ISocRob — Team Description*

Pedro Aparício, Rodrigo Ventura, Pedro Lima, and Carlos Pinto-Ferreira

Instituto de Sistemas e Robótica/Instituto Superior Técnico
Rua Rovisco Pais, 1; 1096 Lisboa Codex ; Portugal
{aparicio,yoda,pal,cpf}@isr.ist.utl.pt

Abstract. This paper provides a technical description of the ISocRob team, participating at the RoboCup'98 robotic soccer competition, in the mid-size league. The ISocRob constitutes a challenge for research on broad and generic approaches for the design of a cooperating society of robots. This paper addresses both design and implementation issues of the soccer "players", as well as hardware construction details.

1 Introduction

Multi-agent systems have become very popular in recent years, especially as a research area in Distributed Artificial Intelligence (DAI) [7]. Simultaneously, several robotic systems based on a fleet of robots have been developed, as an alternative to single-robot systems common in the past [11]. Therefore, it seems that a promising research direction will result from joint work between researchers interested in DAI and Intelligent Robotics.

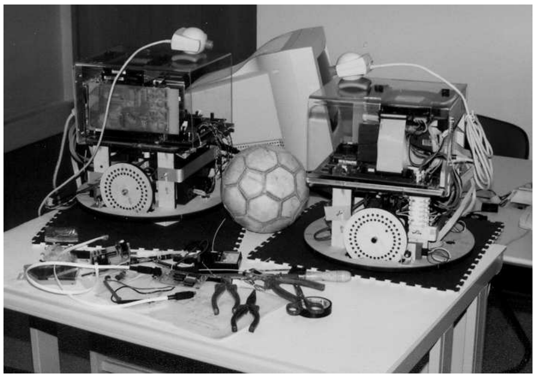

Fig. 1. Two of ISocRob Team members.

The Artificial Intelligence and Manufacturing Systems and the Intelligent Control laboratories of the *Instituto de Sistemas e Robótica* at the *Instituto*

* This work has been supported by grants from the following Portuguese entities: Fundação Calouste Gulbenkian, ISR/IST, FCT - PRAXIS XXI/BM/12937/97.

Superior Técnico (ISR/IST) have started almost one year ago a joint project on Cooperative Robotics, 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* cooperative robots, either as a society and as individuals. The robots are developed from scratch, so that both conceptual and implementation issues are considered. Special attention is taken to cooperation-oriented communication issues, such as the type of information that must be shared and how to distribute that information. This work relies on past experience of both groups regarding topics relevant to robot development [3] and DAI [8].

A case study on Robotic Soccer involving a team of 3 robots, two of which are shown in Figure 1, (the **IS**oc**R**ob Team) is currently underway. The main objective of this case study is to have the team ready to compete at the RoboCup98 in Paris.

This technical report is organized as follows: Section 2 and 3 describe the details of each robot Hardware and Software Architectures, respectively. The Functional Architecture, presented in Section 4, wraps up the whole picture, relating conceptual issues to the two physical architectures explained in the previous sections.

2 Hardware Architecture

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

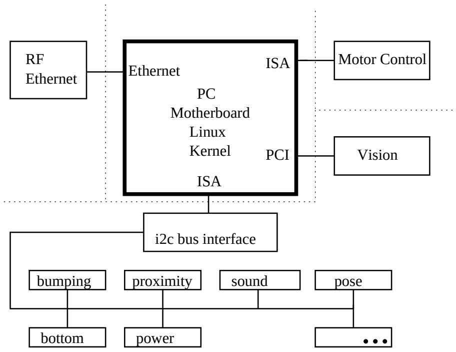

Fig. 2. Hardware Architecture.

2.1 The Processing Unit

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

The PC motherboard was chosen because it gives the best performance/price rate. Another major advantage is the ease of getting software and hardware to work with.

2.2 Sensorial Systems

The sensors of each robot are divided in two main groups:

- **vision sensors**: virtual sensors which extract information from the images acquired by a video camera and its interface board. One physical transducer (the video camera) leads to many (virtual) sensors.
- **pose, bumping, sound and proximity sensors**, each of them physically associated to one transducer.

All transducers but the video camera interface the processing unit through an *i2c* bus. From the sensors side, this interface is implemented on PIC micro-controllers, used together sensor data. This is intended to improve system modularity, simplicity and robustness. Each sensor has a unique *i2c* identification.

The details of the different sensors/transducers are described in the sequel, with the exception of the bottom and power sensors, which are intended for applications other than the RoboCup challenge. Those sensors detect ground color/presence and power failures, respectively.

Video Camera — The video camera is a Phillips XC731/340 interfacing the motherboard through a PCI Captivator board. This combination allows the acquisition of 640×480 images at a frame-rate of 50 interlaced frames per second. Image is used for several purposes, namely, to identify and/or follow/catch the team mates, the opponents, the ball and the goals. The digital camera interfaces with the processing units through the PCI bus interface.

Pose Sensor — Depending on the type of application involved, each robot of the society may need to regularly update its current pose with respect to a reference frame (e.g., located in the field center). This may be accomplished based on the *triangulation principle*: from the measurement of the angles between the robot longitudinal axis and the direction of maximum signal reception from infra-red (IR) beacons whose location in the reference frame is known, the robot is able to compute its pose relative to that frame.

The IR beacons are active, emitting a signal of unique frequency, modulating a 40kHz carrier in amplitude. Each beacon uses a different frequency, set in a local micro-controller, in order to be easily identified. Three beacons must be simultaneously visible at any location, so that the vehicle can locate itself. Possible locations regarding the RoboCup challenge are shown in Figure 3.

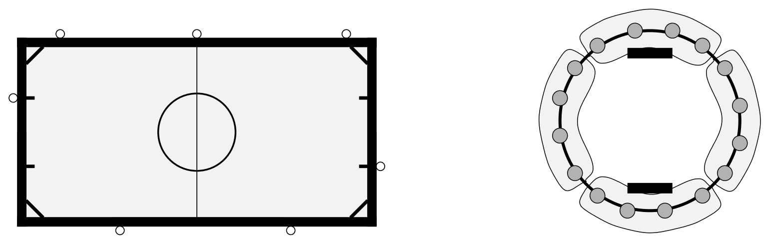

Fig. 3. On the left, some possible beacon locations, and on the right, the micro-switch sensor (bumpers) placement.

The receiver block corresponds to a rotating IR receiver (SHARP GPU158Y decoder), placed at the same height of the emitters, on the top of each robot. When the algorithm finishes its loop execution, the micro-controller has two vectors in memory, that keep the angular position and the identification of each beacon found. This information is available through the *i2c* sensor bus.

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. Figure 3 presents a possible location of the bumping sensors around the vehicle (horizontal plane). These sensors are made with micro-switches, arranged in a serial connection, divided in 4 sets of 4 micro-switches.

Sound Sensor — Sound is a powerful way of communicating with robots, as it is omnidirectional and does not need a physical connection. Our robots are endowed with a sound detector, used to detect a whistle blowing as their starting condition. A special circuit was developed, based on a commercial microphone, which detects a whistle blow in the 1kHz audio band.

Proximity Sensor — Proximity sensors are also based on IR technology. They use the same IR detector modules of the pose sensor, with a small modification. The modification allows the measure of an analog value proportional to the object distance (depending on the material reflectance). The six emitter/receiver pairs are equidistantly located around the vehicle, pointing outside, in order to detect objects in the near vicinity. The typical range of this sensor goes from 20 cm to 2 m.

2.3 The Actuators

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. This is done in closed loop, using two National LM629 motor controller chips. A special board was built to interface the motor controllers with the motherboard, through the ISA bus. The motor controllers have two inputs: the wheels current speed, measured with incremental encoders; and the velocity set points, established by the central processing unit. The output is a PWM signal and a bit indicating the direction of rotation for each wheel. Those signals go to the power amplifier, based on a LM18200 H-bridge. The motor controller outputs to the power amplifiers are optically isolated, in order to prevent malfunctions caused by motor-generated spikes. As such, the vehicle has two batteries, one for motor power and the other for the electronics.

2.4 Communications

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.The RF link may work in a peer-to-peer or client/server configuration. The latter requires an Access Point box external to the robots.

3 Software Architecture

Each robotś software runs under the Linux [1] operating system. 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).

In order to support the robots' custom hardware (motor controllers, sensors, etc.), kernel-level devices were developed, except for the i^2c driver [9]. The separation provided by the kernel between device drivers and user code made their development and bug tracking less problematic than usual. The design strategy was to consider these devices as usual UNIX special files (`/dev/*`), and communicate with them via `ioctl()` system calls.

The video frame grabber is handled by third party software (bttv driver [6]), which provides an API similar to the one described above. The grabbing process runs on a frame by frame basis: an appropriate `ioctl()` call dumps a complete video frame to a pre-allocated (non-swappable) memory region. The vision processing code then works on this memory area.

A set of libraries are provided in order to hide most of these device handling details from higher level software modules. On top of these libraries, a fairly platform independent software layer implements a set of *primitives*. Examples of primitives may include avoiding opponents, finding a teammate, ball tracking, forward/backward motion and so on.

The top-level software, which is responsible for each robots' behavior is implemented in an agent programming language — RUBA — developed by one of the authors in previous work[8]. Briefly, RUBA is a language that implements a society of agents, that communicate between them and with the exterior, by the means of a blackboard structure. The software underlying RUBA was re-implemented in the Scheme programming language [2], computationally lighter and structurally cleaner than the original version, written in Common Lisp. The whole team is viewed as a single agent society with a common blackboard (distributed among them, but considered as being unique). Additionally, each robot has an individual blackboard (as RUBA supports multiple blackboards) to handle issues related to it as an individual. In the next section it will be explained how all these pieces can be put together under a common conceptual framework. The RUBA language is based on production rules interleaved with state machine (multiple machine states are supported) statements, *i.e.,* the rules can be grouped together in a specific state of a state machine. The expressions that fill the `IF`, `THEN` and `ELSE` fields are essentially Scheme expressions with the extension to the *primitives* referred above. This mechanism bridges the gap between high-level agent programming and specific robot actions/sensing.

4 Functional Architecture

From a functional standpoint, the whole robot society, composed of functionally heterogeneous robots, is organized into three levels: *organizational, relational,* and *individual.* This division was first established by Drogoul [4, 5]. The rest of this section will be devoted to the description of each layer in the context of this work.

4.1 Organizational level

This level deals with the issues unconditionally common to the whole society. In the soccer team context, these are:

- the state of the game according to the rules (before kick-off? in-game? off-side?) and the way the team has to behave in order to follow them;
- the global strategy of the team (time to re-positioning of the team? time to attack? time to defend?).

These issues involve the whole team, and thus have to be established at this global level.

4.2 Relational level

In a cooperative robotics context, in order to accomplish useful cooperation, relationships between robots have to be accomplished. This involves an important characteristic of the agent concept: social ability [10]. This means that one given agent has to be aware of the existence of other agents like him, with whom it has to negotiate. This is where the relational level comes into play: at this level, groups of agents negotiate and eventually come to an agreement about some objective (common or not). The issues involving the formation of groups and its disbanding are handled at this level. The key idea of this process is negotiation among agents. The blackboard structure provides the common medium through which the necessary communication circulates.

In any soccer team (human or robotics, hopefully) this kind of commitments are essential if a fruitful result from teamwork is desired. Therefore it plays a decisive role as far as net result of the soccer robotic team is concerned. For instance, if one robot wants to pass a ball to another player (of the same team, hopefully), it has to find someone available and sufficiently well positioned. The ball pass is arranged via a negotiation process, and after an agreement is reached, the pass is performed. In fact, this is what happens in human soccer — the eye glimpse between two soccer players (and several years of experience).

4.3 Individual level

This level encapsulates each robot as an entity, comprising all aspects of a robot as an individual. This includes the individual primitives, regardless of whether they are invoked by the relational level (e.g., path planning) or are generated by the robot itself (e.g. collision avoidance), the robot performance and monitoring, down to the robot constructive issues.

The individual level is responsible for accomplishing each behavior, running the sense-think-act loop of its primitives. This involves processing vision (as well as other sensors) data and driving the motors according to the desired behavior. In most cases, this is generated at the relational level. However, there are exceptional cases (e.g. collision), which demand quick action from the robot, and that have to be taken care of at this level.

References

1. Linux online. URL: `http://www.linux.org`, 1998.
2. H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press and McGraw-Hill, 1985.
3. Pedro Aparício, Jo ao Ferreira, Pedro Raposo, and Pedro Lima. Barbaneta - a modular autonomous vehicle. In *Preprints of the 3rd IFAC Symposium on Intelligent Autonomous Vehicles*, March 1998.
4. Alex Drogoul and C. Dubreuil. A distributed approach to n-puzzle solving. In *Proceedings of the Distributed Artificial Intelligence Workshop*, 1993.
5. 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.
6. Ralph Metzler. Bttv — a linux driver for bt848 based frame grabbers. URL: `http://www.thp.uni-koeln.de/~rjkm/linux/bttv.html`, 1998.
7. 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.
8. 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.
9. Simon G. Vogl. i^2c — bus for linux. URL: `http://www.tk.uni-linz.ac.at/-~simon/private/i2c`, 1998.
10. Mike Wooldridge and Nick Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.
11. 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.