of the communications module

Communication is carried out by the sending of 8-bit packages in the form of commands, from the central entity to the robots. The 8-bit packages implement checksum for possible transmission errors. However, this specific means of communication is not entirely reliable: it does not implement a communications protocol that allows for retransmitting lost packages. Consequently, assessment of successful sending of a package is carried out one layer above the present layer.

The velocity vector messages (linear, angular) are sent from the central entity to the robots by 8-bit packages that are divided into two sets of 4-bit packages. One set of 4-bit packages is responsible for the left-side motor control (linear velocity), and the other, for the right-side motor control (angular velocity).

C. Implementation of the EyeBot module

The implementation of the initialization module for the EyeBot consists of a hardware component check for the robots, and of the integration of all the layers in the embedded software. The component check diagnoses radio, camera, motor, and kick mechanism problems. The process of integration of all the layers unifies all layers in the embedded system, and frees-up memory after each layer is successfully executed.

Implementation of the view system: the SoccerBot (EyeBot) was developed with the ability to recognize only three colors: orange, blue, and yellow. Orange is the color of the soccer ball, and blue and yellow are the colors of the goals. Color identification is implemented with color tone identification and storage of the corresponding value as a variable. Following assessment of the whole image, the algorithm establishes a histogram for the horizontal axis, and one for the vertical axis, and then intercalates them. This allows for obtaining an approximate value for the position of the objective.

The value obtained from the variable that stores the coordinates for the ball is adjusted based on the distance between the ball and the robot, both horizontally and vertically.

Information regarding the distance between the ball and the robot is obtained and updated every time an image is captured and analyzed. This information is readily made available for the layers Move with Information, Approach the Ball, Carry the Ball, and Kick.

Implementation of the sensor system: the SoccerBot Plus (EyeBot) model has three PSD (Position-Sensitive Detection) sensors, which are responsible for indicating the distance between an obstacle and the

robot. The sensor system is implemented to, first, determine an acceptable distance in order to detect all obstacles. This distance value depends on the position of the sensor on the EyeBot. The sensor positioned on the front of the robot has to be able to detect obstacles at a greater distance, in order to avoid collisions. The sensors positioned on the left and right sides of the robot, in turn, do not need to be able to detect obstacles at greater distances, since they assist only in avoiding obstacles during rotational movement of the robots.

The process of obstacle identification consists of predefining values that detect the presence or absence of an obstacle. Two values are set for each sensor, one lower value, indicating the presence of an obstacle, and one higher value, indicating the absence of an obstacle.

During the game, data on the distance between obstacles and the robot is updated in real-time. This information is also made available, at the same time, to the agent system.

Implementation of the agent system: the agent system is responsible for the control of the robots in the soccer game environment. The system controls two DC engines: one engine produces linear velocity, and the other, controls angular velocity. In this sense, robot movement during the game requires a command with a velocity vector, which is made up of the linear and the angular velocity values.

The system is implemented such that, firstly, it requires the linear and angular velocity values. The calculation of such values depends on the desired movement. In that sense, the values for movement of the robots are variable, that is, they may vary in situations where it is necessary to calculate the angle of approach to the ball, or to avoid obstacles.

The algorithm implemented calculates linear and angular velocity based on an objective. Next, the agents execute the commands based on these values, and this command is executed until a new command is received.

Implementation of the move without information system: this algorithm was developed to allow the robot to continue to move during the game based on information that is drawn from its sensors. It allows for the robot to move about during the game independently of having information on the location of the ball; all the while the robot is able to maintain a reactive status.

The algorithm implemented includes the possible situations the robot may face during a robot soccer game, and the respective actions for each possible situation. The respective actions are executed until the moment the soccer ball is detected by the system. Once the robot has input on the location of the ball, it will have an objective, and, thus, the layer Move with Information is activated.

Implementation of the move with information system: this layer was developed to trigger only after the robot has information on the location of the ball. Ball location information is obtained by the embedded sensors.

Once the system has information on the location of the ball, this layer is activated and the algorithm calculates the trajectory for the robot to move towards the ball. The calculation of the trajectory is based on the central location of the ball. However, once information on the present location of the ball is no longer available, the algorithm shifts back to the Move Without Information layer, which then determines the robot's trajectory.

Implementation of the approach the ball system: the approach the ball algorithm was developed to solve the difficulty that robots have when they go from the status of not having possession of the ball, to having possession of the ball. The approach has to be as smooth as possible, since a fast approach may result in a collision between the robot and the ball, thus pushing the ball away from the robot. This algorithm, in this sense, aims at making the robot move slower until it approaches and has control of the ball. Once the robot has approached the ball, it resumes normal navigation speed and, as the robot carries the ball, the layer Carry the Ball is activated and takes control over the robot.

Implementation of the carry the ball system: this layer determines robot action when the ball is being carried. It activates when control of the ball is being lost by the robot, in other words, when the ball is moving toward the outer extremities of the kicking mechanism of the robot. The objective of this layer is to control the ball when moving towards the ultimate objective, the opposing team's goal.

This algorithm simply assesses whether the ball is located at the center of the robot's kicking mechanism; if it is not at the center, it adjusts the trajectory of the robot to try to maintain control of the ball, at the center of the mechanism.

Implementation of the kick layer system: this layer controls the robot kicking mechanism. This mechanism is activated in two situations: one, when the robot is able to kick the ball towards the opposing team's goal; and two, when the robot tries to pass the ball to a robot teammate. The prerequisites for the kicking the ball towards the opposing goal are that the robot must be in an ideal kicking situation, which includes view of the opposing team's goal, and that the robot is in a position that is near the opposing goal. The passing of the ball to a teammate depends only on the dynamics of the game.

The decision that involves the movement of the mechanism of kick is attributed so much to the robot as regards central entity. For the case where the decision is of the central entity, is necessary send a message through the system of communication that corresponds to the responsible command by the movement of the mechanism of kick. This command is interpreted by the layer kick and performed by the agent that is responsible by the control of the mechanism of kick.

IV. Results

The results obtained showed a satisfactory performance. The central entity system was consistent in its ability to detect the localization of the ball and of the robots. The system processes digitized 640x480 pixel RGB images at a rate of 29.97 fps. The rate of processing was limited by the card applied in the system, a PCI PixelView PV-TV 304-p.

In relation the definition of moving of the robot involving the levels, strategic and of behavior, we observed that in function of the dynamic game, the result obtained for the moving of the robot regarding his direction in general converge for the same point. Of that form, the message of command sent by the strategic level simply is discarded by the robot. On the other hand, when the strategic levels and of behavior do not converge for a point in common, and being that the robot did not possess the information that corresponds to the location of the ball, he discards the messages sent by the level of behavior and performs the messages sent by the strategic level. In this sense, this model of architecture attends to the objectives that consist in the construction of a model of hybrid architecture and distributed acting in real time.

In relation to the results for the embedded system, we observed that the robots did behave as the implementation would foresee. The implementation of the layers and their respective status and actions were satisfactory to avoid robot collision during the game and to seek and approach the ball, with the objective of carrying it towards the opposing team's goal.

V. Conclusion and Future Work

A. Conclusions

In this paper, we described and assessed the development of a hybrid architecture for multiagent systems in robot soccer games. The architecture included an adapted global view system and local, embedded view sensors for the dynamic capture of information from the field of play. The processing of images was successfully carried out by the robots and the central entity.

The hybrid architecture implemented allowed for strategic processing of information, and for an improvement in the performance of the system. In this sense, both the global and local view systems benefited from the implementation. This architecture allowed for precise strategic actions and for improvement in the behavior and execution levels, in relation to having control of the environment.

The architecture proposed in this paper presents itself as an adequate solution for hybrid and distributed control of real-time, multiagent systems capable of playing robot soccer.

B. Future work

As a direction for future work, it is our objective to improve the strategic level of the system, to allow for better decision-making during the game; it is also our objective to implement a hardware prototype for robot soccer that is still being developed.

References

[1] R. Brooks, "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, 1986, pp 14-23.

[2] FIRA, Federation of International Robotsoccer Association, (http://www.fira.net/), 2007.

[3] F. A. Silva, and D. S. Strassmann, and J. G. F. Costa, and R. G. Lóssio, and G. Bittencourt, and M. Roisenberg, Estratégia para o Controle dos Robôs EyeBot do UFSC-Team: Categoria Small Size do Futebol de Robôs, 2006.

[4] J. Norbert, Decision marking and image processing robot soccer the challenge of the FIRA MIROSOF league, 2006.

[5] G. Novak, and R. Springer, An Introduction to a Vision System used for a MiroSOT Robot Soccer System, IEEE International Conference on Computational Cybernetics, 2004.

[6] R. R. Murphy, Introduction to AI Robotics, The MIT Press, London, England, 2000.

[7] RoboCup, Robot Soccer World Cup, The rules, (http://www.robocup.org), 2007.

[8] H. S. Shim, and P. Vadakkepat, A Hybrid Control Structure for Vision Based Soccer Robot System, International Journal Intelligent Automation and Soft Computing, 2000.

[9] H. Hu, and M. Brady, A Parallel Processing Architecture for Sensor-Based Control of Intelligent Mobile Robots, Robotics and Autonomous Systems 17, 2006, pp 235-257.

[10] Z. D. Wang, LOGUE: an architecture for task and behavior object transmission among multiple autonomous robots, Robotics and Autonomous Systems 44, 2003, pp 261-271.

[11] P. Vadakkepat, Evolution of fuzzy behaviors for multi-robotic system, and Autonomous Systems, 2006.

[12] W. Schwartz, Reconhecimento em tempo real de agentes autônomos em futebol de robôs, VI Simpósio Brasileiro de Automação Inteligente - SBAI, 2003.

[13] H. Kitano, RoboCup: A challenge Problem for AI. AI Magazine, v. 18, Spring, 1997.