An Omnidirectional Vision System for Soccer Robots

António J. R. Neves*, Gustavo A. Corrente and Armando J. Pinho

Dept. de Electrónica e Telecomunicações / IEETA
Universidade de Aveiro, 3810–193 Aveiro, Portugal
an@ua.pt, gustavo@ua.pt, ap@ua.pt

Abstract. This paper describes a complete and efficient vision system developed for the robotic soccer team of the University of Aveiro, CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture). The system consists on a firewire camera mounted vertically on the top of the robots. A hyperbolic mirror placed above the camera reflects the 360 degrees of the field around the robot. The omnidirectional system is used to find the ball, the goals, detect the presence of obstacles and the white lines, used by our localization algorithm. In this paper we present a set of algorithms to extract efficiently the color information of the acquired images and, in a second phase, extract the information of all objects of interest. Our vision system architecture uses a distributed paradigm where the main tasks, namely image acquisition, color extraction, object detection and image visualization, are separated in several processes that can run at the same time. We developed an efficient color extraction algorithm based on lookup tables and a radial model for object detection. Our participation in the last national robotic contest, ROBOTICA 2007, where we have obtained the first place in the Medium Size League of robotic soccer, shows the effectiveness of our algorithms. Moreover, our experiments show that the system is fast and accurate having a maximum processing time independently of the robot position and the number of objects found in the field.

Keywords: Robotics; robotic soccer; computer vision; object recognition; omnidirectional vision; color classification.

1 Introduction

After Garry Kasparov was defeated by IBM's Deep Blue Supercomputer in May 1997, forty years of challenge in the artificial intelligence (AI) community came to a successful conclusion. But it also was clear that a new challenge had to be found.

"By mid-21st century, a team of fully autonomous humanoid robot soccer players shall win the soccer game, complying with the official rules of the FIFA, against the winner of the most recent World Cup." This is how the ultimate goal was stated by the RoboCup Initiative, founded in 1997, with the aim to foster the development of artificial intelligence and related field by providing a standard problem: robots that play soccer.

It will take decades of efforts, if not centuries, to accomplish this goal. It is not feasible, with the current technologies, to reach this goal in any near term. However,

---

this goal can easily create a series of well directed subgoals. The first subgoal to be accomplished in RoboCup is to build real and simulated robot soccer teams which play reasonably well with modified rules. Even to accomplish this goal will undoubtedly generate technologies with impact on broad range of industries.

One problem domain in RoboCup is the field of Computer Vision. Its task is to provide basic information that is needed to calculate the behavior of the robots. Especially omnidirectional vision systems have become interesting in the last years, allowing a robot to see in all directions at the same time without moving itself or its camera [13, 12, 10, 11]. Omnidirectional vision is the method used by most teams in the Middle Size League.

The main goal of this paper is to present an efficient vision system for processing the video acquired by an omnidirectional camera. The system finds the white lines of the playing field, the ball, goals and obstacles. The lines of the playing field are needed because knowing the placement of the playing field from the robot's point of view is equal to know the position and orientation of the robot.

For finding the goals, the ball and the obstacles, lines are stretched out radially from the center of the image and, if some defined number of pixels of the respective colors are found, the system saves that position associated to the respective color. For finding the white lines, color transitions from green to white are searched for.

For color classification, the first step of our system, a lookup table (LUT) is used. Our system is prepared to acquire images in RGB 24-bit, YUV 4:2:2 or YUV 4:1:1 format, being necessary only to choose the appropriated LUT. We use the HSV color space for color calibration and classification due to its special characteristics [1].
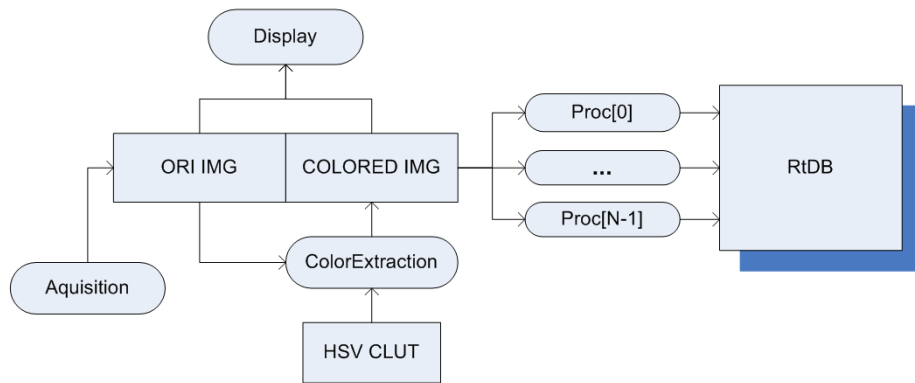

Fig. 1. The architecture of our vision system. It is based on a multi-process system being each process responsible for a specific task.

This paper is organized as follows. In Section 2 we describe the design of our robots. Section 3 presents our vision system architecture, explaining the several modules devel-

oped and how they are connected. In Section 4 we present the algorithms used to collect the color information of the image using radial search lines. In Section 5 we describe how the object features are extracted. Finally, in Section 6, we draw some conclusions and propose future work.

2 Robot overview

CAMBADA players were designed and completely built in-house. The baseline for robot construction is a cylindrical envelope, with 485 mm in diameter. The mechanical structure of the players is layered and modular. The components in the lower layer are the motors, wheels, batteries and an electromechanical kicker. The second layer contains the control electronics. The third layer contains a computer. The players are capable of holonomic motion, based on three omni-directional roller wheels [2].

The vision system consists on a firewire camera mounted vertically on the top of the robots. A hyperbolic mirror placed above the camera reflects the 360 degrees of the field around the robot. This is the main sensor of the robot and it is used to find the ball, the goals, detect the presence of obstacles and the white lines.

The robots computing system architecture follows the fine-grain distributed model [7] where most of the elementary functions 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. Firewire. For this purpose, Controller Area Network (CAN) has been chosen [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) [4–6] 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.

3 Vision system architecture

A modular multi-process architecture was adopted for our vision system (see Fig. 1).

When a new frame is ready to download, one process is automatically triggered and the frame is placed in a shared memory buffer. After that, another process analyzes the acquired image for color classification, creating a new image with "color labels" (an 8 bpp image). This image is also placed in the shared image buffer, which is afterward analyzed by the object detection processes, generically designated by $Proc[x]$, $x = 1, 2, \ldots N$. These applications 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 and position) is placed in the real-time database. This database may be accessed by any other processes in the system, particularly for world state update.

The activation of the distinct image-processing activities is carried out by a process manager. Scheduling of vision related processes relies on the real-time features of the Linux kernel, namely the FIFO scheduler and priorities. 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.
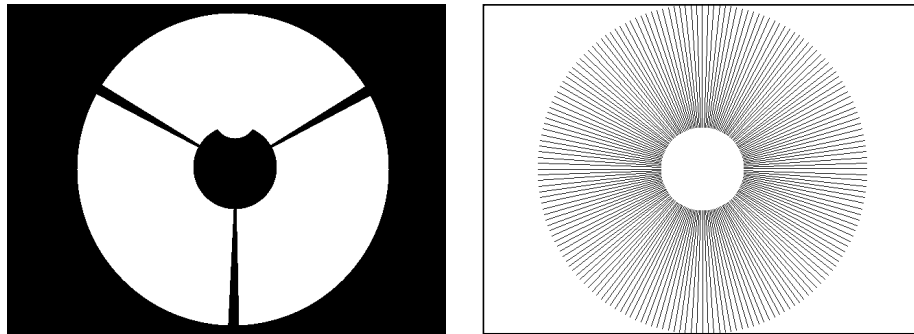
4 Color extraction

Fig. 2. On the left, an example of a robot mask. White points represent the area that will be processed. On the right, the position of the radial search lines.

Image analysis in the RoboCup domain is simplified, since objects are color coded. Black robots play with an orange ball on a green field that has yellow and blue goals and white lines. Thus, a pixel's color is a strong hint for object segmentation. We exploit this fact by defining color classes, using a look-up table (LUT) for fast color classification. The table consists of 16777216 entries (2^{24}, 8 bits for red, 8 bits for green and 8 bits for blue), each 8 bits wide, occupying 16 MB in total. If another color space is used, the table size is the same, changing only the "meaning" of each component. Each bit expresses whether the color is within the corresponding class or not. This means that a certain color can be assigned to several classes at the same time. To classify a pixel, we first read the pixel's color and then use the color as an index into the table. The value (8 bits) read from the table will be called "color mask" of the pixel.

The color calibration is done in HSV (Hue, Saturation and Value) color space due to its special characteristics. In our system, the image is acquired in RGB or YUV format and then is converted to HSV using an appropriate conversion routine.

There are certain regions in the received image that have to be excluded from analysis. One of them is the part in the image that reflects the robot itself. Other regions are the sticks that hold the mirror and the areas outside the mirror. For that, we have an image with this configuration that is used by our software. An example is presented in Fig. 2. The white pixels are the area that will be processed, black pixels will not. With

this approach we can reduce the time spent in the conversion and searching phases and we eliminate the problem of finding erroneous objects in that areas.

To extract the color information of the image we use radial search lines to analyze the color information instead of processing all the image. This approach has two main advantages. First, that of accelerating the process due to the fact that we only process about 30% of the valid pixels. Second, the use of omnidirectional vision difficults the detection of the objects using, for example, their bounding box. In this case, it is more desirable to use the distance and angle. The proposed approach has a processing time almost constant, independently of the information around the robot, being a desirable property in Real-Time Systems. This is due to the fact that the system processes almost the same number of pixels in each frame.

A radial search line is a line that starts in the center of the robot with some angle and ends in the limit of the image (see the image on the right of Fig. 2). They are constructed based on the Bresenham line algorithm [8, 9]. For each search line, we iterate through its pixels to search for two things: transitions between two colors and areas with specific colors.

We developed an algorithm to detect areas of a specific color which eliminates the possible noise that could appear in the image. Each time that a pixel is found with a color of interest, we analyze the pixels that follows (a predefined number) and if we don't find more pixels of that color we "forget" the pixel found and continue. When we find a predefined number of pixels with that color, we consider that the search line has this color.

To accelerate the process of calculating the position of the objects, we put the color information found, in each search line, into a list of colors. We are interested in the first pixel (in the respective search line) where the color was found and the number of pixels with that color found in the search line. Then, using the previous information, we separate the information of each color into sets that we named blobs (see Fig. 3). For each blob some useful information is calculated that will help in the detection of each object:

- average distance to the robot;
- mass center;
- angular width;
- number of pixels;
- number of green pixels between blob and the robot;
- number of pixels after blob.

The algorithm to search for the transitions between green pixels and white pixels is described as follows. If a non green pixel is found, we will look for a small window in the "future" and count the number of non green pixels and the number of white pixels. Next, we look for a small window in the "past" and a small window in the future and count the number of green pixels. If these values are greater than a predefined threshold, we consider this point as a transition. This algorithm is illustrated in Fig. 4.
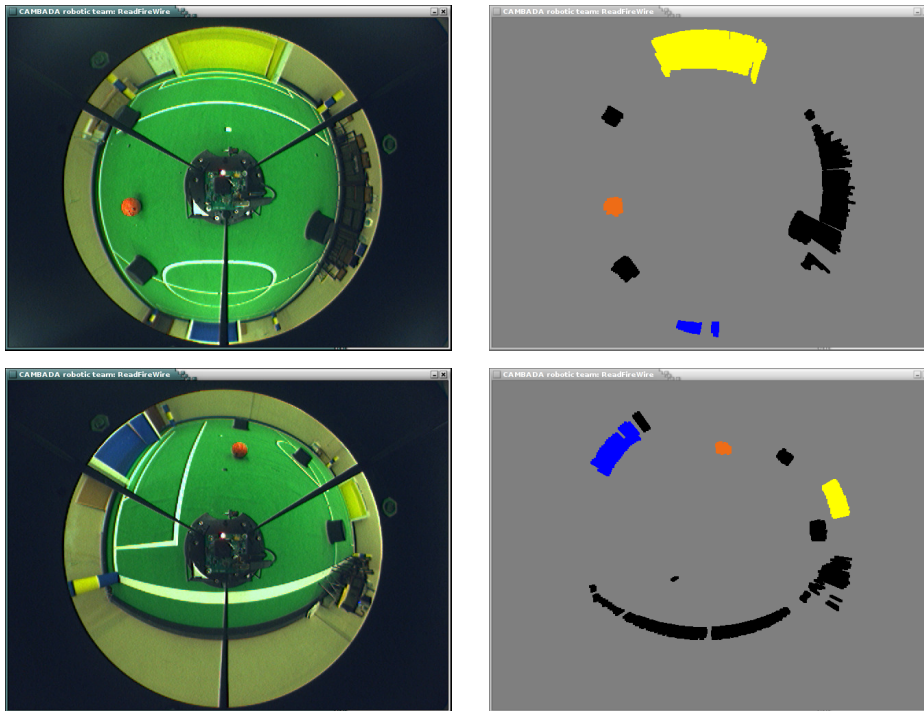
Fig. 3. An example of the blobs found in two images. On the left, the original images. On the right, the blobs found. For each blob, we calculate useful information that is used later to calculate the position of each object.
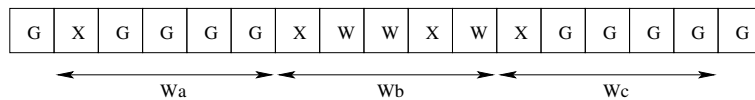
G	X	G	G	G	G	X	W	W	X	W	X	G	G	G	G	G

$$\longleftarrow \text{Wa} \longrightarrow \longleftarrow \text{Wb} \longrightarrow \longleftarrow \text{Wc} \longrightarrow$$

Fig. 4. An example of a transition. "G" means green pixels, "W" means white pixels and "X" means pixels with a color different from green or white.

5 Object detection

The objects of interest that are present in the RoboCup environment are: a ball, two goals, obstacles (other robots) and the green field with white lines. Currently, our system detects efficiently all these objects with a set of simple 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 transition points detected in the color extraction phase are used for the robot localization. All the points detected are sent to the Real-time Database, afterward used by the localization process.

To detect the ball, we use the following algorithm:

1. Separate the orange information into blobs.
2. For each blob, calculate the information described in the previous section.
3. Sort the orange blobs that have some green pixels before or after the blob by descending order, using their number of orange pixels as measure.
4. Choose the first blob as candidate. The position of the ball is the mass center of the blob.

Regarding the goals, we are interested in three points: the center, the right post and the left post. To do that, we use the following algorithm:

1. Ignore the information related to radial search lines which have both blue and yellow information (they correspond to the land marks).
2. Separate the valid blue and yellow information into blobs.
3. Calculate the information for each blob.
4. Sort the yellow and the blue blobs that have some green pixels before the blob by descending order, using their angular width as measure.
5. Choose the first blob as candidate for each goal. Their position is given by the distance of the blob relatively to the robot.
6. The right post and the left post is given by the position of the goal and the angular width of the blob.

Another important information regarding the goals, is the best point to shoot. To calculate it, we split the blob chosen into several slices, and choose the one with most pixels blue or yellow. The best point to shoot is the mass center of the slice chosen.

To calculate the position of the obstacles around the robot, we use the following algorithm:

1. Separate the black information into blobs.
2. If the angular width of one blob is greater than 10 degrees, we split the blob into smaller blobs, in order to obtain better information about obstacles.
3. Calculate the information for each blob.
4. The position of the obstacle is given by the distance of the blob relatively to the robot. We are also interested in the limits of the obstacle and to obtain that we use the angular width of the blob.

In Fig. 5 we present some examples of acquired images and their correspondent segmented images. As we can see, the objects are correctly detected (see the marks in images on the right).

6 Final remarks

This paper presents the omnidirectional vision system that has been developed for the CAMBADA team. We present several algorithms for image acquisition and processing. The experiments already made and the last results obtained in the ROBOTICA 2007 competition prove the effectiveness of our system regarding the object detection and robot self-localization.
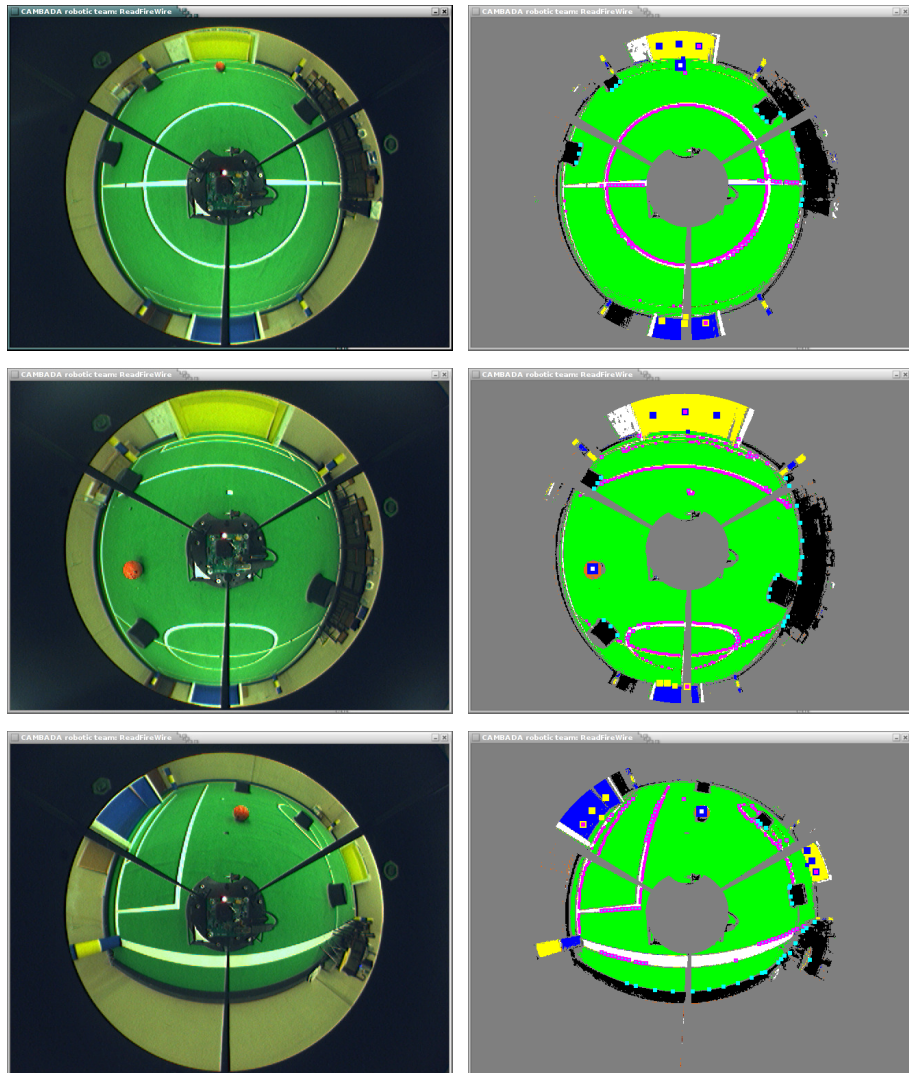
Fig. 5. On the left, examples of original images. On the right, the corresponding processed images. Marks in the blue and yellow goals mean the position of the goal (center) and the possible points to shoot. The mark over the ball points to the mass center. The several marks near the white lines (magenta) are the position of the white lines. The cyan marks are the position of the obstacles.

The objects in RoboCup are color coded. Therefore, our system defines different color classes corresponding to the objects. The 24 bit pixel color is used as an index to a 16 MBytes lookup table which contains the classification of each possible color in a 8 bit entry. Each bit specifies whether that color lays within the corresponding color class.

The processing system is divided in two phases: color extraction, using radial search lines, and object detection, using specific algorithms. The objects involved are: a ball, two goals, obstacles and white lines. The processing time and the accuracy obtained in the object detection confirms the effectiveness of our system.

As future work, we are developing new algorithms for camera and color calibration, in particular autonomous algorithms. Moreover, we are improving the presented algorithms in order to use the shape of the objects instead of using only the color information to improve the object recognition.

References

1. Pedro M. R. Caleiro, António J. R. Neves and Armando J. Pinho, Color-spaces and color segmentation for real-time object recognition in robotic applications, *Revista do DETUA*, Vol. 4, N. 8, Junho 2007, pp. 940-945.
2. Carter, B., *EST LA.*: Mechanical Design and Modeling of an Omni-directional RoboCup Player, *RoboCup-2001*, A. Birk, *et al* (edrs), Springer Verlag.
3. Almeida, L., P. Pedreiras and J.A. Fonseca: FTT-CAN: Why and How, *IEEE Trans. Industrial Electronics*, 2002.
4. Almeida, L., F. Santos, T. Facchinetti, P. Pedreira, V. Silva and L. Seabra Lopes: Coordinating Distributed Autonomous Agents with a Real-Time Database: The CAMBADA Project, *Computer and Information Sciences – ISCIS 2004: 19th International Symposium, Proceedings*, Lecture Notes in Computer Science, Vol. 3280, p. 876-886.
5. 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.
6. Santos, F., L. Almeida, P. Pedreiras, L. Seabra Lopes, T. Facchinetti: 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.
7. Kopets H.: Real-Time Systems Design Principles for Distributed Embedded Applications, Kluwer.
8. Bresenham, J. E.: A linear algorithm for incremental digital display of circular arcs, *CA CM*, 20(2), pp. 100-106, 1977.
9. Bresenham, J. E.: Algorithm for computer control of a digital plotter, *IBM Systems J.*, 4(1), pp.25-30, 1965.
10. P. Heinemann *et al.*: Fast and Accurate Environment Modelling using Omnidirectional Vision, *Dynamic Perception*, pp. 9-14, Infix, 2004.
11. P. Heinemann *et al.*: Tracking Dynamic Objects in a RoboCup Environment - The Attempto Tübingen Robot Soccer Team, *RoboCup-2003: Robot Soccer World Cup VII*, LNCS, Volume 3020, Springer, 2003.
12. Jose Gaspar, Niall Winters, Etienne Grossmann, Jose Santos-Victor: Toward Robot Perception using Omnidirectional Vision, *In Innovations in Machine Intelligence and Robot Perception*, Springer-Verlag, 2004.
13. Jan Hoffmann *et al.*: A vision based system for goal-directed obstacle avoidance, *RoboCup-2004: Robot Soccer World Cup VIII*, LNCS, Springer, 2004.