# CAMBADA’2006: Team Description Paper

L. Almeida, J.L. Azevedo, G. Corrente, M.B. Cunha, A. Ferdowsi,
J.P. Figueiredo, P. Fonseca, S. Lopes, R. Marau, N. Lau, P. Pedreiras, A. Pereira,
A. Pinho, J. Rocha, F. Santos, L. Seabra Lopes, V. Silva, J. Vieira

Transverse Activity on Intelligent Robotics
IEETA/DET – Universidade de Aveiro
3810-193 Aveiro, Portugal

**Abstract.** The CAMBADA middle-size robotic soccer team is described in this paper for the purpose of qualification to RoboCup’2006. This team was designed and developed by the authors, from scratch, in the last three years. The players, completely built in-house, incorporate several innovations at the hardware level, particularly the sensing and computational subsystems. At the software level, cooperative sensing uses a real-time database implemented over a real-time Linux kernel. Previous experience of the team in the simulation league has been highly relevant. The paper focuses on recent advances on vision, localization and monitoring/debugging software as well as a new ultrasound-based localization system.

## 1 Introduction

CAMBADA[1] is the RoboCup middle-size league soccer team of the University of Aveiro. This project, started officially in October 2003, is funded by the Portuguese research foundation (FCT) [2]. CAMBADA participated in RoboCup’2004 and in the last two editions of the Portuguese Robotics Festival (RoboCup’2004 and ’2005).

The previous CAMBADA Team Description Paper [2], prepared for RobCup’2004, provides a detailed overview of the team, as it was initially designed and developed. Some aspects of its design were demonstrated in RoboCup’2004 while others were implemented since then. The present paper provides a shorter overview of the project and then focuses on recent developments.

The CAMBADA players were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter, which allows for a team of 5 robots, according to the rules. The mechanical structure of the players is layered and modular (Figure 1). Each layer can easily be replaced by an equivalent one. The components in the lower layer, namely motors, wheels, batteries and an electromechanical kicker, are attached to an aluminium plate placed 8 cm above the floor. The second layer contains the control electronics. The third layer con-

[1] CAMBADA is acronym of *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*; ‘cambada’ is also a Portuguese word for ‘band’ or ‘mob’.

[2] Project POSI/ROBO/43908/2002, partially funded by FEDER.

tains a computer, at 22.5 cm from the floor. The players are capable of holonomic motion, based on three omni-directional roller wheels [5].

The main sensors in each player are two webcams, both equipped with wide-angular lenses and installed at approximately 80cm above the floor. Both cameras deliver 320x240 YUV images at a rate of 20 frames per second (fps). One of the cameras faces the field orthogonally, enabling to capture a 360 degrees view around the robot, approximately with a 1m radius. This so-called omni-directional vision system is used for obstacle avoidance and ball handling.

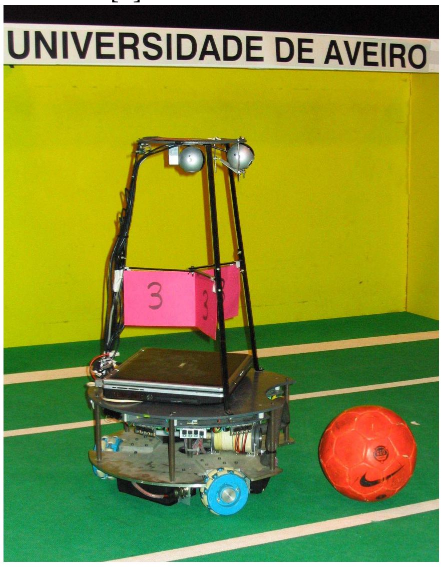

**Fig. 1.** One of the CAMBADA players

The other camera points forward in the direction of the front of the robot, with 57º inclination of with respect to its vertical axis. This frontal system is used to track the ball at medium and long distances, as well as the goals, corner posts and players. All the objects of interest are detected using simple color-based analysis, applied in a color space obt ained from the YUV space by computing phases and modules in the UV plane.

The robots computing system architecture follows the fine-grain distributed model [6] where most of the elementary functions, e.g. closed-loop control of complex actuators, are encapsulated in small microcontroller-based nodes, connected through a network. A PC-based node is used to execute higher-level control functions and to facilitate the interconnection of off-the-shelf devices, e.g. cameras, through standard interfaces, e.g. USB or Firewire (Fig. 3). For this purpose, Controller Area Network (CAN), a real-time fieldbus typical in distributed embedded systems, has been chosen. This network is complemented with a higher-level transmission control protocol to enhance its real-time performance, composability and fault-tolerance, namely the FTT-CAN protocol (Flexible Time-Triggered communication over CAN) [3]. The communication among robots uses the standard wireless LAN protocol IEEE 802.11x profiting from large availability of complying equipment.

The software system in each player is distributed among the various computational units. High level functions run on the computer, in Linux operating system with RTAI (Real-Time Application Interface). Low level functions run partly on the microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1,2,4,7,8] has been adopted. The RTDB is a data structure where players share their world models. It is updated and replicated in all players in real-time.

The high-level processing loop starts by integrating perception information gathered locally by the player. This includes information coming from the vision processes, which is stored in a Local Area of the RTDB, and odometry information coming from the holonomic base via FTT-CAN. After integration, the world state can be updated in the shared area of the RTDB. The next step is to integrate local information with information shared by teammates. This will be the basis for taking decisions according to a finite state machine. Each state is characterized by the behavior pattern that is executed. A very basic coordination mechanism is currently supported. According to this mechanism, the player that takes control of the ball is the player closest to the ball. Other players take strategic positions in the field based on their distances to the goals. We expect to improve the coordination mechanism as soon as localization capabilities are fully evaluated. This also depends on the availability of monitoring and debugging tools, which are under development.

## 2 Real-time vision architecture

A modular multi-process architecture was adopted for the vision software subsystem (Figure 2) [7]. For each camera, one process is automatically triggered whenever a new image frame is ready for dowload. The frame data are placed in shared image buffers, which are afterwards analyzed by the object detection processes, generically designated by proc_obj:*x*, *x*={1,2,…***N***}. These processes are encapsulated in separate Linux processes. Once started, each process gets a pointer to the most recent image frame available and starts tracking the respective object. Once finished, the resulting information (e.g. object detected or not, position, confidence) is placed in the real-time database. This database may be accessed by any other processes on the system, particularly for world state update.

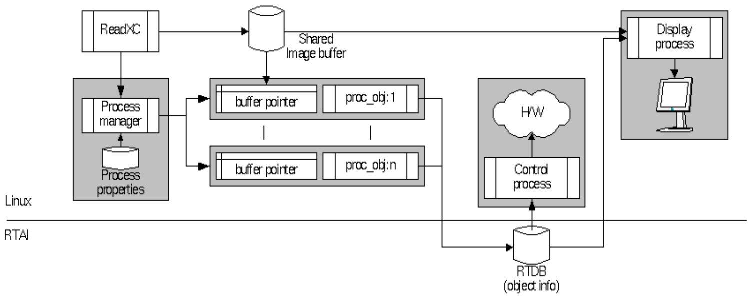

**Fig. 2.** Vision subsystem software architecture

The activation of the distinct image-processing activities is carried out by a process manager. Each object tracking process ($i$) is associated with a period ($P_i$) and a phase ($\varphi_i$), expressed as integer number of image frames. For every frame $f$, the process manager activates all the processes that verify $[(f-\varphi_i)\%\ \mathrm{P_i}\ ]=0$. This allows allocating periods according to the specific attributes of each object (e.g., the ball is highly dynamic and is tracked in every frame while the relative goal position is less dynamic and can be tracked every four frames) as well as to de-phase them in the time domain, minimizing the mutual interference and consequently their response time and jitter.

Scheduling of vision related processes relies on the real-time features of the Linux kernel, namely the FIFO scheduler and priorities in the range 15-50. At this level, Linux executes each process to completion, unless the process blocks or is preempted by other process with higher real-time priority. This ensures that the processes are executed strictly according to their priority with full preemption. The real-time features of Linux are sufficient at this time-scale (periods multiple of 50ms).

## 3 Information Integration and Localization

Localization in the play field is a very basic requirement for implementing advanced coordination and cooperation strategies. Localization includes self-localization and localization of the ball and players. Localization is the main outcome of local and team-level information integration. As expected, odometry information is not enough to maintain sufficiently accurate localization information in CAMBADA [4]. After long distances or through collisions between players, it is very easy to reach positional error levels not acceptable for team coordination purposes. In collision-free runs of 100 m, we verified that the positional error grows roughly linearly with the distance travelled by the player. The error is around 1.5% to 2.% of the distance. Therefore, position errors of 2m can easily occur.

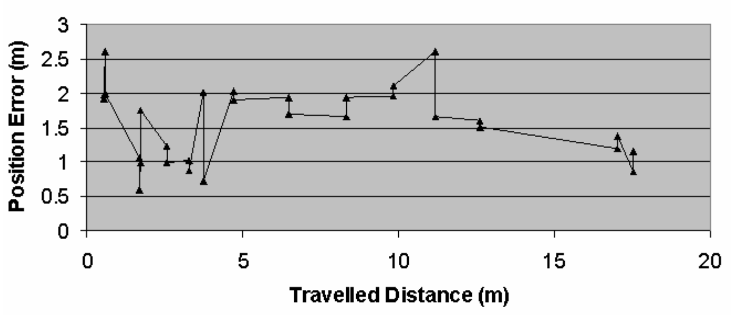

**Fig. 3.** Effect of opportunistic vision-based calibration (example run)

Localization in the currently working CAMBADA team is based on odometry information, updated in each iteration of the control loop, and calibrations performed based on vision information. The calibration mechanisms can be grouped as follows:

– Opportunistic, based on a single landmark (goal, corner post, line) – not enough to derive the player's position and orientation but enables calibration.
– Opportunistic, based on two successively seen landmarks – enables to calculate position/orientation; error inherent to the vision system only.
– Active – The player actively searches for two landmarks, e.g. by performing a full turn around itself.

In opportunistic calibration, the vision-based positions/orientations are averaged with the internally kept values. Active localization is called in extreme situations, and the obtained values replace the previous values. When a change in internal values takes place, the new values are sent down to the odometry micro-controller.

While monitoring/debugging tools are being developed, we have been resorting to time-consuming "manual" evaluation experiments. These experiments show that the position and orientation errors can be reduced to acceptable levels using the methods enumerated above. Figure 3 shows the localization performance in one experiment, in which the initial position error was set to 2.24m. We see that, after running for around 17 meters and having performed 16 opportunistic calibrations, the position error was gradually reduced to ~1m.

## 4 Ultrasound-based Localization

In parallel with the vision-based localization capabilities described above, we have been developing an alternative/complementary localization system based on ultrasound sensors [9]. Advantage is taken from the fact that the goalkeeper is near the goal, being easy to obtain an absolute position in the field using visual information. The goalkeeper has one ultrasound emitter that transmits a pulse in a periodic way. The other robots have several ultrasound receivers that cover all the 360º around, and they reply a certain time after receiving the pulse from the goalkeeper. Each robot uses a different reply time in order to implement a time multiplexing of the answers. The goalkeeper knows the reply times of each robot, and in this way it can measure the propagation time of the sound to each robot and from this calculate the distance of each robot. The goalkeeper also has two ultrasonic sensors in order to measure the angle of the signal received from each robot. With these two values, it computes the *x*, *y* coordinates of the robots in the field.

The ultrasound signals are processed using the DSP from Texas Instruments 2812. An initial proof-of-concept prototype was developed using simple algorithms. We use chirps as the transmitted signal and matched filters to detect the pulses. This way we managed to solve some of the problems related to multipath propagation and it is also possible to share the acoustic channel using different chirp signals for each robot. The first field experiments showed that the system works in real conditions measuring the coordinates of the robots with good accuracy. The system has full room to improve the accuracy of the measures by only changing the signal processing algorithms.

## 5 Monitoring framework for multi-process/multi-agent systems

Cambada robots run several processes and at the same time they interact with each other. They operate autonomously, taking many decisions per second based on sensory information that changes dynamically and on shared information that is also subject to frequent changes. It is very hard to follow the robot’s reasoning based only on the external observation of its behavior. To aid this tuning and debugging process, a framework was developed to allow the visualization of the robots reasoning and synchronize it with the observed robot’s behavior.

Several constraints must be considered. The robot is executing several processes and in certain situations we should tune the behavior of the team as a whole instead of tuning individual robots. The framework is prepared to provide high-level information to the developer, useful for online observation, individual and team behavior tuning.

During execution, robots may send information to a socket or to a local logfile. The following debug data is attached to every item of information:

– Timestamp: Used to timeline the sequence and to synchronize information from several sources;
– Category: A tree of categories may be defined to better organize and visualize information. A certain tree or subtree of categories can be hidden/displayed;
– Level of detail: Useful to truncate the visualization at a certain level.

Several types of records are allowed, like text, bookmarks, video images, etc. To synchronize logfiles from different robots two options are available: Use of the regular clock of the PC with the inclusion of a NTP server in the team's base station PC and NTP clients in the robots or the use of the RTAI distributed clock available from the RTAI layer in Linux.

For file processing and reading, several features were implemented:

– Multiple file opening and managing;
– Time based interlace of records from the logfiles; this gives the user the feeling of one big logfile and allows to navigate the data on a unique time line.

An application is being developed for reading and analyzing logfiles.

## References

1. Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes (2004) Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences -- ISCIS 2004: 19th International Symposium, Proceedings*, Aykanat, Cevdet; Dayar, Tugrul; Korpeoglu, Ibrahim (Eds.), Lecture Notes in Computer Science, Vol. 3280, p. 876-886.
2. Almeida, L., J.L. Azevedo, P. Bartolomeu, E. Brito, M.B. Cunha, J.P. Figueiredo, P. Fonseca, C. Lima, R. Marau, N. Lau, P. Pedreiras, A. Pereira, A. Pinho, F. Santos, L. Seabra Lopes, J. Vieira (2004) *CAMBADA: Team Description Paper, RoboCup Symposium: Papers and Team Description Papers* [CD].
3. Almeida, L., P. Pedreiras and J.A. Fonseca (2002) "FTT-CAN: Why and How", *IEEE Tran. Industrial Electronics*.
4. Bartolomeu, P., L. Seabra Lopes, N. Lau, A. Pinho, L. Almeida (2005) Integração de Informação na Equipa de Futebol Robótico CAMBADA, *Electrónica e Telecomunicações*, vol. 4 (4), Universidade de Aveiro, Portugal, p. 467-477.
5. Carter, B., et al. (2002) "Mechanical Design and Modeling of an Omni-directional RoboCup Player", *RoboCup-2001*, A. Birk, et al (edrs), Springer Verlag.
6. Kopetz, H. (1997) Real-Time Systems Design Principles for Distributed Embedded Applications, Kluwer.
7. Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos (2005) Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques, *RoboCup Symposium: Papers and Team Description Papers* [CD], to appear in I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, editors, *RoboCup-2005: Robot Soccer World Cup IX*, LNAI, Springer, 2006.
8. Santos, F., L. Almeida, P. Pedreiras, L. Seabra Lopes, T. Facchinetti (2004) An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among, Mobile Autonomous Agents, *Proc. WACERTS'2004, Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems* (in conjunction with RTSS 2004), Lisboa, Portugal.
9. Vieira, J.M.N., S.I. Lopes, C.C. Bastos and P.N. Fonseca (2005) "Sistema de Localização Mútua para Robots Utilizando Ultra-Sons", *JETC05*, ISEL, Lisboa, Portugal.