ELSEVIER

Robotics and Autonomous Systems 29 (1999) 65–77

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

Cooperative behaviors in multi-robot systems through implicit communication

Enrico Pagello [a,b,*], Antonio D'Angelo [c], Federico Montesello [a], Francesco Garelli [a], Carlo Ferrari [a]

[a] *Electronics and Informatics Department, The University of Padua, Via Gradenigo 6a, 35131 Padua, Italy*

[b] *LADSEB-CNR, Corso Stati Uniti 4, 35100 Padua, Italy*

[c] *Mathematics and Computer Science Department, The University of Udine, Viale delle Scienze 206, 33100 Udine, Italy*

Abstract

We illustrate the *Cooperation through Implicit Communication* behavior-based approach used for developing the *PaSo-Team* (The University of *Pa*dua Simulated *So*ccer Robot *Team*), a multi-robot software system for soccer robot competitions promoted by the RoboCup Simulation League. The configuration of the environment, namely the robots' relative positions depending on both the global task and the game dynamics, provides a source of implicit information about the robots' intention to be involved in collective actions, making them able to cooperate implicitly. The soccer team performance can be tuned by triggering the arbitration module of any single robot to generate, as many as possible, suitable situations which hint to the team the action of scoring the goal. Some macroscopic parameters can be usefully introduced to evaluate the evolution of the whole multi-robot software system. © 1999 Elsevier Science B.V. All right reserved.

Keywords: Multi-robots; Robot coordination; Emergent behaviors; Implicit communication; RoboCup

1. Introduction

Building a group of robots that shows a cooperative behavior while executing a complex task is an interesting research challenge. As well-discussed in [3,6,8], many efforts have been made to control individual robots acting inside a dynamic environment, in order to make them act cooperatively with other robots in performing collective tasks. Starting from Brooks' pioneering approach [7], several scientists have devised innovative architectures, generally known as behavior-based architectures, giving rise to the new field of "Behavior-Based Robotics" [5]. Their strength is the ability to activate separate skills, each of which well-triggered on some specific properties and on the dynamics of the environment. But, their principal drawback is the difficulty to design an appropriate arbitration, that is the base for a good performance, in order to make the desired behavior emerging from the system. When we consider the same problem from the point of view of a colony of autonomous robots, we have to integrate communication and cooperation. Since cooperation is usually based on some form of communication, we may consider as such any form of observed behavior of other robots, and call it *implicit communication*, namely a non-intentional kind of communication.

Looking at a group of robots playing a soccer game, we learned that we should successfully deal with the

* Corresponding author. E-mail: epv@dei.unipd.it, pagello@ladseb.pd.cnr.it

0921-8890/99/$ – see front matter © 1999 Elsevier Science B.V. All rights reserved.
PII: S0921-8890(99)00039-1

reactive phase of the game, before attempting to explore the reasoning and planning phases. This approach stems from the observation that at each time of the game, *scoring a goal* can be easily recognized by each player to be an individual target, besides a global one. This allows the single robot to try, whenever possible, to score the goal directly, in contrast with a situation in which, before starting any useful action, a robot should acquaint with the world and build a map to be used later to find out how to fulfil its task. Thus, a planning phase is needed, because the robot has no way to know immediately what action to perform towards the achievement of the goal. In the simulated soccer game, instead, at each time of the game, each player knows what to do to realize its task, by using the environment information. This means that it is possible to consider every player to be able to bring to completion his own task.

Instead of considering deliberative planning capabilities, it is possible to focus on the lowest level of interaction among players and opponents. Then, designing a multi-robot system able to play soccer games, starting from a reactive approach, can be done from the simplified assumption that a number of low level reasoning capabilities can be endowed into the system as a set of basic behaviors, like, for example, the obstacle avoidance ability which detects the presence of other players, team-mates or opponents, the former cooperating and the latter interfering with robot's task of scoring a goal. Each robot's governor is implemented in a behavior-based fashion with an arbitration mechanism driven by the sensor data incoming from the environment, while the problem of coordination can be solved without using any form of reasoning about robot's intentions using the implicit communication paradigm [14].

When the evolution of a group of robots can be characterized by some kind of regularities, we can introduce *macroscopic* parameters, that depend on both the global goal of the multi-robot system and the environment dynamics. Their tuning can be driven by the environment knowledge shared by the robots. Neither effective intention nor negotiation take place, because they simply raise from the spatial robot displacement. They can be used as the implicit communication protocol.

We have experimentally investigated these problems by participating to soccer competitions organized by *RoboCup* [12], in the particular case when a team of robots play in the *Simulation League*, where each player is a software agent that can be displayed in 2D or in 3D modelling animation. The experimental software field offered by the Simulation League has motivated us to develop a reactive software architecture, where the behaviors are triggered by an arbitration module and their coordination arises mainly exploiting the implicit communication. Each software agent is provided with a set of states, partly referred to its individual acting (local flag) and partly referred to the whole team acting (global flags). A robot, modifying the environment, communicates not explicitly, but implicitly, its intention to be involved into the acting of another robot, by simply making the software agent aware through a pattern that can be easily recognized by looking at the environment. Each software agent becomes a simulator of a robot able to play soccer games, while the whole simulated robot team show an emergent cooperative capability through implicit communication.

The design of the whole software system is based on the following criteria:

- All interactions of each player with its team-mates and opponents are realized by low-level actions and processes. No high level reasoning is necessary to manage these interactions.
- The coordination among robots is realized through cooperative or competitive interactions respectively with team-mates and opponents.
- Arbitration is done separately over each robot, but cooperative coordination is obtained by ball exchanges between a pair of robots, by pressing and similar attitudes among several robots, and by dynamically changing some default positions over the whole robot team.

The paper is structured in the following way. In Section 2, we discuss how to use behavior-based architectures for designing multi-robot systems. In Section 3, we discuss how emergent intelligent behaviors can arise from implicit communication while playing simulated soccer games, and how the macroscopic parameters can give useful information. Finally, in Section 4, we give details about the *PaSo-Team* (the University of *Pa*dua Simulated *So*ccer Robot *Team*), our multi-robot system used in the RoboCup Simulation League competitions.

2. A behavior-based approach to multi-robot systems

Since a sound arbitration mechanism is the base for an appropriate performance of a behavior-based autonomous system [1], a number of innovative extensions over the subsumption architecture [7] have been proposed in the literature (see, for instance, [4,9]). These approaches are dominated by the concept of arbitration which results in an either spatial or temporal ordering of behaviors. The former causes the concurrent activation of a set of primitive reflexive behaviors, also referred to as *static arbitration*; the latter brings about a sequential activation of different sets of primitive reflexive behaviors, also referred to as *dynamic arbitration*. Several authors assume task decomposition into a number of parallel behaviors which compete for the control of the agent with a winner-take-all mechanism. Only one behavior dominates the agent at any time, although the dominant behavior can change frequently in rapid response to environmental sensing. In the subsumption approach each new behavior is hardwired on the top of a low level behavior featuring an inherent built-in coupling between each of the behaviors and providing their spatial and temporal ordering. An arbitration mechanism, based on a chemical machine, has been proposed in [10] to simulate activation/inhibition of behaviors as molecular reactions inside a solution. Different arrangements of behavior coupling result into different molecular combinations.

In the soccer robot case, a further difficulty arises, due to the simultaneous presence of several playing agents in the same environment. Coordination among robots may arise when single individuals plan complementary actions, and when some kind of prediction is available [24]. Because in simulated soccer robot application the inclusion of temporal ordering appears too problematic, we have implemented a static arbitration as a special purpose behavioral module where pre-processed sensor data are always channeled to discriminate a candidate skill to be enabled as a response to typical perceived patterns, simulating a kind of low level reasoning. Every time sensor data are directly channelled between the perception block and the selected behavior, this behavior is activated whereas the remaining ones are inhibited.

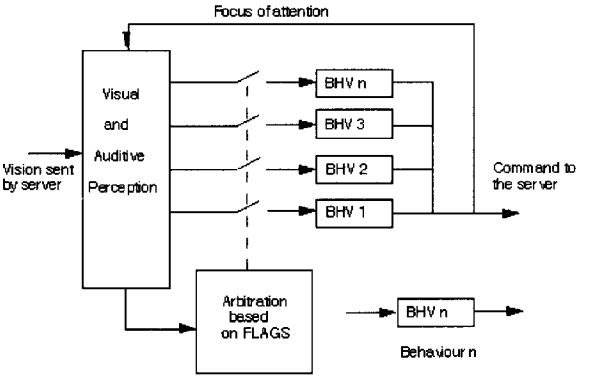


Fig. 1. Arbitrating the behaviors.

Let us explain with some details our approach. For every robot, a set of basic behaviors is supplied. Furthermore, an arbitration module is needed. This arbitration module can be hand-coded by intensive use of heuristic from soccer domain experience, as we did in our simulated application. In the resulting architecture, shown in Fig. 1, a collection of boolean values (flags), updated using information supplied from sensor data pre-processing, defines a coarse-grained global state of the agent which controls behavior switches as a rough inhibitor/activation mechanism. Moreover, the arbitration module is implemented in a way to allow the selected behavior to *focalize* agent attention and visual perception on some particular aspect of the world, as discussed in [20]. In fact each action performed by an agent acts as an indirect action on its own *sensorial apparatus*. This is quite similar to the case of a mobile robot that, during the activation of a *patrol* behavior with a wide visual range, recognizes a moving target and thereafter activates a *track* behavior with a narrower visual range to follow its target more efficiently. Then, to get *coordination* among the robots, we can usefully consider the different aspects of coordination, that is the *cooperation* and the *competition*. In simulated competitive game both the aspects must be considered.

2.1. Robot coordination

The problem of coordinating robot groups has been considered by robotics scientists from different points of view, encompassing the entire spectrum of AI approaches. Two kinds of cooperation among robots

have been considered, namely implicit and explicit communication, where we mean passing information respectively non-intentional and intentional. The first is realized by *looking* at the external behavior of the other agents, without any robot transmitting whatever information. The second is realized by sending voluntarily explicit coordination messages to the other agents. A fully distributed architecture based on explicit broadcast communication and active perception, that considers the cooperative side of coordination among heterogeneous mobile robots, with attention to fault tolerance, has been proposed in [18], where the cooperation is obtained by *observing* other robot's activity, *recognizing* the action of a certain robot, and making use of broadcast *communication*. Another architecture has been proposed in [21], built on a multiple physical robot system, with emphasis on cooperation, where the coordination via implicit communication is exploited only to perform low-level coordination, as *following, collision avoidance*, and the so-called *modest cooperation*, letting the higher-level cooperation to the explicit communication. Efficient cooperation among two robots has been obtained in [25] by a communication system based on an explicit negotiation protocol performed when an action partner is selected in order to reach a collective decision.

If the term *collective behavior* indicates any global behavior in a multi-robot system, then a *cooperative behavior* is a collective behavior that is characterized by cooperation. Cao et al. defined the *cooperative behavior design problem* as the problem of investigating "how given a group of robots, an environment and a task, cooperative behavior should arise" [8]. If we consider a behavior as an *emergent* one, if it can only be defined using descriptive categories which are not necessary to describe the behavior of the constituent components [22], then getting emergent cooperative behaviors in multi-robot systems becomes an interesting research goal. Thus, we have considered the case of the *implicit cooperation*, where a set of actions, performed by a single robot to achieve its own goal, affects the world and helps other robots to achieve their goal. As the form of communication, we considered only the observing of the *behavior* of other robots, that is, we rely on implicit communication only, which is referred to as *stigmergic* the behavior in the biological literature, in the meaning of "incitement to work by observing other people's work". This kind of communication is based on affecting the environment rather than on passing explicit messages. Then, an interesting example of emergent cooperative behavior arises from this form of communication [16].

To approach correctly the problem of controlling a multi-robot system, we have also to consider the interference among robots as a potential to inhibit, or limit, the behavior of each single robot in the case of resource competition. This potential increases as the number of robots increases, causing the degeneration of global performance, and forcing the use of social rules. Then, the focus in our architecture is on the separation between the individual and the social mind in the *brain* of each single robot. This allows the single robot to react differently to the environment depending on the strength of what we may call *social pressure*, that is the stimulation due to belonging to a group, namely the robot team. This idea can be viewed as an extension of Pfeifer's principle of the Value System [19], adding to it a social dimension, that allows a single robot to act positively in cooperation with other robots working in the same environment.

These two value systems, the *individual one* and the *social one*, are not separated within the single robot, but use the same sensorial apparatus. The difference is given by the different relevance that each value system gives to the same information. The individual value system reflects what is relevant to the individual point of view. The social and the individual value systems are continuously exchanging information between themselves, allowing the single robot to activate those behaviors with highest score points in both value systems. Thus, the emerging cooperative behavior, that may appear during a simulated robot game, can be considered a kind of *eusocial behavior*, that is a collective behavior due to the interaction of genetically determined individual behaviors, as introduced in [13].

2.2. Situatedness and embodiment

As it has been recently pointed out in the literature, *situatedness* and *embodiment* are the unavoidable features on which autonomous systems should be appropriately built. Implementing a real robot makes it hard to separate situatedness from embodiment. *Situatedness* defines systems acting in their environment by performing a transduction, receiving a stream of input

sensor signals from the environment and generating a stream of output actions. While, the term *embodiment* is referred to systems which are realized with a physical, rather than a virtual, structure, meaning that their components must be regarded as a part of the environment too. In this sense, a simulated autonomous robot cannot cope correctly with the unpredictable dynamics of the interaction with its environment because of the simplified assumptions made in the simulation itself. On the other hand, only physically implemented systems show their ultimate behavior. But, the understanding of how and why they work or don't work is a very difficult question from the designer point of view. The soccer robot game introduced by the RoboCup Simulation League overcomes this difficulty because each autonomous system can be implemented as a situated robot, but its embodiment can be neglected. The embodiment provided by the RoboCup soccer simulator is very simple. Each robot is shaped by a little circle of fixed radius with an orientation which establishes the front of the player and the direction of its motion. No impulse-based dynamics is supplied while a robot comes into contact with either the ball or other robots. In these cases the simulator implements only speed reduction and inversion (a simplified inelastic impact). Nevertheless, this simple embodiment allows testing robot communication and coordination capabilities in a very interesting way.

2.3. Thermodynamic parameters

The problem of controlling a team could be formalized by designing a set of structural constraints that drive the team to fulfil a desired behavior. In the most general cases, these constraint functions should involve a detailed description of the interaction dynamics of groups of players, of their intentions to cooperation, and under which conditions, and so on. In the case of a football game, however, we noticed the presence of some regular pattern in the evolving configurations of team-playing. Within these hypotheses we may suppose that a team should be perceived in a coarse-grained fashion zooming out the point of view of both an external observer and each component. So, we have postulated that when the evolution of a multi-robot system exhibits some kind of regularities inside its own environment, the state of the environment itself could be characterized by some macroscopic parameters in a same fashion as *thermodynamic* parameters can be introduced in physics.

The motivation for this approach is two-fold: from a designing point of view these parameters provide the programmer with a useful tool to tailor the implementation of the arbitration mechanism for each robot, and from the robot point of view they characterize the environment with properties that are easily detectable and measurable. Any such system has its own parameters which depend on both the kind of global goal the system has to pursue and the kind of dynamics of the environment. The choice of these parameters, however, should be driven from the type of information the robots should share if they were intelligent in the symbolic sense. So the environment is like a blackboard where robots read and write information about the global state of the distributed problem solver which drives the behavior of the multi-robot system. Macroscopic parameters should be chosen in such a way they are both *observable* and *controllable*. Then, they can be used as an *implicit negotiation protocol*, namely, a robot which wants to participate to a common global goal, informs the current attending robots of its *intention* by modifying one parameter or more.

3. Playing simulated soccer game by multi-robot systems

To apply multi-robot systems to play soccer successfully, the simulated robot team must have both basic and complex skills. Since the behavior of a single player, located in the simulated field, cannot be predicted exactly, due to sensor and actuator limitations and noise, primitive player's capabilities as *position predictors*, *shooting routines*, *turning_with_ball abilities*, and so on, are necessary. One possibility is to allow the players to learn low-level individual skills, as shooting to the goal, or intercept the moving ball, by using neural networks, or other ways. Then, the high-level skills can be learned by layering increasingly complex learned behaviors [23]. Individual basic skills can also be created by an analytical approach using the motion laws, as we did in our system, described in Section 4. Typical basic behaviors are *shoot_the_ball, chase_the_ball, avoid_the_opponent*, and so on. These behaviors can be designed mainly as reactive, since each behavior looks for sensorial data given by the

server in real time. Then, an *implicit communication* mechanism, like the one described in Section 2, applies usefully to show complex cooperative behaviors, like *defensive* and *offensive manoeuvres*, to move players to correct positions for attacking/defending phases. A collective behavior to force the adversary team in case of opponent's possession of the ball can also be introduced.

3.1. Voidance and pressing

As macroscopic parameters, two parameters are able to characterize the properties of interest when a robot team is playing a football game, *voidance*, and *pressing*. *Voidance* gives the free open space or the *wideness*, that can be used by the attacking team-mates. *Pressing* defines how near the team is to the opponent goal. They are both defined in terms of the free space around each player, which is summed over all team-mates and is weighted according to the distance from either the ball or the opponent goal. These parameters are defined in such a way that they focus on what happens around the ball and near the opponent goal, enhancing the fact that the ball is the attraction point of the game. The amount of value provided by the players whose position is more than a given distance from these crucial points in the field is neglectable.

Let us denote the position of a *team-mate* with $\langle x_i, y_i \rangle$, the position of an *opponent team-mate* with $\langle \xi_k, \eta_k \rangle$ whereas the *ball position* and the *goal position* are given, respectively, by $\langle x_B, y_B \rangle$ and $\langle x_G, y_G \rangle$. Then, we can evaluate the distance of the team-mate S_i from the ball-keeper d_i and the goal r_i used to weight the contribution of any team-mate taking part to a team action.

$$d_i \equiv \sqrt{(x_i - x_B)^2 + (y_i - y_B)^2},$$

$$r_i \equiv \sqrt{(x_i - x_G)^2 + (y_i - y_G)^2},$$

$$p_k \equiv \sqrt{(\xi_k - x_B)^2 + (\eta_k - y_B)^2}.$$

Please notice that $d_i \equiv c$, with c the kickable radius, when S_i is the ball keeper. In the same way, it is possible to consider the distance p_k of an opponent team-mate from the ball. The weight is defined so that we can suppose that all team-mates always participate

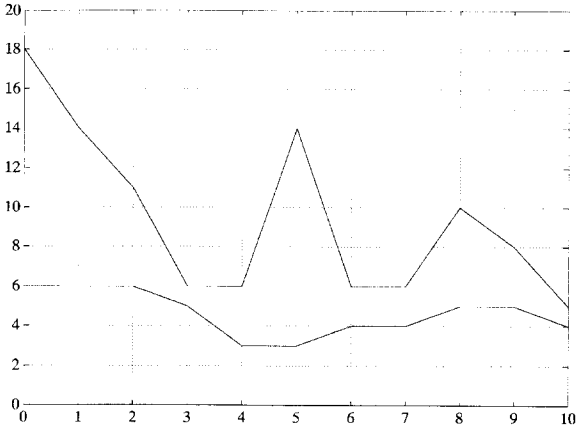


Fig. 2. Voidance plotting.

to team actions but their contribution is neglectable if they are a long away from the ball

$$w_i \equiv \frac{1}{d_i}, \qquad g_i \equiv \frac{1}{r_i}.$$

For every team-mate S_i we consider its *action free space* A_i,

$$A_i \equiv \min_k \left\{ \sqrt{(x_i - \xi_k)^2 + (y_i - \eta_k)^2} \right\},$$

which represents the distance between the team-mate S_i and the nearest opponent E_k.

Within these premises we can define the coarse-grained parameters V (voidance), and P (pressing), whose meanings have been previously discussed and whose motivation is that of formalizing the competion of two opponent teams for acquiring common resources.

$$\text{voidance } V \equiv \frac{\sum_{i=1}^{n} A_i w_i}{\sum_{i=1}^{n} w_i},$$

$$\text{pressing } P \equiv \frac{\sum_{i=1}^{n} A_i g_i}{\sum_{i=1}^{n} g_i}.$$

The value of V during an attack action is given in Fig. 2, which has been automatically generated by a simple evaluation program over a passing_ball action played between few robots. We can see a peak in the middle of the figure, that has been plotted in correspondance with the action phase when the ball has been gained by one of our unmarked team-mates. The

action of gaining the control over the ball has generated an increase of V which was previously declining because of the restriction of free areas around the ball itself generated by a pressing from the opponents.

The most interesting aspect of V is that it is able to take into account how much free space a team has to its disposal during an attacking action. Usually this value is very high when all the opponent team-mates are far away from the player holding the ball and his supporters. If the value decreases and approaches the value of the opponent team, it means that it is time to pass the ball to another team-mate. The choice of which supporter should receive the ball could be driven by an evaluation of which supporter can increase the value of V by receiving the ball. Usually it happens for the most unmarked player involved into the attack action.

A more precise evaluation of V can be easily obtained by using some formula that estimate the amount of free space in the game field, by evaluating the free playing space of each supporter, and their relative weighted distribution around the ball [11]. Some parameters, like, for example, a *supporting rate*, i.e. the current space percentage of field where the supporting team-mates are located, and so on, can be introduced. Thus, it is possible to simply manipulate the definition of V by supposing that the attacking team, moving towards the opponent goal, is composed by a player holding the ball and m supporters whose position is always kept inside a circle, centered on the ball, with a given radius R_a. The choice of this value is made in such a way that the team-mates out of the circle can be considered not actually involved on the current action. Within this hypothesis, V can be put into correspondence with the playing free space and the *kickable radius* for the player holding the ball, the playing free space and the distance from the ball, for all supporters, and a residual component given by the remaining team-mates.

Thus, generally the value of V for a team with $m + 1$ attacking players is given by a formula which strongly depends on the locations of the supporters around the team-mate holding the ball and trying to reach the opponent goal. The main parameters affecting V are, besides the free playing space for each supporter and their relative weighted distribution around the ball, the *supporting rate* and the *overbalance rate*. The former delimits the current space percentage of field where are located all the supporting team-mates, whereas the latter denotes the *counterattack* attitude of a team.

3.2. An emergent ball_passing behavior

Let us consider the situation of an attacking team with two players which are running with the ball towards the opponent goal. In this case the *voidance* parameter is meaningful only for these players. V can be computed with relation to the free playing space for the attacking player and its supporter, the kickable space and the distance of the supporter from the ball. The quality of the collective behavior of attacking the opponent team and bringing the ball to score a goal is increased when the value of V is locally maximized. The *unmarking* behavior shows this attitude, where *unmarking* is the behavior shown by a player without ball, when it moves quickly to a free area. Thus, during the attacking phase, it is important to produce the *unmarking* as soon as possible, to provide a hint for passing the ball.

We can see the emergence of a cooperative behavior, from the illustration of how players can manage ball triangulation actions through implicit communication. Triangulation, or non-explicit ball pass, arises mainly from the interaction between the single player and the environment, namely, in this case between our players and the opponents, exploiting the action dynamics as illustrated in the following (see Fig. 3):

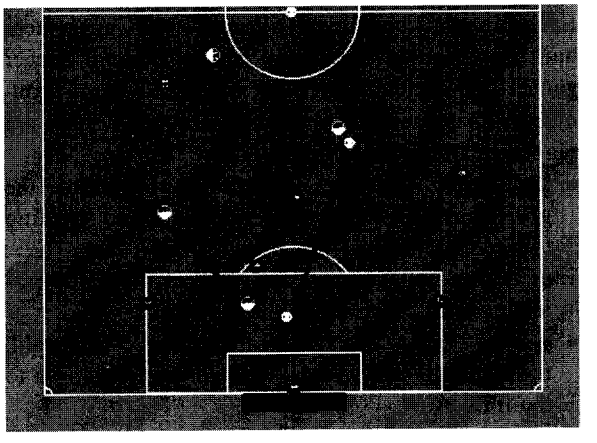


Fig. 3. Emergent pass behavior.

- In (1) a defender (from opponent team) chases the ball owned by a player (from our team). This is a hypothetical reasonable behavior to be assumed by the opponent player nearest to the ball.
- Since in this way the defender creates a free area in its rear, in (2) a forward player (from our team) finds a new free wide area and goes to take a new position.
- In (3) the coordinated action is accomplished: the ball owner see the forward occupying a good position and passes the ball. Without making any communication.

The *unmarking* ability of the team increases the value of the voidance parameter V. Thus, to activate implicit communication, it is sufficient that a player increases the value of V whereas the owner of the ball is monitoring the environment to detect this value diversion. The value of V can be continuously updated taking into account the new achieved situations. Thus, voidance can help in describing the previous situation, giving a good representation of the wideness in game field that can be used by the attacking team-mate.

4. The University of Padua Simulated Soccer Robot Team

PaSo-Team (the University of *Pa*dua Simulated *So*ccer Robot *Team*) is under development since two years [15], to test our approach to multi-robot systems by playing soccer games organized by the RoboCup Simulator League [12]. Building a soccer robot team to this aim requires to run concurrently *eleven Clients* each one able to interact with the *Soccer Server* [2]. Each Client is structured as a reactive software system built up on several behaviors specialized for soccer games that generates actions according to environmental changes. The following individual roles are played by our simulated robots:

- *goalkeeper*: it stays in the neighbourhood of the goal, chasing every approaching ball, and shooting to a free team-mate;
- *defender*: it stays in its own half of the field, protecting with other defenders their goal;
- *midfielder*: it stays around the centre of the field, receiving the ball from a defender and passing it to the forward;
- *wing*: it stays along the field's side lanes, receiving the ball from a defender and supporting the forward;
- *forward*: it stays in the opponents' half of the field, keeping unmarked, waiting for the ball and cooperating with other team-mates.

Developing Clients for RoboCup requires the reception and transmission of commands from and to the Soccer Server via a *socket*, that requires a continuous control of the state of the socket itself, because of the unpredictability of the exact instant of the *perceptions* sent by the Soccer Server. Each Client controls the socket looking at a SIG-IO, given by a C++ library, that is activated when messages are present in the receiving queue. A signal handler pops the message from the queue and sends it to a *parser* which extracts the suitable information. All communications are realized using the UDP/IP protocol. Every behavior, activated by a Client, generates the commands for the Soccer Server, that are stored in another queue, the *command list*, using the same protocol. This list is read at fixed interval times by a timer synchronised with the SIG-IO controlling the socket. In such a way, commands are sent to the server without losing a single command.

4.1. Sensor information

How can we perceive significant events from the messy and redundant information sent from the Soccer Server to each player? To extract significant information, we can build for each player, immediately after every received visual or auditive message, a data structure representing the world as the player saw it the last time, containing the last absolute seen *position*, *speed* and *orientation* information for each mobile object in the field. The RoboCup Simulation League soccer field is characterized by satisfactory local information and scarce global information. Then, the players can extract information only inside a limited zone. Summing up all significant information extracted from the environment gives a global evaluation to generate locally optimal actions. Thus, suitable parameters, like player's distance from opponents and from the ball, become crucial. Both absolute and relative information are used for acting in, and reacting to, the environment. For chasing the ball, turning with ball, and so on, we use relative information, because of the better accuracy in the estimate of values. Then, we can evaluate the trajectory of the moving objects (usually the ball) predicting the future positions of the ball itself

and trying to intercept or control it. After the parser has received and stored the sensorial information for each player, it is convenient to extract the *absolute data* from the relative ones. These data allow to reason about the absolute position, orientation, and speed, of the players and of all objects in the field. The estimate of the positions and orientation, using the relative information sent by the server, can be done with a geometrical approach, looking for every fixed object and making triangulation to recover the absolute ones.

4.2. Player's data structure

In soccer games the portion of field relevant to the actions is usually centred around the ball, causing an increase of relevance for the players near the ball and their own perceptions. Both the collective unmarking action, performed by team-mates without ball, and the individual skills of the ball's owner, are effectively relevant. Thus, to improve individual skills of the player, that is handling the ball, the player acquires the visual information from the environment under a representation structured in *Visual Fields* and *Visual Maps* [17].

The *Visual Field* is a data structure built for containing objects that represent relevant features of the environment. It is instantiated once for each player of our team. The typical objects that should be inserted in this structure are the *team-mates, the opponents, the ball, and the goal area*. Other geometrical shapes, like *the whole game field, the half sides of the field, the penalty areas, the circle in the middle, and the offside areas*, must be dynamically inserted in the Visual Field structure, following the evolution of the environmental dynamics. Thus, during the game, the Visual Field is constantly updated inserting, updating, and removing objects representing the vision frame of the considered player. For geometrical shapes, as the whole field or the offside area, the instantiation is straightforward, involving only the insertion of the object in the structure. The same operation is different in case the objects are players, team-mates or opponents. In fact, they don't have a predefined geometrical shape and so this operation must be done explicitly. We have assigned to each player a circle-shaped area, but with a parameterized diameter, depending on the team and the distance.

To deal with player's actions such as *obstacle avoidance, pass to mate, dribbling*, we have extended the Visual Fields with another data structure, the *Visual Map*, built by looking at the Visual Field instantiated for a given player at a given time (see Fig. 4). It is composed of 360 directions (degrees) toward which a player can decide to dash or to kick. A high value in the map indicates that the considered direction is a wrong way, while a zero or negative value suggests that the considered direction is a right way.

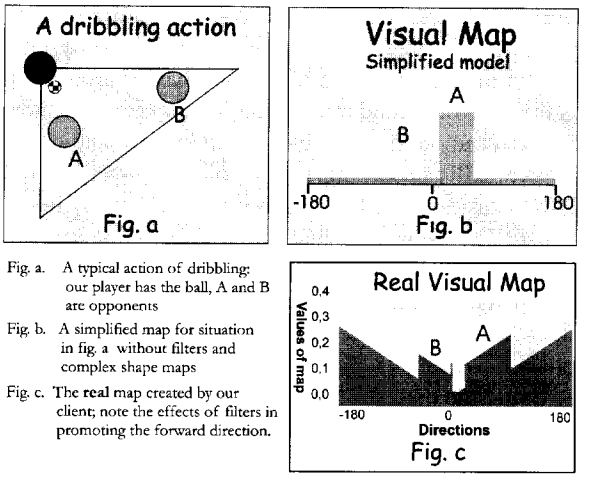


Fig. a. A typical action of dribbling: our player has the ball, A and B are opponents

Fig. b. A simplified map for situation in fig. a without filters and complex shape maps

Fig. c. The real map created by our client; note the effects of filters in promoting the forward direction.

Fig. 4. Visual Field and Visual Maps.

Suppose that our player has to dribble some opponents. At the beginning, the Client has a void Visual Map (all values are zero). Then, he asks for the objects in the Visual Field, that are dynamically updated during the perception phase, to create their mark in the Visual Map (i.e. an opponent adds some positive values in the elements of the Visual Map corresponding to wrong directions). After all objects in the field have modified the map, the Client applies some filters to promote a particular direction (the enemy goal port for example) or to create other useful effects. Finally, the player looks for the minimum in the Visual Map and goes in the suggested direction. The parameters used in the creation of the Visual Map depend on the particular action. For example, in obstacle avoiding, the opponents' radius is kept small, to represent the real radius of a player, while in dribbling it is kept larger, to include all the kickable area.

4.3. Arbitrating the behaviors

Since scoring a goal is eventually performed by a single player, the primary task for players is acquiring

individual skills. Beside these, a simulated robot team must also be able to show *cooperative skills*, as discussed in Section 3. An efficient behavior arbitration is the base of a good multi-robot system performance. The arbitration module activates individual and cooperative skills according to the roles assumed by the individual simulated robots to play consistently in the game. The behaviors that we use as robot skeletal structure were developed to capture the abilities of a real soccer player, adding abilities for communicating the state of the game, such as attacking or defending phases.

Using the data structures described above, a player is able to extract the *flags* used by the arbitration module, like *ball_kickable*, *ball_stolen*, *ball_near*, etc., that represent *individual states* for a single robot. Another flag, namely the *attack/defence flag*, is able to trigger the arbitration module. This flag is used as a descriptor of the *global state* of the system, and is used to identify the attacking or defending states of the team, namely the *attack/defence playmode*. It is set when a team-mate become owner of the ball.

The basic coordination mechanisms activated by the arbitration module usually involve two behaviors at a time. The coordination arise through the simultaneous activation of behavior pairs during the attack or the defence playmode. To realize it, the arbitration looks at the global flags representing the states of the whole teams and at the local flags related to the state of the single player. In the *attack playmode*, for example, such a coordination characterizes two players belonging to the same team, namely the *ball-holder* and a *potential receiver* of the ball, in the case that the *play-ball* behavior, for the player holding the ball, is simultaneously activated with the *unmarking* behavior, for the next ball-holder candidate player. One more coordination schema is possible on the whole team, during the game. It is realized changing dynamically the *default_pos* of every player in the team, accordingly with the ball position. The default_pos are data structure to which players make appeal when they move. Updating them drags all the players, according to their roles, to follow the ball in case of attack playmode, or to back gradually to the defence position in case of defence playmode.

In order to keep the system architecture more agile, the arbitration module has been strongly simplified by pruning the decision tree. Several exception handlers, like action schema and timeout-based decisions, have been cut off. Other simplifications have arisen from exploiting some features of the *environmental niche* inside which the players live, i.e. of the soccer field, like the "dynamic environment hypothesis". Such assumption is based on the fact that in the soccer field the players keep moving in a non-negligible way, even if they may be not always active during the game. In such a way the complexity of the arbitration module is lowered, by assuming that the environmental dynamics is sufficient to keep informed the players on the ball position. Then, the explicit action, of monitoring the ball, is avoided.

4.4. Active behaviors

To exploit the above approach several simple and complex behaviors have been actually implemented. We list some of them in the following.

Playing is an individual behavior. It provides a player with the commands able to realize the correct actions when it owns the ball, that is bringing the ball along a *free_from_enemies* direction, and deciding whether to pass it or not, by testing dangerous situations. If running with the ball (or passing it) becomes too dangerous, then the player throws away the ball in a free_from_enemies direction.

Chasing is an individual behavior. It forces the player to chase the ball. This may happen in two ways, named stamina-preserving and stamina-consuming. The former is invoked, when the ball is far, and the latter, when the ball is near.

Going_to_default_pos is a collective behavior. It moves players to the default position according to the invoked planning schema, depending on the ball position. This means that for an attacking team the player's default positions change with the ball position, so that the team itself is lined up in the right manner to perform a coordinated attack action.

Recovering is a collective behavior. It is activated in defence playmode. It provides the defenders, the wings and the midfielder with the skills of controlling and breaking the enemy's play. For wing and midfielder roles, the player moves dynamically between the ball and the enemies, to interdict the play. For defender roles, the outsiders dynamically keep a position between the enemy forwards and the nearest goalpost, to avoid a direct shoot to goal, while the

insiders keep symmetrically a position that allows to control an approaching central enemy, avoiding field areas to be not defended.

Unmarking is a collective behavior. It is activated in attack playmode. It moves dynamically the players without ball toward the most free area, preparing them to receive a pass. It is based on a random search algorithm that looks for a minimum of the degree of freedom of the player in the field. It can be put in correspondence with the estimation of voidance introduced in the previous sections.

5. Conclusions

We assumed that high reactivity is the main characteristic of soccer games. Recognizing configuration patterns of the dynamically changing environment makes possible an efficient arbitration over the behaviors set of simulated soccer players. Thus, the problem of coordination among a group of robots playing such a game is solved through implicit communication, without using any form of reasoning about robot's intentions.

Since the evolution of multi-robot systems may exhibit some kind of regularities, inside its own environment, we proposed to characterize the model of the environment through coarse-grained dynamic properties which can be easily detected by single individuals of a multi-robot system to coordinate their actions to get a common target. Thus, the state of the environment itself can be characterized by some macroscopic parameters in a same fashion as thermodynamic parameters can be introduced in physics. These parameters can also be modified by a single robot to hint the other robots for a possible coordination, so that they can be usefully employed by a group of robots as a mechanism of implicit communication.

This approach can be applied to develop cooperative multi-robot systems, and it has been extensively tested by using *PaSo-Team* (the University of *Pa*dua Simulated *So*ccer Robot *Team*), our simulated multi-robot software system that has participated in soccer robot competitions promoted by the RoboCup Simulation League. *PaSo-Team* software architecture allows to exploit the interaction dynamics among the robots and the environment, by generating, whenever possible, an emergent cooperative behavior. Experimental results were encouraging, since the whole robot team showed harmonic performance playing against other teams.

Acknowledgements

We like to thank the students of the Padua University Engineering School that contributed to the development of the PaSo-Team Software System. In particular we acknowledge the work of F. Bidinotto, A. Bissacco, F. Candon, P. Chioetto, M. DalSanto, W. Frasson, S. Griggio, and A.F. Grisotto. Financial support has been provided by both CNR, and MURST. A particular thank is due to the Industrial Firm Calearo S.R.L., a car aerials and cables manufacturer, located in Isola, Vicenza (Italy), that has financially supported the participation of our research group to RoboCup Competitions, both in Nagoya (1997) and Paris (1998).

References

[1] T.L. Anderson, M. Donath, Animal behaviour as a paradigm for developing robot autonomy, in: P. Maes (Ed.), Designing Autonomous Agents, MIT Press, Cambridge, MA, 1990, pp. 145–168.
[2] I. Noda, H. Matsubara, K. Hiraki, I. Frank, Soccer Server: A tool for research on multiagent systems, Applied Artificial Intelligence 12 (1998) 233–250.
[3] T. Arai, J. Ota, Let us work together – Task planning of multiple mobile robots, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'96), Osaka, Japan, 1996, pp. 298–303.
[4] R.C. Arkin, Motor schema based navigation for a mobile robot: An approach to programming by behavior, in: Proceedings of IEEE/ICRA Conference on Robotics and Automation, Raleigh, NC, 1987, pp. 264–271.
[5] R.C. Arkin, Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.
[6] H. Asama, Elemental technologies for collective robotics Journal of Robotics and Mechatronics 8 (5) (1996).
[7] R. Brooks, A layered intelligent control system for a mobile robot, IEEE Journal on Robotics and Automation RA-2, pp. 14–23.
[8] Y. Cao, Cooperative mobile robotics: Antecendents and directions, in: R.C. Arkin, G.A. Bekey (Eds.), Autonomous Robots 4 (1) (1997) 7–27. Special Issue on Robot Colonies.
[9] J.H. Connell, A behaviour-based arm controller, IEEE Transactions on Robotics and Automation 5 (6) (1989) 784–791.
[10] A. D'Angelo, Using a chemical metaphor to implement autonomous systems, in: Topics in Artificial Intelligence,

Lecture Notes in Artificial Intelligence, Vol. 992, Springer, Berlin, 1995, pp. 315–322.

[11] A. D'Angelo, F. Montesello, E. Pagello, A design paradigm for emerging cooperative behaviors in multiagent systems, in: S. Badaloni, C. Minnaja (Eds.), Proceedings of VI Convention of AI*IA, Edizioni Progetto, Padova, 1998.

[12] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, H. Matsubara, RoboCup. A Challenge Problem for AI, AI Magazine 18 (1) (1997) 73–85.

[13] D. McFarland, Towards robot cooperation, in: From Animals to Animats 3, Simulation of Adaptive Behavior (SAB-94), Brighton, 1994, pp. 440–444.

[14] F. Montesello, A. D'Angelo, C. Ferrari, E. Pagello, Implicit coordination in a multi-agent system using a behavior-based approach, in: T. Lüeth, R. Dillmann, P. Dario, H. Worm (Eds.), Distributed Autonomous Systems 3, Springer, Berlin, 1998, pp. 351–360.

[15] E. Pagello, F. Montesello, A. D'Angelo, C. Ferrari, A reactive architecture for RoboCup competition, in: H. Kitano (Ed.) RoboCup-97: Robot Soccer World Cup I, Lecture Notes on Artificial Intelligence, Vol. 1395, Springer, Berlin, 1998, pp. 434–442.

[16] E. Pagello, A. D'Angelo, F. Montesello, C. Ferrari, Emergent cooperative behavior for multirobot systems, in: Y. Kakazu, M. Wada, T. Sato (Eds.), Intelligent Autonomous Systems, IOS Press, Amsterdam, 1998, pp. 45–52.

[17] E. Pagello, F. Montesello, F. Garelli, F. Candon, P. Chioetto, S. Griggio, Getting global performance through local information in PaSo-Team '98, in: M. Asada (Ed.), RoboCup-98: Robot Soccer World Cup II, Lecture Notes on Artificial Intelligence, Springer, Berlin, 1999.

[18] L.E. Parker, ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-94), Munich, Germany, 1994, pp. 776–783.

[19] R. Pfeifer, Building fungus eaters: Design principles of autonomous agents, in: P. Maes, M. Mataric, J. Meyer, J. Pollack, S. Wilson (Eds.), From Animals to Animats 4, Simulation of Adaptive Behaviour (SAB-96), MIT Press, Cambridge, MA, 1996, pp. 3–12.

[20] S.J. Rosenschein, L.P. Kaelbling, A situated view of representation and control, Artificial Intelligence 73 (1–2) (1995) 149–173.

[21] M. Rude, T. Rupp, K. Matsumoto, S. Sutedjo, S. Yuta, Iron: An inter robot network and three examples on multiple mobile robots' motion coordination, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97), Grenoble, France 1997, pp. 1437–1444.

[22] L. Steels, The artificial life roots of artificial intelligence, Artificial Life Journal 1 (1) (1993).

[23] P. Stone, M. Veloso, A layered approach to learning client behaviours in the RoboCup Soccer Server, Journal of Applied Artificial Intelligence 12 (2–3) (1998) 165–188.

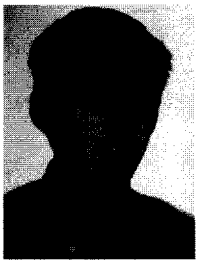
[24] F. von Martial, Coordinating plans of autonomous agents, in: Lecture Notes in Artificial Intelligence, Vol. 610, Springer, Berlin, 1992.

[25] K. Yokota, K. Ozaki, A. Matsumoto, K. Kawabata, H. Kaetsu, A. Asama, Modeling environment and task for cooperative team play, in: T. Lüeth, R. Dillmann, P. Dario, H. Worm (Eds.), Distributed Autonomous Systems 3, Springer, Berlin, 1998, pp. 361–370.

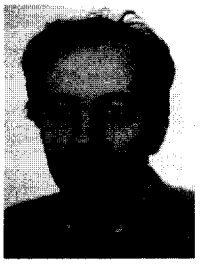
Enrico Pagello is an Associate Professor of Computer Science, since 1983, at the School of Engineering of the University of Padua, where he belongs to the Department of Electronics and Informatics (DEI). He has also a joint appointment with the Institute LADSEB of CNR as a part-time Senior Scientist. He has been a Visiting Scholar at the AI Laboratory of Stanford University in 1977–1978. He is the Chairman of both the Working Group on Robotics (GLR) the Italian Association for Artificial Intelligence (AIIA) and the Italian RoboCup Scientific Committee. He is a member of the Council Board of the Italian Association of Robotics (SIRI). His current research interests are multirobot systems, motion planning, assembly planning and soccer robotics.

Antonio D'Angelo is an Assistant Professor of Computer Science at the University of Udine. He received an M.S. on Electrical Engineering from the University of Padua in 1981 and since 1984 he has been working at the Laboratory of Artificial Intelligence and Robotics at the Department of Mathematics and Computer Science at the University of Udine. His current research interests cover multiagent autonomous system coordination, behavior-based planning and control including complex system models for autonomous robots.

Federico Montesello received his Master degree from the University of Padua in 1995. He is now running towards his Ph.D. degree within the Intelligent Robotic Systems Group of the Department of Electronics and Informatics, of the University of Padua. His current research interests are multiple autonomous robotic systems, self-organizing systems, complex dynamical sytems.



Francesco Garelli is pursuing his Master degree in Computer Engineering at the University of Padua, within the Intelligent Robotic Systems Group of the Department of Electronics and Informatics. His current research interests include multiagent systems, autonomous robotic systems and client–server architectures.



Carlo Ferrari was born in 1960 in Genua (Italy). He received the Laurea degree from the University of Genua in 1985, and the doctoral degree from the University of Padua in 1992. He was visiting the University of California at Berkeley from 1990 to 1991. Since 1992 he is an Assistant Professor in Computer Science at the School of Engineering of the University of Padua, within the Department of Electronics and Informatics (DEI). His research interests are motion planning for multiple mobile robot system, assembly planning, RISC robotics, dynamic simulation.