UAVision: A Modular Time-Constrained Vision Library for Soccer Robots

Alina Trifan(✉), António J.R. Neves, Bernardo Cunha, and José Luís Azevedo

IRIS Group, DETI / IEETA, University of Aveiro, 3810–193 Aveiro, Portugal
{alina.trifan,an,jla}@ua.pt, mbc@det.ua.pt

Abstract. The game of soccer is one of the main focuses of the RoboCup competitions, being a fun and entertaining research environment for the development of autonomous multi-agent cooperative systems. For an autonomous robot to be able to play soccer, first it has to perceive the surrounding world and extract only the relevant information in the game context. Therefore, the vision system of a robotic soccer player is probably the most important sensorial element, on which the acting of the robot is fully based. In this paper we present a new modular time-constrained vision library, named UAVision, that allows the use of video sensors up to a frame rate of 50 fps in full resolution and provides accurate results in terms of detection of the objects of interest for a robot playing soccer.

1 Introduction

The research area of robotic vision is greatly evolving by means of international competitions such as those promoted and organized once per year by the RoboCup Federation. The RoboCup initiative, through competitions like RoboCup Robot Soccer, RoboCup Rescue, RoboCup@Home and RoboCup Junior, is designed to meet the need of handling real world complexities, while maintaining an affordable problem size and research cost. It offers an integrated research task covering the broad areas of artificial intelligence, computer vision and robotics.

The soccer game in the RoboCup Middle Size League (MSL) is a standard real-world test for autonomous multi-robot systems. In this league, 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 [1]. The environment of this league is not as restricted as in the others and the pace of the game is faster than in any other league (currently with robots moving with a speed of 4 m/s or more and balls being kicked with a velocity of more than 10 m/s), requiring fast reactions from the robots. In terms of color coding, in the fully autonomous MSL the field is still green, the lines of the field and the goals are white and the robots are mainly black. The two teams competing are wearing cyan and magenta markers. For the ball color, the only rule applied is that the surface of the ball should be 80 % of a certain color, which is usually decided before a competition. The colors of the objects of interest are important hints

R.A.C. Bianchi et al. (Eds.): RoboCup 2014, LNAI 8992, pp. 490–501, 2015.
DOI: 10.1007/978-3-319-18615-3_40

for the object detection, relaxing thus the detection algorithms. Many teams are currently taking their first steps in 3D ball information retrieving [2,3]. There are also some teams moving their vision systems algorithms to VHDL based algorithms taking advantage of the FPGAs versatility [2]. Even so, for now, the great majority of the teams base their image analysis in color search using radial sensors [4–6].

In this paper we present a library for color-coded object detection, named UAVision, that is currently being used by the robots of the team CAMBADA, participating in the Middle Size League. The design of the library follows a modular approach as it can be stripped down into several independent modules (that will be presented in the following sections). Moreover, the architecture of our software is of the type "plug and play". This means that it offers support for different vision sensors technologies and that the software created using the library is easily exportable and can be shared between different types of vision sensors. These facts, on the other hand, make it appropriate for being used by robots in all other leagues. Another important aspect of our library is that it takes into consideration time constraints. All the algorithms behind this library have been implemented focusing on maintaining the processing time as low as possible. Realtime processing means to be able to complete all the vision dependant tasks within the limits of the frame rate.

The vision system for color-coded object detection within the RoboCup soccer games of Middle Size League that we have implemented using the UAVision library can work with frame rates up to 50 fps using a resolution of 1024×1024 pixels, both in Bayer, RGB or YUV color modes. Detailed processing time obtained will be presented in Sect. 3. Moreover, we provide experimental results showing the difference of working with the different frame rates in terms of the delay between the perception and the action. As far as we know, there is no previous published work that presents so detailed information about this issue.

The library that we are proposing comes as a natural development of the work already presented within the RoboCup community. After having implemented vision systems for robotic soccer players that perform both in the Standard Platform League [7] and Middle Size League [8–10], we are proposing this new cross-library that can be used by robots whose architecture might be different, but the goal remains the same: the game of soccer. We consider our work an important contribution for the RoboCup Soccer community since so far, there are no machine vision libraries used for the games of soccer that take into consideration time constraints. UAVision aims at being an open-source free library that can be used for robotic vision applications that have to deal with time constraints as are the RoboCup competitions. Moreover, we made publicly available the video sequences used in the experimental results of this paper, both in Bayer and RGB color modes, so that other researchers can reproduce our results and test their own algorithms.

This paper is structured in five sections, the first of them being this introduction. Section 2 describes the modules of the vision library. Section 3 presents the results that have been achieved using the library in the MSL robots. Section 4

concludes the paper and future lines of research are highlighted. Finally, in Acknowledgements section the institutions that have supported this work are acknowledged.

2 Library Description

The library that we are presenting is intended for the development of artificial vision systems for the detection of color-coded objects, being the robotic soccer the perfect application for its usage. The library contains software for image acquisition from video cameras supporting different technologies, for camera calibration and for blob formation, which stands at the basis of the object detection.

2.1 Image Acquisition

UAVision provides the necessary software for accessing and capturing images from three different camera interfaces, so far: USB cameras, Firewire cameras and Ethernet cameras. For this purpose, the Factory Design Pattern [11] has been used and a factory called "Camera" has been implemented. The user can choose from these three different types of cameras in the moment of the instantiation. An important aspect to be mentioned is that UAVision uses some of the basic structures from the core functionality of OpenCV library: the *Mat* structure as a container of the frames that are grabbed and the *Point* structure for the manipulation of points in 2D coordinates. Images can be acquired in the YUV, RGB or Bayer color format.

The module of Image Acquisition also provides methods to convert images between the most used color spaces: RGB to HSV, HSV to RGB, RGB to YUV, YUV to RGB, Bayer to RGB and RGB to Bayer.

2.2 Camera Calibration

The correct calibration of all the parameters related to the system is very important in any vision system. The module of camera calibration includes algorithms for calibration of the intrinsic and extrinsic camera parameters, the computation of the inverse distance map, the calibration of the colormetric camera parameters and the detection of the mirror, robot center and the definition of the regions of the image that do not have to be processed.

The result of the vision system calibration can be stored in a configuration file which contains four main blocks of information: camera settings, mask, map and color ranges. The mask is a binary image representing the areas of the image that do not have to be processed, since they contain only parts of the body of the robot, which are not relevant for the object detection. By ignoring these areas of the image, both the noise in the image and the processing time can be reduced. The map, as the name suggests, is a matrix that represents the mapping between pixel coordinates and real world coordinates.

The camera settings block is where the basic information is registered. Among others, these include the resolution of the image acquired, the Region of Interest regarding the CCD or CMOS of the camera and colormetric parameters, among others.

The color ranges block contains the color regions for each color of interest (at most 8 different colors as we will explain later) in a specific color space (ex. RGB, YUV, HSV, etc.). In practical means, it contains the lower and upper bounds of each one of the three color components for a specific color of interest.

The UAVision library contains algorithms for the self-calibration of most of the parameters described above, including some algorithms developed previously within our research group, namely the algorithm described in [8] for the automatic calibration of the colormetric parameters and the algorithms presented in [1,9] for calibration of the intrinsic and extrinsic parameters of catadioptric vision systems used to generate the inverse distance map. For the calibration of the intrinsic and extrinsic parameters of a perspective camera, we have used and implemented the algorithm for the "chessboard" calibration, presented in [12].

2.3 Color-Coded Object Detection

The color-coded object detection is composed by four sub-modules that are presented next.

- **Look-Up Table**

For fast color classification, color classes are defined through the use of a look-up table (LUT). A LUT represents a data structure, in this case an array, used for replacing a runtime computation by a basic array indexing operation.

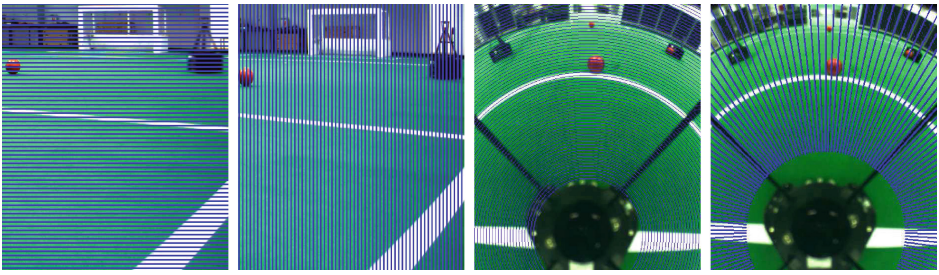
This approach has been chosen in order to save significant processing time. The images can be acquired in the RGB, YUV or Bayer format and they are converted to an index image (image of labels) using an appropriate LUT for each one of the three possibilities.

The table consists of 16,777,216 entries (2^{24}, 8 bits for R, 8 bits for G and 8 bits for B) with one byte each. The table size is the same for the other two possibilities (YUV or Bayer), but the meaning of each of the components changes. Each bit in the table entries expresses if one of the colors of interest (white, green, blue, yellow, orange, red, blue sky, black, gray - no color) is within the corresponding class or not. A given color can be assigned to multiple classes at the same time. For classifying a pixel, first the value of the color of the pixel is read and then used as an index into the table. The 8-bit value then read from the table is called the "color mask" of the pixel. It is possible to perform image subsampling in this stage in systems with limited processing capabilities in order to reduce even more the processing time. The color classification is only applied to the valid pixels if a mask exists.

- **Scanlines**

To extract color information from the image we have implemented three types of search lines, which we also call scanlines: radial, linear (horizontal or vertical) and circular. They are constructed once, when the application starts, and saved

in a structure in order to improve the access to these pixels in the color extraction module. This approach is extremely important for the reduction of processing time. In Fig. 1 the three different types of scanlines are illustrated.



a) b) c) d

Fig. 1. Examples of different types of scanlines: (a) horizontal scanlines; (b) vertical scanlines; (c) circular scanlines; (d) radial scanlines.

- **Run Length Encoding (RLE)**

For each scanline, an algorithm of Run Length Encoding is applied in order to obtain information about the existence of a specific color of interest in that scanline. To do this, we iterate through its pixels to calculate the number of runs of a specific color and the position where they occur. Moreover, we extended this idea and it is optional to search, in a window before and after the occurrence of the desired color, for the occurrence of other colors. This allows the user to determine both color transitions and color occurrences using this approach.

When searching for run lengths, the user can specify the color of interest, the color before, the color after, the search window for these last two colors and three thresholds that can be used to determine the valid information.

As a result of this module, we obtain a list of positions in each scanline and, if needed, for all the scanlines, where a specific color occurs, as well as the amount of pixels in each occurrence (Fig. 2).

- **Blob Formation**

To detect objects with a specific color in a scene, we have to be able to detect regions in the image with that color, usually named blobs, and validate those blobs according to some parametric and morphological features, namely area, bounding box, solidity, skeleton, among others. In order to construct these regions, we use information about the position where a specific color occurs based on the Run Length module previously described (Fig. 2).

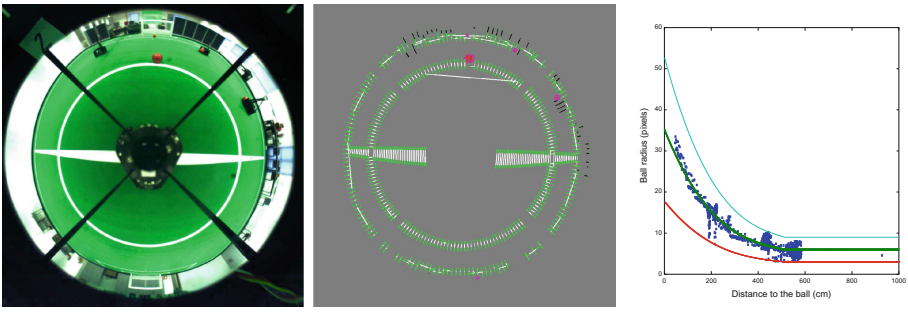
We iterate through all the run lengths of a specific color and we apply an algorithm of clustering based on the euclidean distance. The parameters of this clustering are application dependent. For example, in a catadioptric vision system, the distance in pixels to form blobs changes radially and non-linearly regarding the center of the image.

While the blob is being built, its descriptor is being updated. The description of the blobs currently calculated are, to name a few, center, area, width/height relation, solidity, etc.

- **Object Detection**

The last step of the vision pipeline is the decision regarding whether the colors segmented belong to an object of interest or not. In the vision system developed for the CAMBADA team using the proposed library, the white and black points that have been previously run-length encoded are passed directly to higher level processes, where localization based on the white points and obstacle avoidance based on the black points are performed.

For the ball detection, the blobs that are of the color of the ball have to meet the following validation criteria before being labelled as ball. First, a mapping function that has been experimentally designed is used for verifying a size-distance from the robot ratio of the blob (Fig. 2(c)). This is complemented by a solidity measure and a width-height ratio validation, taking into consideration that the ball has to be a round blob. The validation was made taking into consideration the detection of the ball even when it is partially occluded.



a) b) c)

Fig. 2. On the left, an image captured using the Camera Acquisition module of the UAVision library. In the center, the run length information annotated. On the right, illustration of the radius (in pixels) of the ball relative to the distance (in centimeters) from the robot at which it is found. The blue marks represent the measures obtained, the green line the fitted function and the cyan and red line the upper and lower bounds considered for validation (Color figure online).

3 Experimental Results

The UAVision library is currently used by the MSL team of robots CAMBADA team from University of Aveiro. These robots are completely autonomous, able to perform holonomic motion and are equipped, in terms of hardware, with a catadioptric vision system that allows them to have omnidirectional vision [10]. The architecture of the vision system is presented in Fig. 3.

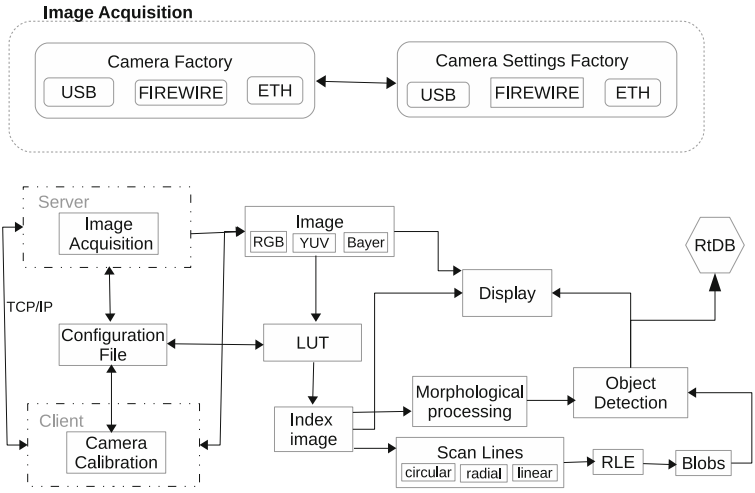
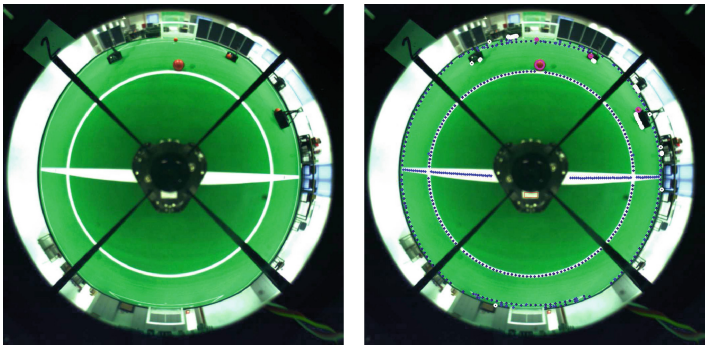


Fig. 3. Software architecture of the vision system developed based on the UAVision library.

The pipeline of the object detection procedure is the following: after having an image acquired, using a LUT previously built, the original image is transformed into an image of labels. This image of color labels, also denominated in our software by index image, will be the basis of all the processing that follows. The index image is scanned using one of the three types of scanlines previously described (circular, radial or linear) and the information about transitions between the colors of interest is run length encoded. Transitions between green and other colors of interest (white, ball color, black) are searched in order to ensure that the objects detected are inside the field area. Blobs are formed by merging adjacent RLEs of the ball color. The blob is then labeled as ball if the blob area/distance from the robot respects a certain function that has been experimentally determined (see Fig. 2(c)). Moreover, the width/height relation and solidity are also used for ball validation. If a given blob passes the validation criteria, its center coordinates will be passed to higher-level processes and shared on a Real-time Database (RtDB) [13]. For the obstacles and line detections, the coordinates of the detected points of interest are passed to higher-level processes through the RtDB.

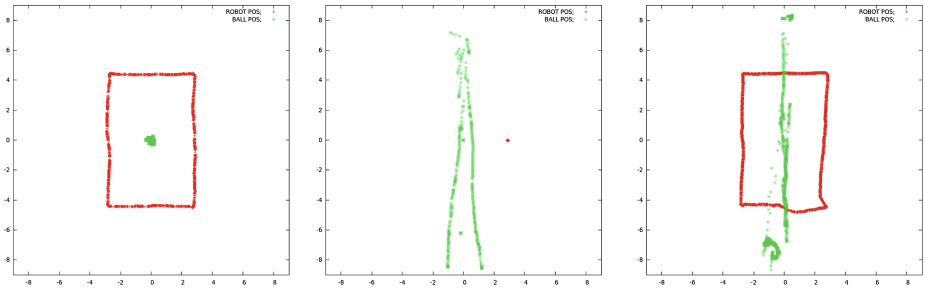
A visual example of the detected objects in an image acquired by the vision system is presented in Fig. 4. As we can see, the objects of interest (balls, lines and obstacles) are correctly detected even when they are far from the robot. Moreover, the balls can correctly be detected up to 9 m (notice that the robot is in the middle line of the field and the further ball is over the goal line) even when they are partially occluded or engaged by another robot. No false positives in the detection are observed.

Several game scenarios have been tested using the CAMBADA autonomous mobile robots. In Fig. 5(a) we present a graphic with the result of the ball



a) b)

Fig. 4. On the left, an image acquired by the omnidirectional vision system. On the right, the result of the color-coded object detection. The blue circles mark the white lines, the white circles mark the black obstacles and the mangenta circles mark the orange blobs that passed the validation thresholds (Color figure online).



a) b) c)

Fig. 5. On the left, a graph showing the ball detection when the robot is moving in a tour around the soccer field. In the middle, ball detection results when the robot is stopped on the middle line on the right of the ball and the ball is sent across the field. On the right, ball detection results when both the robot and the ball are moving.

detection when the ball is stopped in a given position (the central point of the field, in this case) while the robot is moving. The graphic shows a consistent ball detection while the robot is moving in a tour around the field. The field lines are also properly detected, as it is proved by the correct localization of the robot in all the experiments. The second scenario that has been tested is illustrated in Fig. 5(b). The robot is stopped on the middle line and the ball is sent across the field. This graph shows that the ball detection is accurate even when the ball is found at a distance of 9 m away from the robot. Finally, in Fig. 5(c) both the robot and the ball are moving. The robot is making a tour around the soccer field, while the ball is being sent across the field. In all these experiments, no false positives were observed and the ball has been detected in more than

90 % of the frames. Most of the times the ball was not detected was due to the fact that it was hidden by the bars that hold the mirror of the omnidirectional vision system. The video sequences used for generating these results, as well as the configuration file that has been used, are available at [14]. In all the tested scenarios the ball is moving on the ground floor since the single camera system has no capability to track the ball in 3D.

The processing time shown in Table 1 proves that the vision system built using the UAVision library is extremely fast. The full execution of the vision pipeline software only takes on average a total of 12 ms, allowing thus a framerate greater than 80 fps. Moreover, the maximum processing time that we measured was 13 ms, which is a very important detail since it shows that the processing time is almost independent of the scene complexity. The time results have been obtained in a computer with a Intel Core i5-3340M CPU @ 2.70 GHz 4 processor, processing images with a resolution of 1024×1024 pixels (a Region Of Insterest centered in the CMOS of the camera used). In the implementation of this vision system we didn't use multi-threading. However, both image classification and the next steps can be parallelized if needed.

Table 1. Average processing times measured using the video sequences that we provide along with this paper.

Operation	Time (ms)
Acquisition	1
RLE	4
Blob creation	2
Blob validation	3
Total	12

The LUT is created once, when the vision process runs for the first time and it is saved in the cache file. If the information from the configuration file does not change during the following runs of the vision software, the LUT will be loaded from the cache file, reducing thus the processing time of this operation by approximately 25 times.

For the video sequences that we provide, the following number of scanlines have been built during the performance of the vision software:

- 720 radial scanlines for the ball detection.
- 98 circular scanlines for the ball detection.
- 170 radial scanlines for the lines and obstacle detection.
- 66 circular scanlines for the lines detection.

The cameras that have been used can provide 50 fps at full resolution (1280×1024 pixels) in RGB color space. However, some cameras available on the market can only provide 50 fps accessing directly to the CCD or CMOS data, usually a

single channel image using the well known Bayer format. As described before, the LUT in the vision library can work with several color spaces, namely RGB, YUV and Bayer format. We repeated the three scenarios described above acquiring images directly in the Bayer format also at 50 fps and the experimental results show that the detection performance is not affected as expected, since the conversion between Bayer and RGB does not generate new information regarding the perception.

In addition to the good performance in the detection of objects, both in terms of number of times that an object is visible and detected and in terms of error in its position, the vision system must also perform well in minimizing the delay between the perception of the environment and the reaction of the robot. It is obvious that this delay depends on several factors, namely the type of the sensor used, the processing unit, the communication channels and the actuators, among others. To measure this delay in the CAMBADA robots, a setup was developed which is presented in Fig. 6. The setup consists of a led that is turned on by the motor controller board and the same board measures the time that the whole system takes to acquire and detect the LED flash, and send the respective reaction information back to the controller board. The vision system detects the led on and when it happens, the robotic agent sends a specific value of velocities to the hardware (via HWComm application). This is the normal working mode of the robots in game play.

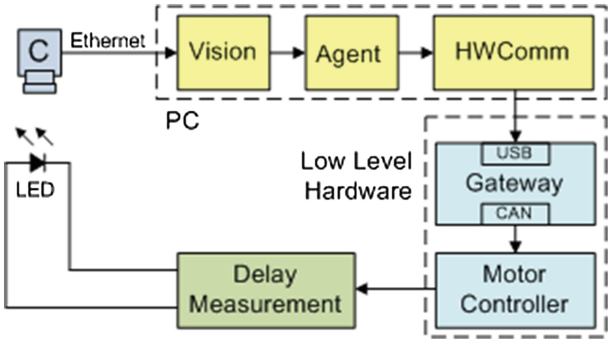


Fig. 6. The blocks used in our measurement setup. These blocks are used by the robots during game play.

As presented in Fig. 7, the delay time between perception and the reaction of the robot significantly decreases when working at higher frame rates. The average delay at 30 fps is 65 ms and at 50fps it is 53 ms, which corresponds to an improvement of 22 %. The jitter verified reflects the normal function of the several modules involved, mainly because there is no synchronism between the camera and the processes running on the computer.

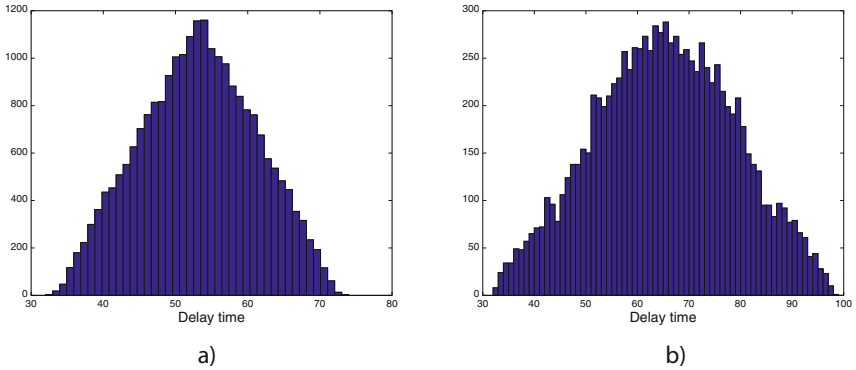


Fig. 7. Histograms showing the delay between perception and action on the CAMBADA robots. On the left, the camera is working at 50 fps (average = 53 ms, max = 74 ms, min = 32 ms). On the right, the camera working at 30 fps (average = 65 ms, max = 99 ms, min = 32 ms).

4 Conclusions and Future Work

In this paper we have presented a novel time-constrained computer vision library that has been successfully employed in the games of robotic soccer. The proposed library, UAVision, encompasses algorithms for camera calibration, image acquisition and color coded object detection and allows frame rates of up to 50 fps.

In what concerns the future work, the next step will be to use the developed library in other RoboCup Soccer Leagues and the first concern is adding support for the cameras used by the robots in the Standard Platform and Humanoid Leagues and employing the same vision system on them. Moreover, we aim at providing software support for image acquisition from several other types of cameras and complement the library with algorithms for generic object detection, relaxing thus the rules of color coded objects and supporting the evolution of the RoboCup Soccer Leagues.

Acknowledgements. This work was developed in the Institute of Electronic and Telematic Engineering of University of Aveiro and was partially supported by FEDER through the Operational Program Competitiveness Factors - COMPETE and by National Funds through FCT - Foundation for Science and Technology in a context of a PhD Grant (FCT reference SFRH/BD/85855/2012) and the project FCOMP-01-0124-FEDER-022682 (FCT reference PEst-C/EEI/UI0127/2011).

References

1. Neves, A.J.R., Pinho, A.J., Martins, D.A., Cunha, B.: An efficient omnidirectional vision system for soccer robots: from calibration to object detection. Mechatronics **21**(2), 399–410 (2011)

2. Kanters, F.M.W., Hoogendijk, R., Janssen, R.J.M., Meessen, K.J., Best, J.J.T.H., Bruijnen, D.J.H., Naus, G.J.L., Aangenent, W.H.T.M ., van der Berg, R.B.M., van de Loo, H.C.T., Heldes, G.M., Vugts, R.P.A., Harkema, G.A., van Brakel, P.E.J., Bukkums, B.H.M., Soetens, R.P.T., Merry, R.J.E., can de Molengraft, M.J.G.: Tech united eindhoven team description. In: RoboCup 2011, Istanbul, Turkey (2011)
3. Kappeler, U.P., Zweigle, O., Rajaie, H., Hausserman, K., Tamke, A., Koch, A., Eck-stein, B., Aichele, F., DiMarco, D., Berthelot, A., Walter, T., Levi, P.: RFC stuttgart team description. In: RoboCup 2011, Istanbul, Turkey (2011)
4. Huang, M., Ge, X., Hui, S., Wang, X., Chen, S., Xu, X., Zhang, W., Lu, Y., Liu, X., Zhao, L., Wang, M., Zhu, Z., Wang, C., Huang, B., Ma, L., Qin, B., Zhou, F., Wang, C.: Water team description. In: RoboCup 2011, Istanbul, Turkey (2011)
5. Lu, H., Zeng, Z., Dong, X., Xiong, D., Tang, S.: Nubot team description. In: RoboCup 2011, Istanbul, Turkey (2011)
6. Nassiraei, A.A.F, Ishida, S., Shinpuku, N., Hayashi, M., Hirao, N., Fujimoto, K., Fukuda, K., Takanaka, K., Godler, I., Ishii, K., Miyamoto, H.: Hibikino-musashi team description. In: RoboCup 2011, Istanbul, Turkey (2011)
7. Trifan, A., Neves, A.J.R., Cunha, B., Lau, N.: A modular real-time vision system for humanoid robots. In: Proceedings of SPIE IS&T Electronic Imaging 2012, January 2012
8. Neves, A.J.R., Trifan, A., Cunha, B.: Self-calibration of colormetric parameters in vision systems for autonomous soccer robots. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS, vol. 8371, pp. 183–194. Springer, Heidelberg (2014)
9. Cunha, B., Azevedo, J., Lau, N., Almeida, L.: Obtaining the inverse distance map from a non-svp hyperbolic catadioptric robotic vision system. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 417–424. Springer, Heidelberg (2008)
10. Neves, A.J.R., Corrente, G.A., Pinho, A.J.: An omnidirectional vision system for soccer robots. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007. LNCS (LNAI), vol. 4874, pp. 499–507. Springer, Heidelberg (2007)
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co. Inc., Boston (1995)
12. Zhang, Z.: Flexible camera calibration by viewing a plane from unknown orientations. In: ICCV, pp. 666–673 (1999)
13. Neves, A.J.R., Azevedo, J.L., Cunha, B., Lau, N., Silva, J., Santos, F., Corrente, G., Martins, D.A., Figueiredo, N., Pereira, A., Almeida, L., Lopes, L.S., Pinho, A.J., Rodrigues, J., Pedreiras, P.: CAMBADA soccer team: from robot architecture to multiagent coordination. In: Papic, V. (ed.) Robot Soccer. ch. 2. I-Tech Education and Publishing, Vienna (2010)
14. http://sweet.ua.pt/an/uavision/. (Last visited March 2014)