ISocRob 2003: Team Description Paper

Pedro Lima, Luis Custódio, Miguel Arroz, Hugo Costelha, Bruno Damas,
Cláudio Gil, Gonçalo Neto, Pedro Pinheiro, Vasco Pires

Instituto de Sistemas e Robótica, Instituto Superior Técnico,
Av. Rovisco Pais, 1 – 1049-001 Lisboa, PORTUGAL,
pal@isr.ist.utl.pt,
WWW home page: http:socrob.isr.ist.utl.pt

Abstract. This paper describes the status of the RoboCup Middle-Size League team ISocRob after RoboCup 2003. The current hardware of the team robots, the agent-based software architecture, the research results reached so far and the current research goals for the SocRob (*Soccer Robots* or *Society of Robots*) project are described.

1 Introduction and Overview

The ISocRob team and its regular participation in the Middle-Size League (MSL) of RoboCup since 1998 are the competition side of the SocRob project[1], a research endeavour of the Intelligent Systems Laboratory of the *Instituto de Sistemas e Robótica* of *Instituto Superior Técnico* (ISR/IST), which started in 1997. The project goal is to develop a novel approach to the design of a population of cooperative robots based on concepts borrowed from Systems Theory [1] and Distributed Artificial Intelligence [2].

This team description paper concerns the current status of the ISocRob team, after RoboCup 2003. The remaining of the paper is divided in three sections: Section 2 describes the details of the robots current hardware. In Section 3, the software architecture is presented. Section 4 addresses the on-going and some past research developed under the project.

2 Robot Team Hardware

ISocRob's team is composed of four Nomadic Super Scout II robots, three of which are displayed in Figure 1. Over the years, several modifications were made to the robots, endowing them with extra sensor capabilities and actuation. A more detailed view of the goalkeeper robot, showing most of the added features, is presented in Figure 2.

The Super Scout II commercial platform was recently modified so as to improve its computing power. This was achieved by endowing each of the four

[1] The acronym of the project stands both for "Society of Robots" and "Soccer Robots", the case study where we are testing our population of robots.

Fig. 1. ISocRob's team: three of the robots.

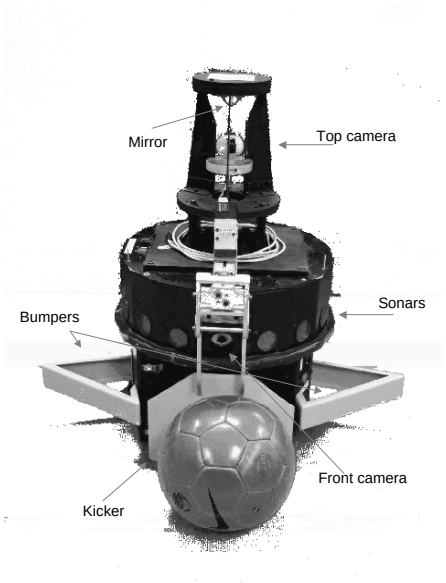

Fig. 2. Details of ISocRob's goalkeeper.

robots with a Pentium III 1000MHz based motherboard, with 512MB RAM and a 8GB disk. The original Motorola MC68332-based daughterboard that connects, through an RS-232 interface, the motherboard with sonars, bumpers and motors, was kept. The power consumption is roughly 400 W.h, most of it due to the electronics (motherboard, daughterboard, DC-DC converter), but also related to the control of the two DC motors. To provide such power, two 18 A.h lead-acid batteries are used per robot.

Two Pittman GM 14902E032 30,3 V (operated at 24 V) DC motors with 5.9:1 gear ratio support each robot differential-drive kinematic structure. The maximum velocity achievable with such motors is 1 m/s, a clear current limitation. One additional actuator per robot is the kicker, based on an electro-pneumatic solution supported on FESTO components. The kicker can make the ball cross the current MSL field length (10 m).

Each robot is endowed with the following sensors:

- 2 Philips 740K Pro, ToUCam USB web cams: one looking forward at the robot front, the other being part of an omni-directional catadioptric vision sensor, whose mirror was designed to obtain a bird's eye-view of the soccer field that preserves in the image the distances in the field, related by a simple gain.
- 14 sonars disposed in a ring around the robot (the original front and back sonars were replaced by the front web cam and a ventilating fan, respectively).
- 1 512 CPR encoder per motor for motor control and odometry.

Communications among robots and with the external human-interface computer are based on WaveLan Turbo 11Mbs wireless Ethernet PCMCIA cards. These follow the IEEE 802.11b protocol.

3 Software Architecture

The software architecture developed for the SocRob project has been defined based on three essential concepts: *micro-agents*, *blackboard* and *plugins*. Each functional module of the SocRob architecture was implemented by a separate process, using the parallel programming technology of threads. In this context a functional module is named micro-agent [3]. Information sharing is accomplished by a distributed *blackboard* concept, a memory space shared by several threads where the information is distributed among all team members and communicated when needed. SocRob software architecture distinguishes between the displayed behaviour and its corresponding implementation through an operator. Operators can be easily added, removed and replaced using the concept of *plugin*, in the sense that each new operator is added to the software architecture as a plugin, and therefore the micro-agent CONTROL, the one responsible for running the intended operator, can be seen as a multiplexer of plugins. Examples of already implemented operators are: dribble, score, or go, to name but a few. Each virtual vision sensor is also implemented as a plugin. The software architecture is supported on the Linux Operating System.

3.1 Micro-Agents and Plugins

A micro-agent is a Linux thread continuously running to provide services required for the implementation of the reference functional architecture, such as reading and pre-processing sensor data, depositing the resulting information in the BB, controlling the flow of behaviour execution or handling the communications with other robots and the external monitoring computer. Each micro-agent can be seen as a plugin for the code. The different plugins are implemented as shared objects. In the sequel, the different micro-agents are briefly described (see also Figure 3).

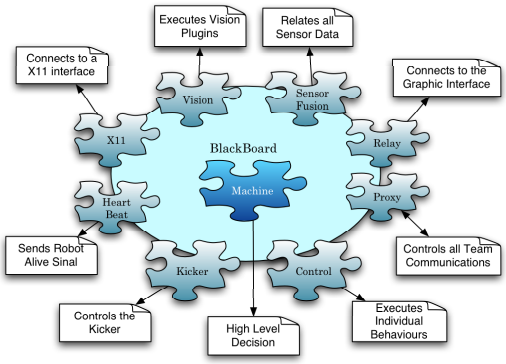

Fig. 3. Software architecture showing micro-agents and the blackboard.

Micro-agent VISION: This micro-agent reads images from one of two devices. Currently, the devices are the USB web cams whose images can be acquired simultaneously. However, the bandwidth is shared among the two cameras. Actually, one micro-agent per camera (up camera and front camera) is implemented. Each of them has several modes available, such as up to detect the ball with the up camera, front to detect the ball and the goals with the front camera, self for self-localisation with the up camera, and emptyspot to determine the region around the robot with the largest amount of green, with the up camera. Each mode is implemented as a plugin for the code.

Micro-agent SENSORFUSION: This micro-agent uses a Bayesian approach to the integration of the information from the two cameras of each robot and from the 4 team robots [4].

Micro-agent CONTROL: This micro-agent receives the operator/behaviour selection message from the MACHINE micro-agent and runs the selected operator/behaviour, by executing the appropriate plugin. Currently, each micro-agent is structured as a finite state machine where the states correspond to primitive tasks and the transitions to logical conditions on events detected through information put in the BB by the SENSORFUSION micro-agent. This micro-agent can also select the vision modes by communicating this information to

the VISION micro-agent. Different control plugins correspond to the available behaviours.

Micro-agent MACHINE: This micro-agent coordinates the different available operators/behaviours (CONTROL micro-agents) by selecting one of them at a time. The operator/behaviour chosen is communicated to the CONTROL micro-agent. Currently, behaviours can be coordinated by:

- a finite state machine, where each state corresponds to a behaviour and each transition corresponds to a logical condition on events detected through information put in the BB by the VISION (e.g., found_ball, front_near_ball) and CONTROL (e.g., BEHAVIOUR_SUCCESS, BEHAVIOUR_FAILURE) micro-agents.
- a rule-based decision-making system, where the rules LHS test the current world state and the rules RHS select the most appropriate behaviour.

Micro-agent KICKER: This micro-agent sends open and close signals to the electronics of the kicker electro-pneumatic valve.

Micro-agent PROXY: This micro-agent handles the communications of a robot with its teammates using TCP/IP sockets. It is typically used to broadcast through wireless Ethernet the BB shared variables (see below).

Micro-agent RELAY: This micro-agent relays the BB information on the state of each robot to the game interface running in an external computer, using TCP/IP sockets. Typically, the information is sent through wireless Ethernet, but for debug purposes a wired network is also supported.

Micro-agent X11: This micro-agent handles the X11-specific information sent by each robot to the external computer, using TCP/IP sockets. It is typically used to send through wireless Ethernet the BB shared variables for text display in an X-window.

Micro-agent HEARTBEAT: This micro-agent sends periodically a message from each robot to its teammates to signal that the sender is alive. This is useful for dynamic role changes when one or more robots "die".

3.2 Distributed Blackboard

The distributed blackboard extends the concept of blackboard, i.e., a data pool accessible to several agents, used to share data and exchange communication among them. Traditional blackboards are implemented by shared memories and daemons that awake in response to events such as the update of some particular data slot, so as to inform agents that require that data updated. ISocRob distributed blackboard consists, within each individual robot, of a memory shared among the different micro-agents, organised in data slots corresponding to relevant information (e.g., ball_position, scout1_posture, own_goal), accessible through data-keys. Whenever the value of a BB variable is updated, a time stamp is associated to it, so that is validity (based on recency) can be checked later.

Some of the blackboard variables are local, meaning that the associated information is only relevant for the robot where the corresponding data was acquired

and processed, but others are global, and so their updates must be broadcasted to the other teammates (e.g., the ball position).

3.3 Hardware Abstraction Layer (HAL)

The Hardware Abstraction Layer is a collection of device-specific functions, providing services such as the access to vision devices, kicker (through the parallel port), robot motors, sonars and odometry, created to encapsulate the access to those devices by the remaining software. Hardware-independent code can be developed on the top of HAL, thus enabling simpler portability to new robots.

4 Addressed Research

From the very beginning of the project, one main concern has been the development of **behaviour coordination and modelling** methods which support our integrated view to the design of a multi-robot population, not necessarily for playing robot soccer. The original architecture considers three types of behaviours to be displayed by the team:

organisational: those concerning the team organisation, such as the roles of each player,
relational: those concerning the display of relations among teammates (coordination and cooperation)
individual: those concerning each robot as an individual.

Behaviours are externally displayed and emerge from the application of certain operators. From the operators standpoint, the architecture has three levels:

Team Organisation Level where, based on the current world model, a *strategy* is established, including a goal for the team. This level is currently not implemented, but it should consider issues such as modelling the opponents behaviour to plan a new strategy. Strategies may simply consist of enabling a given subset of the behaviours at each robot.

Behaviour or Task Coordination Level where switching among behaviours, both individual and relational, occurs so as to coordinate behaviour/task execution at each robot towards achieving the team goal, effectively establishing the team *tactics*. Either a finite state automaton or a rule-based system can currently implement this level, but other alternatives are possible, such as Petri nets.

Behaviour Execution Level where primitive tasks run and where they interface the sensors, through the blackboard, and the actuators, through the navigation functions at each robot. Primitive tasks are linked to each other so as to implement a behaviour. Currently, each behaviour is implemented as a finite state automaton whose states are the primitive tasks and transitions are associated to logical conditions on events that are detected by the system. Behaviours can be *individual*, if they run in one robot only, or

relational, if two or more robots are running behaviours that are coordinated through commitments and synchronisation messages to achieve a common goal.

Figure 4 shows the functional architecture from the operators standpoint. Figure 5 zooms the Behaviour Execution Level. Figure 6 depicts the block diagram of the world model and its interface with sensors and actuators, through the BB, navigation functions and behaviours.

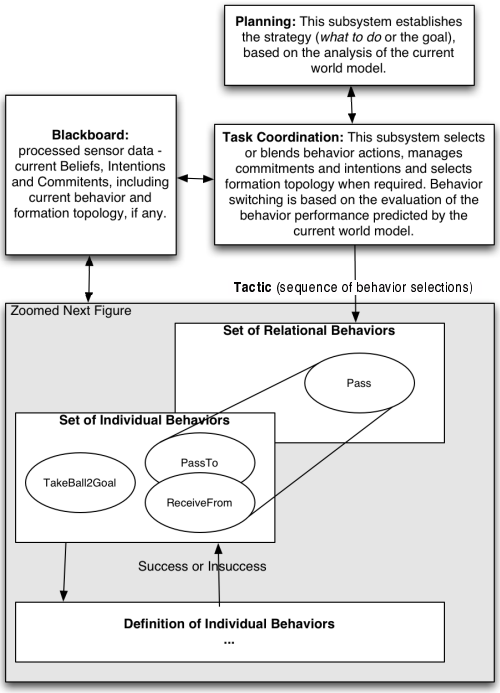

Fig. 4. Functional architecture from the operators standpoint.

As many other other RoboCup MSL teams, we have been spending some of our research time investigating better solutions for **colour segmentation**. A colour segmentation interface was developed, providing two alternatives to discriminate the relevant MSL colours in HSV (Hue-Saturation-Value) colour space: adjusting HSV intervals and graphically selecting regions with a given pixel colour. The two approaches are cumulative. Furthermore, **object segmentation** is a topic directly related to the previous one, as we discriminate objects, namely the ball and the goals, not only based on their colour, but also on their shape (e.g., by fitting circles to observed orange bulbs and identifying the ball with the closest and more circular bulb).

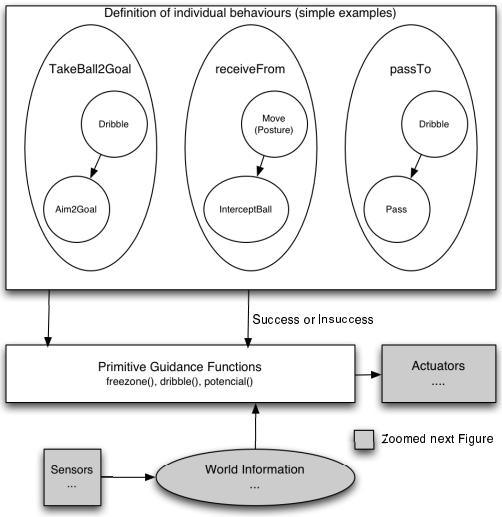

Fig. 5. Functional architecture Behaviour Execution Level.

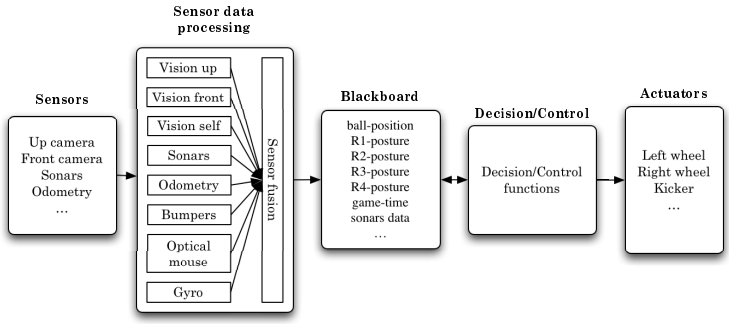

Fig. 6. World model and its interface with sensors and actuators, through the BB, navigation functions and behaviours.

A topic of current research within the project is the use of **sensor fusion** for world modelling. The goal is to maintain and update over time information on the relevant objects, such as ball position and velocity, teammates pose and velocity, opponents pose and velocity, or position of the goals with respect to the robot. Such information is obtained by each robot from the observations of its front and up cameras and then fused among all the team robots [4], using a Bayesian approach to sensor fusion [5]. Currently this approach is used to provide information on ball position to all the team members.

Metric navigation has been one of our major research topics. We developed an omni-directional catadioptric vision system which preserves the Euclidean norm for ground-plane views and used it to simplify each team robot self-localisation [6, 7]. Using this vision-based self-localisation method to reset odometry after the occurrence of relevant events (such as bumping into some obstacle), a multi-sensor (vision, sonar and odometry) navigation system was developed for non-holonomic robots [8]. Later, a modified potential fields solution for navigation was also introduced and is currently used by the team. The new method allows moving the robot avoiding obstacles while keeping the ball [9]. Recent work [10] has concentrated on the goalkeeper navigation, namely providing a novel vision-based self-localisation method which relies on local field features (the goal posts and colour), as well as line following and posture stabilisation methods based on non-linear state feedback techniques [11, 12].

Another research topic concerning multi-team navigation is the development of novel **formation control** methods [13] to dynamically prescribe the robots distribution over the field under different situations. The new methods are based on a general mathematical framework to describe the inter-teammates constraints and the external constraints resulting from obstacles to be avoided, appearing on the formation path. Since the internal constraints can in general be elastic (e.g., simulating a spring-damper system) the formation can temporarily be deformed to avoid an obstacle. We also plan to move in the mid-term to a mixture of metric and **topological navigation** [14] methods, pointing towards **behaviour description languages** which will increase the modelling convenience and power of the current two-level state machine.

Both relational and individual operators [3] have been implemented as state machines. The choice among available behaviours was made until 2003 by a higher level state machine. This two-level state machine implementation naturally leads to a hybrid automata [15] model of the robot behaviours. In [10], the composition of the goalkeeper and ball hybrid automata models lead to the design from specifications of some of the decision-level parameters of the robot.

However, state machines have not only limited modelling power, as the language they generate are limited to *regular languages*, but also lack modelling convenience, i.e., a slightly more complex behaviour may require a very complex state machine to be described. One of the on-going research topics is the development of a distributed decision-making architecture supported on a logical approach to **modelling dynamical systems** [16], which is based on situation calculus, a first order logic dialect. This architecture includes two main mod-

ules: i) a basic logic decision unit, and ii) an advanced logic decision unit. Both run in parallel; the former one intends to quickly suggest, using simple logical decision rules, the next behaviour to be executed, whereas the latter uses more sophisticated reasoning tools (e.g., situation calculus) capable of planning, learning and decision-making, both for individual and cooperative (teamwork) situations. This configures an hybrid architecture where the *basic (reactive) unit* only controls the robot if the *advanced (deliberative) unit* takes too long to make a decision, assuming a situation urgency evaluation [17]. A partial implementation of this architecture, the basic logic decision unit, was already performed using Prolog to replace the higher level state machine [18]. In 2003, this approach was used to select behaviours at the MACHINE micro-agent level. Its modelling convenience allowed the quick development of different roles for field players (attacker *vs* defender), as well as **dynamic role change** between field players (defenders switch with attackers, depending on who is in a better position to get the ball).

The advanced (deliberative) unit will be further developed using an action programming language called Golog [19].

Other approaches to decision-making, such as emotions (e.g., agents with personality) and learning-based methods, are being investigated in the new ISocRob simulation team which has developed a 3D monitor for visualising and debugging RoboCup Simulation League games [20]. The real robot code development and test will also be simplified by the usage of a new simulator whose *beta* version has been recently finished. The simulator is based on CS Freiburg's simulation environment and currently runs the whole ISocRob real robots code. It will also be used to develop a new navigation system and new behaviours for the new omnidirectional motion and perception robots that we are developing and plan to present in RoboCup 2004.

Acknowledgements

This work is supported by the ISR/IST pluriannual funding from the Fundação para a Ciência e a Tecnologia and POSI, in the frame of the QCA, and by ICEP – Investimentos, Comércio e Turismo de Portugal, Alcatel–Portugal and Mota & Teixeira.

References

1. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Kluwer Academic Publishers (1999)
2. Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999)
3. Lima, P., Custódio, L., Ventura, R., Aparício, P.: A functional architecture for a team of fully autonomous cooperative robots. RobCup-99: Robot Soccer World Cup III (1999)
4. Marcelino, P., Nunes, P., Lima, P., Ribeiro, M.I.: Improving object localization through sensor fusion applied to soccer robots. In: Proceedings of the Workshop of ROBOTICA 2003 - 3rd Portuguese Robotics Festival, Lisboa, Portugal (2003)

5. Durrant-Whyte, H.F.: Integration, Coordination and Control of Multi-Sensor Robot Systems. Kluwer Academic Publishers (1988)
6. Lima, P., Bonarini, A., Machado, C., Marchese, F., Marques, C., Ribeiro, F., Sorrenti, D.: Omni-directional catadioptric vision for soccer robots. Journal of Robotics and Autonomous Systems **36** (2001)
7. Marques, C., Lima, P.: A localization method for a soccer robot using a vision-based omni-directional sensor. In P. Stone, T. Balch, G.K., ed.: RoboCup-2000: Robot Soccer World Cup IV. Springer-Verlag, Berlin (2001) 96–107
8. Marques, C., Lima, P.: Multi-sensor navigation for soccer robots. In A. Birk, S. Coradeschi, S.T., ed.: RoboCup-2001: Robot Soccer World Cup V. Springer-Verlag, Berlin (2002)
9. Damas, B., Lima, P., Custódio, L.: A modified potential fields method for robot navigation applied to dribbling in robotic soccer. In: Proceedings of RoboCup-2002 Symposium, Fukuoka, Japan (2002)
10. Lausen, H., Nielsen, J., Nielsen, M., Lima, P.: Model and behavior-based robotic goalkeeper. In: Proc. of RoboCup Symposium 2003, Padova, Italy (2003)
11. de Wit, C.C., Siciliano, B., (Eds)., G.B.: Theory of Robot Control. Springer-Verlag, London, UK (1996)
12. Indiveri, G.: *An Introduction to Wheeled Mobile Robot Kinematics and Dynamics.* Slides for lecture given at RoboCamp, Paderborn (Germany) (2002)
13. García, A., Lima, P.: Robot formations motion dynamics based on scalar fields. In: Proc. of IEEE ICAR'03, Coimbra, Portugal (2003)
14. Neto, G., Costelha, H., Lima, P.: Topological navigation in configuration space applied to soccer robots. In: Proc. of RoboCup Symposium 2003, Padova, Italy (2003)
15. van der Schaft; Hans Schumacher, A.: An Introduction to Hybrid Dynamical Systems. Springer (1999)
16. Reiter, R.: Knowledge in Action. MIT Press (2001)
17. Sadio, R., Tavares, G., Ventura, R., Custódio, L.: An emotion-based agent architecture application using real robots. In: Proceedings of the Emotional and Intelligent II: The Tangled Knot of Social Cognition, AAAI Fall Symposium. (2001)
18. Arroz, M., Pires, V., Custódio, L.: Logic-based distributed decision system for a multi-robot team. In: Proceedings of the Workshop of ROBOTICA 2003 - 3rd Portuguese Robotics Festival, Lisboa, Portugal (2003)
19. Levesque, H., Reiter, R., Lesprance, Y., Lin, F., Scherl, R.: Golog: A logic programming language for dynamics domains. Journal of Logic Programming (1997)
20. Penedo, C., Pavão, J., Nunes, P., Custódio, L.: Robocup advanced 3D monitor. In: Proc. of RoboCup Symposium 2003, Padova, Italy (2003)