Middle Robots League, Team *ISocRob*, pages 150–159
http://www.ep.liu.se/ea/cis/1999/006/24/

ISocRob — Intelligent Society of Robots

ISocRob

Rodrigo Ventura, Pedro Aparicio, Pedro Lima, Luis Custodio

RV, PA, PL, LC: Instituto de Sistemas e Robotica - Instituto Superior Tecnico

Abstract. *The SocRob project was born as a challenge for multidisciplinary research on broad and generic approaches for the design of a cooperative robot society, involving Control, Robotics and Artificial Intelligence researchers. A case study on Robotic Soccer played by a team of 3 robots has started two years ago and was first tested last year during* RoboCup98. *This experience has clearly revealed that the robotic soccer environment is a sufficiently rich, complex and dynamic testbed to study new methodologies both on Robotics and Artificial Intelligence. Therefore, the SocRob team is currently working on the SocRob project improvements and intends to compete at the World Cup of Robotic Soccer,* RoboCup99*, held in Stockholm, Sweden, in the middle-size league. In this paper the basic aspects of last year implementation as well as the improvements made meanwhile are briefly recalled and presented. Naturally, a special emphasis is given here to the novel solutions proposed for this year implementation, the results obtained and the expected future developments.*

1 Introduction

The Artificial Intelligence and the Intelligent Control groups of the ISR/IST have started almost two years ago a joint project on Cooperative Robotics, denominated SocRob, to foster research on methodologies for the definition of functional, hardware and software architectures to support intelligent autonomous behavior and evaluate performance of a group of *real* robots, either as a society and as individuals.

The area of research concerned with the study and development of multi-agent systems able to perform in complex and dynamic environments, where cooperation is the keyword, is becoming more popular everyday. Applications as virtual training [15], interactive education [11], entertainment [10], and collective robotics [2], to name but a few, are good examples of the kind of complex and dynamic domains, in which the creation of multi-agent systems faces an interesting challenge. In fact, the uncertainty usually associated with these aplications, and the corresponding environments, raises problems that could hardly be handled using pre-defined coordination plans for a set of independent agents [18]. For instance, agents "living" in those environments have to deal with unexpected situations, incomplete information about the world and the other agents, differing opinions or perspectives among the agents, and the impossibility to fulfill their tasks. Most of the research work on this area has been carried out by people working on Distributed Artificial Intelligence (DAI) [14]. Moreover, most of the pratical applications known are developed using virtual environments and virtual agents, in order to focus research on "high-level" issues, such as communication protocols, teamwork models, individual and team task planning, distributed knowledge representation, adequate programming paradigms and reasoning about others' beliefs.

The utilization of real robotic agents to perform on real environments, for instance, a robotic soccer game, raises several new questions and perspectives that turn the development of a multi-agent system a much more difficult and challenging problem [19]. Given the past experience of both groups regarding robot development [9] and DAI [17], it was decided to join forces to build a team of initially three and now four robots in order to participate in the RoboCup initiative (the **IS**oc**R**ob Team). Figure 1 shows three elements of ISocRob team.

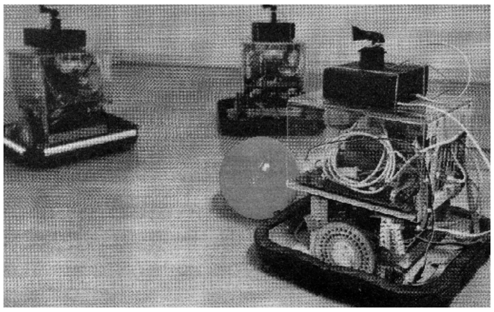

Figure 1: Three ISocRob team members.

The robots were developed from scratch, so that both conceptual and implementation issues were considered. Last year, the ISocRob Team efforts were devoted more to aspects related with building the robots' mechanical structure and hardware design and implementation. Given the results obtained in the *RoboCup98* and some adjustments and improvements made mainly on hardware components (e.g., new robot wheels, a self-localization system, a friendly man-machine interface with the robots, a kicker device, and proximity sensors based on infrared (IR) technology), the team decided to focus its research on conceptual aspects of the SocRob project. As such, special attention was given to:

teamwork models: what kind of structure a multi-agent system control architecture should have in order to explicitly integrate both teamwork and individual tasks;

cooperation-oriented communication issues: what type of information must be shared and how to distribute that information among the agents;

programming languages: which features a programming language should have in order to be adequate for developing and implementing multi-agent systems and allowing explicitly representation and reasoning about teamwork.

This paper is organized as follows: in Section 2 the details of each robot Hardware and Software Architectures are briefly described. The Functional Architecture, presented in Section 3, wraps up the whole picture, relating conceptual issues to the two physical architectures explained in the previous section. In Section 4 the project work concerning the development of a programming language suitable for multi-agent systems is presented. Finally, conclusions and foreseen future work are presented in Section 5.

2 Hardware and Software Description

To interact with the real world, a mobile robot must have the ability to sense the environment, process that information and actuate on the world. Each robot hardware is divided in four main blocks: sensors, main processing unit, actuators and communications. Currently, from the hardware architecture standpoint, the population is composed of homogeneous mobile robots. Figure 2 depicts a block diagram of the hardware architecture of each robot.

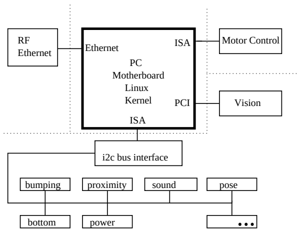

Figure 2: Hardware Architecture.

2.1 The Processing Unit

Each robot has on-board a PC motherboard with a network adaptor, a video adaptor, a motor control board and interface boards for the sensors. The main processor is an AMD K6, running at 200MHz. The system has 16Mb of RAM and a 1.2Gb hard drive.

2.2 Sensorial Systems

The sensors available in each robot are those mentioned in last year's team report [3]. The pose, bumping and proximity sensors are being implemented and will be detailed here:

Pose sensor

Depending on the type of application involved, each robot of the society needs to regularly update its current pose (position and orientation) with respect to a reference frame (e.g., located in the field center). This may be accomplished based on the triangulation principle using for instance a *convex mirror for full scene image* system based on a vision camera and a mirror with a special geometry. Since the RoboCup environment has available a sufficient number of visual landmarks, the SocRob project team decided to experiment the "mirror" solution. The idea is to allow robots, using only one vision camera, to acquire images from the mirror, appropriately positioned above the robot, in order to obtain a global view of the environment. If images are sufficiently broad to include three different and static beacons (e.g., goal plus two field corners), robots may apply the triangulation principle to determine their position. A key aspect for the performance of this type of self-localization systems is image definition and quality. This is directly influenced by the mirror geometry. For instance, we would like robots to unambiguously see both near objects and as far as possible on a RoboCup field. So, one of our main concerns was the choice of mirror geometry. Based on some theoretical studies available in the literature, three basic geometries have been identified as possible options: conical, spherical and hyperboloidal. In order to assess image quality obtained for each geometry, some experiences and simulations were performed. The results have shown that none of the former geometries is really satisfactory. Therefore, we decided to investigate which mirror geometry can give the results we consider adequate for this application. This work is now underway and we expect to have some final results by the time of the RoboCup competition.

Bumping Sensors

Bump sensors are the last resort for a mobile robot in the presence of eminent danger. They detect the collision of the robot with an obstacle in the environment. In the soccer application, they can also be used to sense contact with the ball. They are made of microswitches, arranged in a serial connection, divided in 4 sets of 2 microswitches each.

Proximity Sensor

Proximity sensors are based on IR technology. They allow the measure of an analog value proportional to the object distance (depending on the material reflectance). The five emitter/receiver pairs are equidistantly located around the vehicle, pointing outside, in order to detect objects in the near vicinity. The typical range of those sensors goes from 20 cm to 1.5 m.

2.3 The Actuators

Drive

Each robot has a differential drive kinematic configuration. This implies that it has two independent (DC) motors, one for each wheel. The robot speed and heading are set by independently controlling the wheels speed. For more details, see [3].

Kicker

The kicking ability enables soccer players to move the ball into places that otherwise would not be accessible. The kicker device is divided in two main parts: electronic and mechanical structures.

The kicker electronics is composed of a micro-controller, an IR beam circuit and a power actuator. The micro-controller runs the control program and generates a signal modulation to be used in the IR beam. This signal consists of a square wave, rated 40kHz that is fed to the amplifier powering the IR leds. The 40KHz detector output is also directly connected to the controller. If an object is obstructing the beam the demodulator delivers a 0V constant signal, otherwise should a 40kHz IR beam be received, a 5V constant signal is obtained. The controller output is connected to the circuitry that drives the servomotor. This solution can be seen as an instinctive reaction when the robot senses the ball. However this behavior can be disabled by the processor unit in order the robot to perform different type of actions.

The kicker mechanics is based on a automobile door opening servomotor, which when powered with opposite polarities moves a piston in opposite directions. The piston course is approximately 3cm.

2.4 Communications

Communications is a key point in any cooperative robotics society. Depending on the choices made at this level, different cooperation mechanisms and strategies can be developed.

A wireless RF Ethernet link (WaveCell from Aaron Tech.) was chosen to support communications between the robots. The devices work on two possible switch-selectable frequencies: 2.4GHz and 2.4835GHz. The bandwidth is 2Mbps, and a range of 150m is covered, inside an office environment.

2.5 Software

Each robot's software runs under the Linux operating system [1]. The reasons for this choice were: robustness, lightweight multitasking, scalability, networking facilities, and availability of programming languages compilers, as well as easy integration of programming languages (e.g., Lisp and C). For more details about the low-level software see [3].

The top-level software, which is responsible for each robots' behavior was initially implemented in an agent programming language — RUBA — developed in previous work [17]. Briefly, RUBA is a language that implements a society of agents, that communicate among them and with the exterior, by the means of a blackboard structure. The whole team is viewed as a single agent society with a common blackboard (distributed among them, but considered as being unique).

Since the RUBA language was not originally designed for implementing teamwork in multi-agent systems it has shown a lack of expressiveness for more complex agent interactions, mainly in what concerns cooperation actions and communications, and representation of other agents' beliefs and goals. These aspects are essential for implementing an adequate teamwork behaviour. So, the SocRob project team is now working on a new agent-based programming language, where those aspects are being taken into account.

Future work includes the creation of language primitives for implementing a pre-defined teamwork model. In particular, agents are required to explicitly have a representation of their common beliefs, teamwork plans and team goals, and an efficient mechanism to reason about teamwork action and communication [15].

3 Functional Architecture

From a functional standpoint, the whole robot society is composed of functionally heterogeneous robots. In the particular case of soccer robots, the functionalities correspond to behaviors and currently are *Goal Keeper*, *Defender*, *Midfielder* and *Forward*. The functional architecture is *scalable* regarding the number of robots (or *agents*) involved. This means that, when a new robot joins the society, no changes have to be made to the overall system. The functional architecture establishes three levels, inspired in [5, 6]. Their description is briefly presented in the sequel (for more details see [3] and [16]).

Organizational level — This level deals with the issues common to the whole society. For instance, in the soccer team context, these can be the state of the game according to the rules, the way the team has to behave in order to follow them, and the team's global strategy based on the current game status.

Relational level — At this level, groups of agents cooperate/negotiate in order to establish a mutual agreement(commitment) concerning the execution of a particular action or the achievement of some objective. The issues involving the formation and termination of (sub-)groups, the establishment of mutual commitments and the monitorization of other agents' task performance are handled at this level. The idea is to integrate this functional architecture with a model of teamwork, such as the ones used or presented in [4, 7, 8, 15].

Individual level — The individual level encompasses all the available behaviors of each robot. A behavior is a set of purposive primitive tasks sequentially and/or concurrently executed. These primitive tasks consist of *sense-think-act loops*, a generalization of a closed loop control system which may include motor control, ball tracking, ball following, etc.

The player behaviors already implemented or to be implemented in a near future are (for more details see [3]):

Goal Keeper – A good goal keeper is essential in a team that wants to win. Being so, a significant part of our effort has been devoted to the development of a efficient goal keeper.

Defender – The defender mission is to move the ball from the vicinity of its own goal to the opponents field.

MidFielder – Like in real soccer, this player should be able to play in a variety of positions. Its mission is to receive the ball from its own team field and decide what to do, based on current game status and teamwork in progress.

Forward – This behavior induces the player to preferentially be in the opponents' field. If the ball is into its team field, the forward player mission is to keep tracking the ball, avoiding to leave its zone. When the ball moves into its influence zone, it must try to take control over it and kick it into the opponents' goal.

4 Agent-based Programming Language

The idea beyond the development of a programming language specially adequated for implementing multi-agent systems follows the work previously done by some members of our team — an agent-based programming language called RUBA [17]. The goal is to have a way for defining agents' architecture, creating agents, establishing communication links among agents, specifying cooperation mechanisms (based on a particular teamwork model), creating and deleting temporary sub-groups, and removing agents.

The initial version of the computational model for the language consists of two classes of objects: *agents* and *blackboards*. A blackboard is the basic communication medium among agents, either to communicate among themselves, or between them and the external world. In what concerns RUBA, the current language specifications propose several improvements: extension of the blackboard for a distributed system, efficient blackboard indexing using a hierarchical name-space, and event-driven programming.

4.1 Distributed Blackboard

Conceptually, a blackboard is a centralized repository of data. The idea of a distributed blackboard is to distribute the information (data) among the agents.

Practically, a blackboard is a mapping of symbols (hierarchically organized in nested name-spaces, *e.g.* `robot0.sensors.collision.2`) to variables. This scheme is supposed to uniformly implement several and different processes, such as message passing, shared memory, distributed data and local variables. A blackboard is implemented with an hash table of names to variables. Each variable has a set of atributes, such as scope, location, policy, type and lock. Also, there are a set of primitives to access the variable: `read`, `write`, `hook` and `lambda` [16].

4.2 Agent Programming

The syntax of the language is based on LISP. The agent programming is supported on three elements: (production) rules, states and events. Each agent has a private set of variables. Each one of these elements are defined by clauses.

The syntax of a rule is

```
(:if expression
 :then clauses1 :go-state state-spec1
 :else clauses2 :else-state state-spec2)
```

being interpreted as usual — if *expression* returns true, *clauses1* is evaluated while *state-spec1* specifies a state transition. Otherwise, *clauses2* and *state-spec2* are taken into account.

The state clauses have the syntax:

(`:on-state` *state-cond [clause]**)

which simply takes into account the contained clauses when the state satisfies the *state-cond* condition.

The event concept corresponds to the interactive nature of robotic applications, which *sense-think-act* loops are a much more natural approach than the classical functional recursive decomposition paradigm [13, 12]. The syntax of an event clause is

(`:on-event` *event-spec* `:do` *clauses1* `:scope` *clauses2*)

where the contained clauses are taken into account only when an event satisfying the *event-spec* specification occurs.

The execution model is different from that in RUBA, in the sense that the rules are only scanned once, when the agent is created. These clauses may trigger state changes, which makes the agent scan other clauses. This can be understood as an event based model, which is an implicit loop model.

4.3 Low-level Integration

In a real-robot context, the issue of integrating high-level programming and the hardware interface arises. One possible form of doing so consists of using an FFI (foreign function interface). However, there is a cleaner alternative, which avoids the overhead of a cross function calling — a low-level access to the blackboard. In fact, the low-level interface looks just like any other agent(s) transacting information with the blackboard, reading and writing from/to variables in the blackboard.

5 Conclusions and Future Work

Currently, our robots are capable of simple but essential behaviors, composed of primitive tasks, such as following a ball, kicking a ball, scoring goals and defending the goal, using vision-based sensors and the other available sensors. Our current and future work is centered on i) development of the self-localization system based on a vision camera and a mirror, ii) update and tuning of the low-level software, iii) design and implementation of an agent-based programming language suitable for multi-agent systems, iv) study and development of a teamwork model and its integration with our functional architecture.

The work has been carried out in a bottom-up fashion, since we believe that many conceptual issues can be raised from and are strongly constrained by the actual implementation problems. Nevertheless, the basic framework described in the paper, concerning hardware, software and functional architectures, was defined in the beginning of the project and has been essentially kept unchanged so far.

References

[1] Linux online. URL: `http://www.linux.org`, 1998.

[2] R. Alamis, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot Cooperation in the MARTHA Project. *IEEE Robotics and Automation Magazine*, 5(1), March 1998.

[3] P. Aparicio, R. Ventura, P. Lima, and C. Pinto-Ferreira. *ISocRob - Team Description*, chapter In Minoru Asada and Hiroaki Kitano, editors, RoboCup-98: Robot Soccer World Cup II. Springer-Verlag, Berlin, 1999.

[4] P. Cohen and H. Levesque. Teamwork. Technical Report 504, Center for the Study of Language and Information, SRI International, March 1991.

[5] Alex Drogoul and C. Dubreuil. A distributed approach to n-puzzle solving. In *Proceedings of the Distributed Artificial Intelligence Workshop*, 1993.

[6] Alex Drogoul and J. Ferber. Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies. In *Actes du Workshop MAAMAW'92*, 1992.

[7] B. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–358, 1996.

[8] N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75:195–240, 1995.

[9] P. Lima and C. Cardeira. The MiniRobot Project: Learning from Building Small Mobile Robots. *IST Science & Technology*, 3, December 1998.

[10] Michael Mateas. An oz-centric review of interactive drama and believable agents. Technical Report CMU-CS-97-156, Carnegie Mellon University, 1997.

[11] A. Paiva, J. Self, and R. Hartley. Externalising learner models. In *International Conference on Artificial Intelligence in Education*, 1995.

[12] Lynn Andrea Stein. Preaching what we practice: How ai is changing the concept of computation. AAAI-97 Invited Presentation, July 1997.

[13] Lynn Andrea Stein. *Interactive Programming In Java*. Morgan Kaufmann, 2001. (to appear).

[14] Peter Stone and Manuela Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. Technical Report CMU-CS-97-193, CMU, School of Computer Science, Carnegie Mellon University, May 1997.

[15] M. Tambe. Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[16] R. Ventura, P. Aparício, and P. Lima. Agent-based programming language for multi-agent teams. Technical Report RT-701-99, RT-401-99, Instituto de Sistemas e Robótica, IST-Torre Norte, March 1999.

[17] Rodrigo M. M. Ventura and Carlos A. Pinto-Ferreira. Problem solving without search. In Robert Trappl, editor, *Cybernetics and Systems '98*, pages 743–748. Austrian Society for Cybernetic Studies, 1998. Proceedings of EMCSR-98, Vienna, Austria.

[18] Mike Wooldridge and Nick Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.

[19] Alex S. Fukunaga Y. Uny Cao, Andrew B. Kahng, and Frank Meng. Cooperative Mobile Robotics: Antecedents and Directions. In *http://www.cs.ucla.edu:8001/Dienst/UI/2.0/Describe/-ncstrl.ucla_cs%2f950049?abstract=Cooperation*, December 1995.