CAMBADA'2007: Team Description Paper

J. L. Azevedo, N. Lau, G. Corrente, A. Neves, M. B. Cunha, F. Santos,
A. Pereira, L. Almeida, L. S. Lopes, P. Pedreiras, J. Vieira,
P. Fonseca, D. Martins, N. Figueiredo, J. Puga, J. Taborda

Transverse Activity on Intelligent Robotics
IEETA/DETI – 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'2007. The robots have been developed from scratch in the last four years and, unlike other approaches, using home-made mechanical parts and basic electronic modules. Previous experience of some elements of the team in the RoboCup Simulation League has been highly relevant particularly in the design of the high-level coordination and control framework.

1 Introduction

CAMBADA[1] is the RoboCup middle-size league soccer team of the University of Aveiro, Portugal. This project started officially in October 2003 and, since then, the team has participated in three RoboCup competitions, namely, RoboCup'2004, RoboCup'2006, DutchOpen' 2006, and in the last three editions of the Portuguese Robotics Festival (Robotica2004, Robotica2005 and Robotica2006).

This paper describes the current development stage of the team and is organized as follows: Section 2 briefly presents the robot platform. Section 3 describes the general architecture of the robots focusing both on low-level control hardware aspects and on the general software architecture. Section 4 presents the current version of the vision system. Section 5 briefly describes the high-level coordination and control framework. Finally, section 6 concludes the paper.

2 Robot Platform

The CAMBADA robots were designed and completely built in-house. Each robot is built upon a circular aluminum chassis (with roughly 485 mm diameter), which supports three independent motors (allowing for omnidirectional motion), an electromagnetic kicking device and three NiMH batteries. The remaining parts of the robot are placed in three higher layers, namely: the first layer upon the chassis is used to place all the electronic modules such as motor controllers; the second layer

[1] CAMBADA is an acronym of *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*.

contains the PC (currently a 12" notebook based on an Intel Core2Duo processor); finally on the top of the robots stands an omnidirectional vision system consisting of a standard low cost camera and an hyperbolic mirror (AIS Fraunhofer-Gesellschaft).

The mechanical structure of the robot is highly modular and was designed to facilitate maintenance. It is mainly composed of two tiers: 1) the *mechanical* section that includes the major mechanical parts attached to the aluminum plate (e.g. motors, kicker, batteries); 2) the *electronic* section that includes control modules, the PC and the vision system. These two sections can be easily separated from each other, allowing an easy access both to the mechanical components and to the electronic modules.

Fig. 1. The CAMBADA robot

3 General Architecture of the Robots

The general architecture of the CAMBADA robots has been described in [1], [2]. Basically, the robots architecture is centered on a main processing unit that is responsible for the higher-level behavior coordination, i.e. the coordination layer. This main processing unit (a PC) processes visual information gathered from the vision system, executes high-level control functions and handles external communication with the other robots. This unit also receives sensing information and sends actuating commands to control the robot attitude by means of a distributed low-level sensing/actuating system. The PC runs the Linux operating system over the RTAI (Real-Time Applications Interface [6]) kernel, which provides time-related services, namely periodic activation of processes, time-stamping and temporal synchronization. The communication among team robots uses an adaptive TDMA transmission control protocol [3], on top of IEEE 802.11b, that reduces the probability of transmission collisions between team mates thus reducing the communication latency.

The low-level sensing/actuation system (Fig. 2) is implemented through a set of microcontrollers interconnected by means of a network. For this purpose, Controller Area Network (CAN) [5], 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) [4],[8]. The low-level sensing/actuation system executes four main functions, namely, Motion control, Odometry, Kicking and System monitoring. The Motion control function provides holonomic motion using 3 DC motors. The Odometry function combines the encoder readings from the 3 motors and provides coherent robot displacement information that is then sent to the coordination layer. The Kick function includes the control of an electromagnetic kicker and of a ball handler to dribble the ball. Finally, the System monitor function monitors the robot batteries as well as the state of all nodes in the low-level layer.

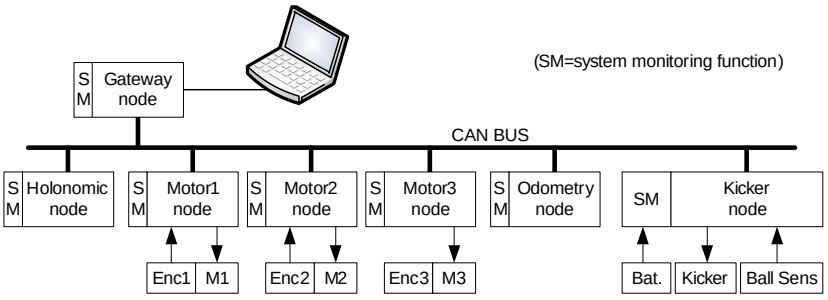

Fig. 2. The CAMBADA hardware architecture.

The low-level control layer connects to the coordination layer through a gateway, which filters interactions within both layers, passing through the information that is relevant across the layers, only.

3.1 Hardware

The low-level layer has a set of nodes, built around a common module, using specialized interfacing to the robot I/O devices. These nodes are interconnected with a CAN network operating at a bit rate of 250Kbps. A gateway interconnects the CAN network to the PC at the high-level layer either through a serial port or a USB port, operating at 115Kbaud in any case.

All modules are based on the same underlying hardware, e.g. a PIC18Fxx8 Microchip [7] microcontroller (@40MHz, i.e., 10 MIPS) which, along with a set of useful peripherals, such as timers, PWM generators, analog to digital converter and serial communications, also integrates a CAN controller. The basic structure of every module includes the CAN port to connect to the network and also includes a 115 Kbps RS232 serial port, which is useful both to program the module firmware and for debugging purposes.

One important characteristic of the CAMBADA hardware design is the galvanic decoupling between the *logic* blocks and the *power* blocks carried out through opto-couplers and/or isolation amplifiers. Along with improved reliability of the whole system it prevents serious damages in expensive equipment (such as the notebook in the high-level layer) whenever any electric problem occurs in the *power* block. The

drawback of this solution is the need of an extra battery for the *logic* part of the system.

The main functions implemented in the low-level layer are described in the following.

Motion control

The robot holonomic motion is obtained combining the speed of 3 DC motors (24V-150W), each with its own speed controller. Each of these controllers is a distinct module of the whole distributed architecture implementing a PI closed loop speed control. It takes as inputs the motor shaft displacement, obtained through a quadrature incremental optical shaft encoder coupled to the motor, and the speed set-point. The computation of the three set-points needed to obtain a coherent robot motion is carried out by another module called *holonomic*. It receives the robot velocity vector (speed, direction and heading) from the higher-level (through the gateway) and translates it into individual set-points that are then sent to each motor controller via CAN messages.

Odometry

The odometry function of the robot is accomplished through the combination of 4 basic functions: the reading of the 3 encoders plus their combination to generate coherent displacement information (Δx, Δy, $\Delta\theta$). The reading of each encoder is naturally allocated to each motor module, using the same readings as those used by the speed feedback control. The combination of the readings is carried out in a specific module, the *odometry node*, which receives the encoder readings from the motors and sends the results to the gateway via CAN messages.

Kicking control

The kicking system is based on an electromagnetic kicker that has been developed by the team for these robots. It allows the higher-level coordination functions to choose one of two kicking modes: direct shooting or lofted kick. Effective control of the kick power is also implemented. The kicking system also includes two IR sensors implemented as an IR barrier which is used to detect the ball when it is in the kicking position, thus avoiding false triggering; and a short distance IR sensor (less than 50 cm) which can be used, in addition to visual information, to determine more precisely the distance between the front of the robot and the ball.

Another feature implemented in this module is an active ball-handler system whose purpose is to dribble the ball throughout the game field in accordance with the RoboCup MSL rules. It is implemented as a quadrature incremental encoder, to measure the ball movement thus providing ball rotation feedback control.

System monitoring

This functionality has two main purposes: measure batteries voltage and monitor modules run-time status. The latter requires this function to be present in all modules, tracking reset situations, namely power-up reset, warm reset, brown-out reset (caused by undervoltage spikes) and watchdog reset, as well as answering to *I'm alive* requests issued by the high-level layer. Battery voltage monitoring is implemented in the same module as the kicker, since it already includes specific voltage monitoring hardware. The battery monitoring function measures, in real-time, the voltage of the

three NiMH batteries used in the robot, namely 2x12V for the *power* blocks of motor controllers and kicker, plus a 9.6V for the *logic* blocks.

The information gathered by the system monitoring function, in all nodes, is sent to the high-level layer for remote monitoring purposes.

3.2 Software

The software system in each robot is distributed among the various computational units. High level functions run on the PC, while low level functions run on the microcontrollers. A cooperative sensing approach based on a Real-Time Database (RTDB) [1], [3], [9] has been adopted. The RTDB is a data structure where the robots 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 robot. This includes information coming from the vision system and odometry information coming from the low-level layer, both stored in a Local Area of the RTDB. 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 team-mates, which is updated by a process that handles the communication with the other robots via an IEEE 802.11b wireless connection. The RTDB is then used by another set of processes that define the specific robot behavior for each instant, generating commands that are passed down to the low-level control layer.

4 Vision System

The current version of the vision system is based on a catadioptric configuration implemented with a low cost Fire-wire web-camera (BCL 1.2 Unibrain camera with a ¼" CCD sensor and a 3.6mm focal distance lens) and a hyperbolic mirror. The camera delivers 640x480 YUV images at a rate of 30 frames per second.

The vision software has been implemented following a modular multi-process architecture (Fig. 3).

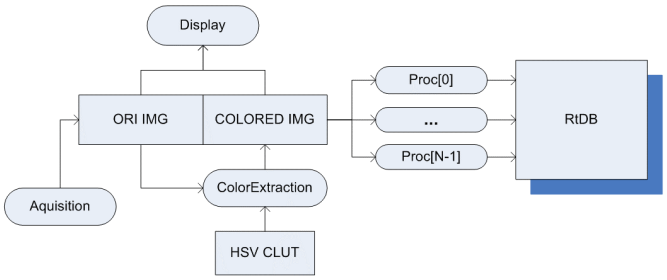

Fig. 3. Architecture of the vision system.

When a new frame is ready to be read, the acquisition process is automatically triggered and the frame is placed in a shared memory buffer. Another process will

then analyze the acquired image for color classification, creating a new one with "color labels" (an 8 bit per pixel image). This image is also placed in the shared image buffer, which is afterwards analyzed by the object detection processes, generically designated by *Proc[x]*, x={0, 1,... N-1}. The output of the detection processes is placed in the real-time database (RTDB) which can be accessed by any other processes on the system, such as the world state update. The activation of the different image-processing processes is carried out by means of a process manager [9].

Image analysis in the RoboCup domain is simplified, since objects are color coded. This fact is exploited by defining color classes, using a look-up-table (LUT) for fast color classification. The table consists of 16777216 entries (24 bits: 8 bits for red, 8 bits for green and 8 bits for blue), each 8 bits wide, occupying 16 MB in total. The classification of a pixel is carried out using its color as an index into the table. The color calibration is done in HSV (Hue, Saturation and Value) color space. In the current setup the image is acquired in RGB or YUV format and is then converted to HSV using an appropriate conversion routine.

The image processing software uses radial search lines to analyze the color information. The regions of the image that have to be excluded from analysis (such as the robot itself, the sticks that hold the mirror and the areas outside the mirror) are ignored through the use of a previously generated image mask.

The objects of interest (a ball, two goals, obstacles and the green to white transitions) are efficiently detected through algorithms that, using the color information collected by the radial search lines, calculate the object position and/or their limits in an angular representation (distance and angle). The green/white detected transition points, that are at a distance smaller than a predefined value, are stored in the RTDB for latter use by the robot self-localization process.

The relationship between image pixels and real world distances is obtained through an analytical method developed by the team (to be published soon) that explores a back-propagation ray-tracing approach and the mathematical properties of the mirror surface.

5 High-level coordination and control

The high-level decision is built around three main modules: sensor fusion, basic behaviors and high-level decision and cooperation. The objective of the sensor fusion module is to gather the noisy information from the sensors and from other robots and update the World State database that will be used by the high-level decision and coordination. The basic behaviors module provides the set of primitives that the higher-level decision modules use to control the robot. It is essential to provide those modules with a good set of alternatives, each of which should be as efficient as possible. The high-level decision module is responsible for the analysis of the current situation and for the performing of decision-making processes carried out by each player in order to maximize, not only the performance of its actions, but also the global success of the team.

The sensor fusion module has recently been redesigned, in what concerns its interface with the other modules, in order to get a common view over all the sensor measures. Now all sensors write into adequate structures, but only the sensor fusion module is allowed to update the World State. A recent, and very important,

development as been the integration into the sensor fusion module of a self-localization lines-based engine, based-on the one described in [12], that allows a high level of confidence in the robots estimated self-position.

The new design of the vision system, which is now omnidirectional, has allowed the development of a new set of basic behaviors. The previous vision system was based on two cameras, one facing the field orthogonally, enabling the capture of a 360 degrees view around the robot with roughly 1m radius, and the other pointing forward in the direction of the front of the robot. With that vision system the robot could sense far objects in front of it, but had a very limited view of its surrounding area in all other directions. As a consequence most of the movements had to be done with the robot turned to the target point. Using the new vision system, the robot can accurately move towards any given point at any given orientation. Several experiments of different alternatives have been carried out and a new set of optimized basic behaviors is now available.

The high-level decision module currently uses state-machine based modeled roles that switch the basic behavior of the robot in accordance with the current situation and the previous state. Coordination is achieved by the definition of formations of different roles [11] and by a higher-level module where role switching is performed. The concepts of roles, formations and set-plays have previously been used in the RoboCup in some Simulation and Middle-Size teams. The coordination is in the process of integrating the information coming from the new self-localization engine, which allows the use of coordination techniques like SBSP [10]. In some cases, such as kick-ins or corners, specific set-plays are activated where a coordinated and synchronized set of basic behaviors is performed by all team robots.

6 Conclusion

This paper described the current development stage of the CAMBADA robots. Since the last submission of qualification material (in January/2006) several major improvements have been carried out, namely: the implementation of a new vision system based on a single camera in a catadioptric configuration; the development of a new tool to calibrate image colors based on the HSV color space; the implementation of vision software processing based on radial sensors; the development of an analytical method to get the relationship between image pixels and real world distances; the implementation and integration of a self-localization algorithm; the re-design of the higher-level coordination and control software; a new kicking device with kick mode selection and power control; the replacement of the lead-acid batteries by smaller and lighter NiMH which allowed for, roughly, 30% robot weight reduction.

References

1. L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, L.S.Lopes, "Coordinating distributed autonomous agents with a real-time database: The CAMBADA project". ISCIS'04, 19th International Symposium on Computer and Information Sciences. 27-29 October 2004, Kemer - Antalya, Turkey.
2. V. Silva, R. Marau, L. Almeida, J. Ferreira, M. Calha, P. Pedreiras, J. Fonseca, "Implementing a distributed sensing and actuation system: The CAMBADA robots case study", IEEE ETFA 2005, Catania, Italy. September 2005.
3. F. Santos, L. Almeida, P. Pedreiras, L.S.Lopes, T. Facchinnetti, "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication Among Mobile Computing Agents". WACERTS 2004, Workshop on Architectures for Cooperative Embedded Real-Time Systems (satellite of RTSS 2004). Lisboa, Portugal, 5-8 Dec. 2004.
4. Almeida L., P. Pedreiras, J. A. Fonseca, "The FTT-CAN Protocol: Why and How, IEEE Transactions on Industrial Electronics", 49(6), December 2002.
5. Controller Area Network - CAN2.0, Technical Specification, Robert Bosch, 1992.
6. RTAI for Linux, available at http://www.aero.polimi.it/~rtai/
7. Microchip website, available at www.microchip.com
8. Calha, M. J., J. A. Fonseca, “Approaches to the FTT-based scheduling of tasks and messages”, Proceedings of the 5th IEEE International Workshop on Factory Communication Systems (WFCS’04), Vienna, Austria, Sep/2004.
9. Pedreiras, P., F. Teixeira, N. Ferreira, L. Almeida, A. Pinho, F. Santos, "Enhancing the reactivity of the vision subsystem in autonomous mobile robots using real-time techniques", RoboCup Symposium: Papers and Team Description Papers, RoboCup-2005: Robot Soccer World Cup IX, Lecture Notes in Artificial Intelligence, Springer, 2006.
10. Reis, L.P. and N. Lau, "FC Portugal Team Description: RoboCup 2000 Simulation League Champion", RoboCup-2000, P. Stone, et al. (edtrs), p. 29-40, Springer Verlag, 2001.
11. Stone, P. and M. Veloso, "Task Decomposition, Dynamic Role Assignment and Low Bandwidth Communication for Real Time Strategic Teamwork", *Artificial Intelligence*, 110 (2), p. 241-273, 1999.
12. Martin Lauer, Sascha Lange and Martin Riedmiller, “Calculating the perfect match: An efficient and accurate approach for robot self-localisation”, In A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, editors, RoboCup 2005: Robot Soccer World Cup IX, LNCS. Springer, 2005