UAVision: A Modular Time-Constrained Vision Library for Color-Coded Object Detection

António J.R. Neves, Alina Trifan, and Bernardo Cunha

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

Abstract. The ultimate goal of Computer Vision has been, for more than half of century, to create an artificial vision system that could imitate the human vision. The artificial vision system should have all the capabilities of the human vision system but must not carry the same flaws. Robotics and Automation are just two examples of research areas that use artificial vision systems as the main sensorial element. In these areas, the use of color-coded objects is very common since it relieves the burden of information processing while being an unobtrusive restraint of the environment. We present a novel computer vision library called UAVision that provides support for different video sensors technologies and all the necessary software for implementing an artificial vision system for the detection of color-coded objects. The experimental results that we present, both for the scenario of robotic soccer games and for traffic sign detection, show that our library can work at more than 50fps with images of 1 megapixel.

1 Introduction

Color segmentation has been one of the most popular approaches in time-constrained machine vision processing. This is due to the fact that segmenting a region based on colors is still easier and less heavy from the point of view of the computational resources involved than the detection of objects based on geometric features. In areas such as Robotics and Automation, the environment is often reduced to a set of objects of interest that are color-coded or color-labeled. The precise detection of these objects almost always stands at the basis of the correct functioning of the system. However, there is little work done in a structural manner in what concerns color-coded object detection.

In this paper we present a library for color-coded object detection, named UAVision. We call the design of the library as being modular as the library 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", meaning that it offers support for different vision sensors technologies. The software created using the library is easily exportable and can be shared between different types of vision sensors. Another important aspect of our library is that it takes into consideration time constraints. All the algorithms behind this

Y. Zhang and J.M.R.S. Tavares (Eds.): CompIMAGE 2014, LNCS 8641, pp. 350–361, 2014.

library have been implemented focusing on maintaining the processing time as low as possible. In Robotics, the notion of "realtime" is an extremely important issue and even though there is not a strict definition of realtime, almost always it refers to the amount of time elapsed between the acquisition of two consecutive frames. Realtime processing means processing the information captured by the vision sensors within the limits of the frame rate. The vision system for color-coded object detection that we have implemented using the UAVision library can work with frame rates up to 50fps and the processing time obtained will be presented in the Experimental Results Section.

The research area of robotic vision is greatly evolving by means of international competitions such as the RoboCup Federation [1]. The RoboCup initiative, through competitions like RoboCup Robot Soccer, RoboCup Rescue, RoboCup@Home and RoboCupJunior is designed to meet the need of handling real world complexities, while maintaining an affordable problem size and research cost. The games of robotic soccer have been used as the test environment for the modular vision library that is being presented. Within the RoboCup soccer competitions, autonomous mobile robots have to play soccer obeying to the FIFA rules. In the RoboCup soccer games, the objects of interest are color-coded, which makes these competitions suitable for employing the proposed library.

The library that we are proposing is an important contribution for the Computer Vision community since so far, there are no machine vision libraries that take into consideration time constraints. CMVision [2], a machine vision library developed at the Carnegie Mellon University was one of the first approaches of building such a library but it remained quite incomplete and has been discontinued in 2004. Several other machine vision libraries, such as Adaptive Vision [3], CCV [4] or RoboRealm [5] provide machine vision software to be used in industrial and robotic application but they are not free. UAVision aims at being an open-source, free library that can be used for robotic vision applications that have to deal with time constraints.

This paper is structured into four sections, first of them being this introduction. Section 2 describes the modules of the vision library. Section 3 presents two practical scenario in which the library has been used and the results that have been achieved. Section 4 concludes the paper and future lines of research are highlighted. Finally, in Section 4 the institutions supporting this work are acknowledged.

2 Library Description

The library that we propose 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. The library can be divided into three main modules, that can be combined for implementing a time-constrained vision system or that can be used individually. These modules will be presented in the following subsections.

2.1 Image Acquisition

UAVision provides the necessary software for accessing and capturing images from three different camera interfaces: USB cameras, Firewire cameras and Ethernet cameras. For this purpose, the Factory Design Pattern [6] 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 [7]: the *Mat* structure as a container of the frames that are grabbed and the *Point* structure for the manipulation of points in 2D coordinates.

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

2.2 Camera Calibration

The correct calibration is very important in every vision system of all the parameters related to the 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 in some vision systems, namely in catadioptric vision systems, the detection of the mirror, robot center and the definition of the regions of the image that 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 camera settings contain the basic information about the resolution of the image acquired, the Region of Interest regarding the CCD or CMOS of the camera and colormetric parameters among others.

The mask is a binary image containing the information about the pixels to be processed by the vision system. For example, in a catadioptric vision system it is not worthy to process the robot's body. Ignoring the pixels that belong to the body of the robot is significant for reducing both the noise in the image, as well as the processing time.

The map is a matrix containing the correspondence between a pixel position in the image and the corresponding position in the real world, considering a specific work plane, for example the ground. This can be used in a specific application as a Look Up Table to translate the position of the objects from pixels to real coordinates.

The color ranges contain 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 practice, 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 [9] for calibration of the intrinsic and extrinsic parameters of catadioptric vision systems used to generate the 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 [10].

2.3 Color-Coded Object Detection

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

- **Look-Up Table**

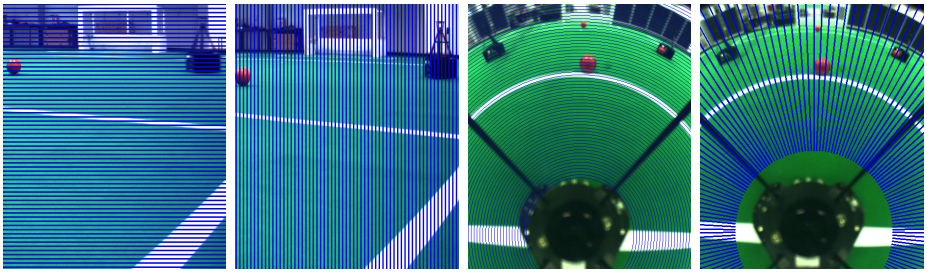
For fast color classification, color classes are defined with the use of a look-up table (LUT). A LUT represents a data structure, in this case an array, used for replacing a runtime computation with a basic array indexing operation.

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, 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 in the valid pixels if a mask exists.

- **Scanlines**

To extract color information from the image we have created three types of search lines, which we also call scanlines: radial, linear (horizontal or vertical) and circular. The radial search lines are constructed based on the Bresenham line algorithm [11]. 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. In more detail, we iterate through its pixels and we 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 filter what can be 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 and also the amount of pixels in each occurrence (Fig. 2).

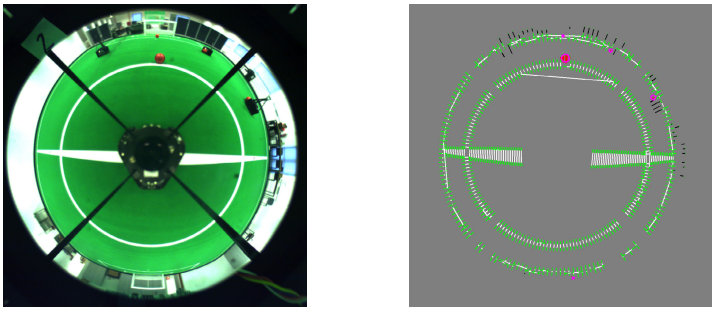


Fig. 2. On the left, an image captured using the Camera Acquisition module of the UAVision library. On the right, the run length information annotated.

- **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 characteristics, 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 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.

3 Experimental Results

We have chosen as a first testbed of the UAVision library the game of robotic soccer, promoted by the RoboCup initiative. We have used the Middle-Size League [12] team of robots CAMBADA [13] from the University of Aveiro (Fig. 3(a)). 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 [14]. In order to play, a robot has to detect, in useful time, the ball, the limits of the field and the field lines, the goal posts and the other robots that are on the field. These robots can move with a speed of up to 4m/s and the ball can be kicked with a velocity of up to 10m/s, which leads to the need of having fast object detection algorithms. In the RoboCup MSL soccer games, the objects of interest are color coded making thus the soccer games a suitable application for the use of the UAVision library. The camera used by the CAMBADA robots is an *IDS UI-5240CP-C-HQ-50i Color CMOS 1/1.8"* Gigabit Ehernet camera [15].

We provide along with this paper three video sequences exemplifying the scenarios that have been tested and a configuration file [17], in order that other researchers are able to reproduce our results, as well as, to serve as a reference for future work.

Using the modules of the UAVision library, a vision system composed of two different applications has been developed with the purpose of detecting the orange ball, the white lines of the field and the black robots or obstacles that are present during the soccer games on the field. The two applications are of the type client-server and they are described next.

The core application, that runs in realtime on the processing units of the robots, was implemented as a server that accepts the connection of a calibration tool client. This client is used for configuring the vision system (color ranges,

a) b) c)

Fig. 3. On the left and center images, an example of a robot setup during a soccer game. On the right, the world as the robot understands it displayed on the team basestation.

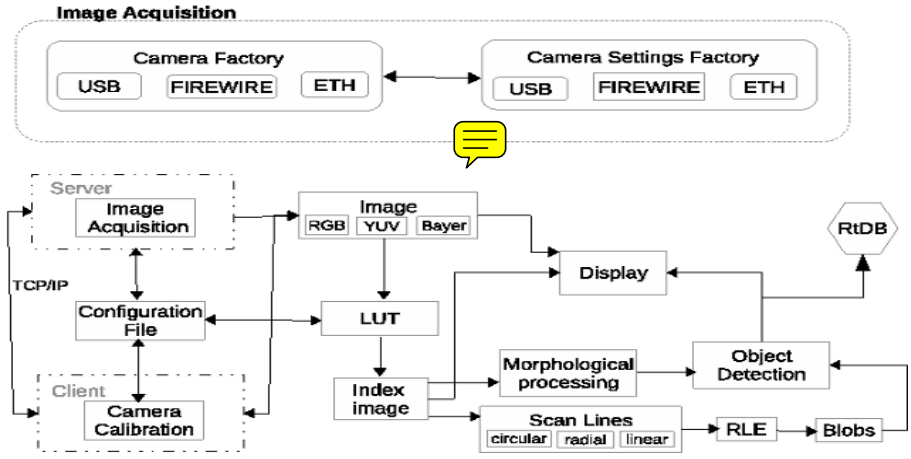


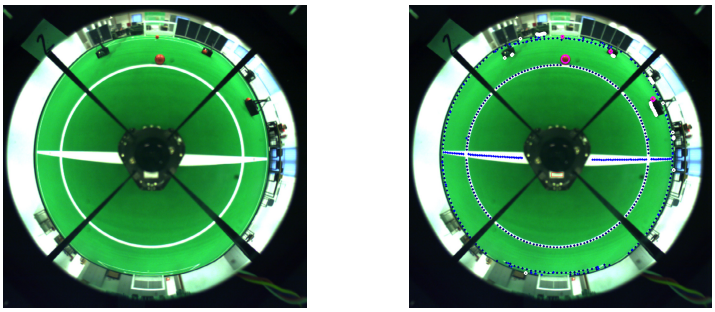
Fig. 4. Software architecture of the vision system developed based on the UAVision library

camera parameters, etc.) and for debug purposes. The main task of the server application is to perform realtime color-coded object detection. The pipeline of the object detection procedure, presented in Fig. 4, 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 garantee 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. 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) [18]. For the obstacles and line detections, the coordinates of the detected points of interest are passed to higher-level processes and shared on the RtDB.

In Fig. 3 we present an example of the soccer games environment. Fig. 3(a) and Fig. 3(b) show a common setup of the robots during a soccer game. The information about the position of the lines is used for the localization of the robot and in Fig. 3(c) we present a screenshot of the team basestation, where it is shown the perception of the robot.

Figure 5(a) shows an image captured by the omnidirectional vision system of the CAMBADA robots and in Fig. 5(b) we have the results of the color detection algorithms, annotated on the image. The blue circles mark the white lines, the white circles mark the black obstacles and the mangenta circles mark the ball color blobs that passed the validation thresholds.

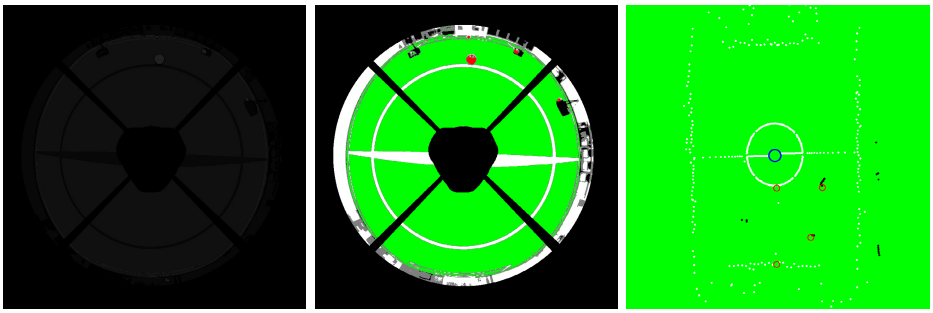


a) b)

Fig. 5. 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.

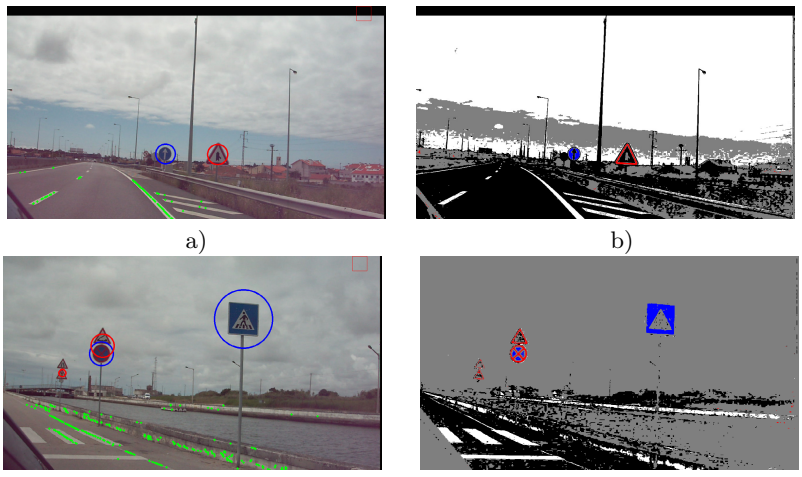
The intermediate steps of the vision pipeline are presented in Fig. 6. In Fig. 6(a) we present the index image, in which all the colors of interest have been assigned a label. Fig. 6(b) presents the color classified image and in Fig. 6(c) we present the "reality" of the robot.

To prove the use of the proposed library in other applications, we installed a digital camera as perspective vision system in a regular vehicle and we made some experiments. Similar results to the ones obtained in soccer robots are presented in Fig. 7. In this case we detect the position of the traffic signs and lines on the road. This could be seen as pre-processing step of a vision system for an autonomous vehicle. The camera used was a Firewire *Point Grey Flea 2, 1 FL2-08S2C with a 1/3" CCD Sony ICX204* [16]. We present results achieved by using the UAVision library for two different cameras, but the architecture of the vision system is the same and it is presented in Fig. 4.



a) b) c)

Fig. 6. On the left, the index image in which all of the colors of interest are labeled. In the middle, the color classified image (a colored version of a)) and on the right, the surrounding world from the perspective of the robot.



a) b)

c) d)

Fig. 7. Results obtained using a perspective camera on a vehicle. (a) and (b) where obtained on a highway and (c) and (d) on a national road. In (a) and (c) - the acquired image where the position of traffic signs and lines on the road are marked; (b) and (d) - color classified image.

Several game scenarios have been tested using the autonomous mobile robots CAMBADA. In Fig. 8(a) we present a graphic with the result of the ball detection when the ball is stopped in a given position, in this case, the central point of the field and 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. 8(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 9m away from the robot. Finally, in Fig. 8(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 detected and the ball has been detected in more than 90% of the frames. Most of the times in which the ball was not detected, it was due to the position of the ball behind the bars holding the mirrorof 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 [17].

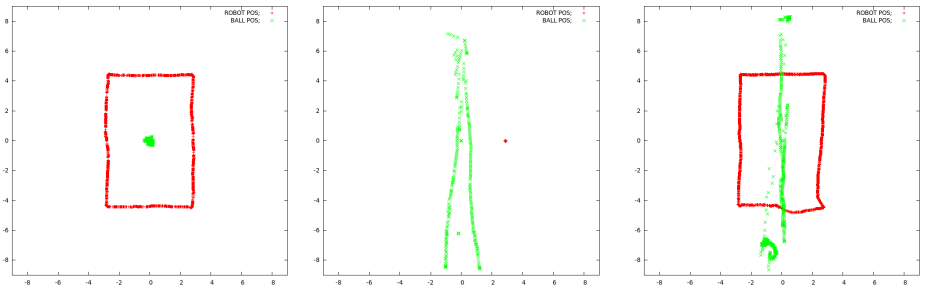


Fig. 8. On the left, a graph showing the ball detection when the robot is moving in a tour around the soccer field. In the middle, ball detection results when the robot is stopped on the middle line on the right of the ball and the ball is sent across the field. On the right, ball detection results when both the robot and the ball are moving.

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 80fps. 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 @ 2.70GHz 4 processor. 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.

The cameras that we have used can provide 50fps at full resolution when acquiring images in the Bayer format. As described before, the LUT in the vision library can work with the Bayer format and the experimental results show that the detection performance is not affected.

There are two initialization processes that are part of the vision pipeline and that take place only once during the execution of the vision system. The execution times of these processes are presented in Table 2.

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 has not

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

Operation	Time (ms)
Image acquisition	1
Run length coding of scanlines	4
Blob creation	2
Bob validation and RtDB filling	3
Total	**10**

Table 2. Processing times for the initialization of the LUT and scanlines

Operation	Time (ms)
LUT creation	5864
LUT loading from a cache file	25
Scanlines initialization	28

been altered, 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. Only when the information in the configuration file changes, the LUT has to be recreated.

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.

4 Conclusions and Future Work

In this paper we have presented a novel computer vision library that supports the development of artificial vision systems for the detection of color coded objects. The proposed library, UAVision, encompasses algorithms for camera calibration, image acquisition and color coded object detection. These algorithms take into consideration time constraints, making the library suitable for applications that have to run in a specific, limited, interval of time. As future work, we aim at providing software support for image acquisition from more types of cameras and complement the library with algorithms for generic object detection.

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. http://www.robocup.org (last visited March, 2014)
2. http://www.cs.cmu.edu/~jbruce/cmvision/ (last visited March, 2014)
3. https://www.adaptive-vision.com/en/home/ (last visited March, 2014)
4. http://libccv.org/ (last visited March, 2014)
5. http://www.roborealm.com/index.php (last visited March, 2014)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston (1995)
7. http://docs.opencv.org (last visited March, 2014)
8. Alina Trifan, A.J.R.: Neves and Bernardo Cunha. Self-calibration of colormetric parameters in vision systems for autonomous soccer robots. In: Proc. of RoboCup 2014 Symposium (2014)
9. 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)
10. Zhang, Z.: Flexible camera calibration by viewing a plane from unknown orientations. In: ICCV, pp. 666–673 (1999)
11. Jack, E.: Bresenham. Algorithm for computer control of a digital plotter. IBM Systems Journal 4(1), 25–30 (1965)
12. http://wiki.robocup.org/wiki/Middle_Size_League (last visited March, 2014)
13. http://robotica.ua.pt/CAMBADA (last visited March 2014)
14. Neves, A.J.R., Corrente, G., 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)
15. https://en.ids-imaging.com (last visited March, 2014)
16. http://www.ptgrey.com/products/flea2/flea2_firewire_camera.asp (last visited March, 2014)
17. http://sweet.ua.pt/an/uavision/ (last visited March, 2014)
18. Neves, A., et al.: CAMBADA soccer team: from robot architecture to multiagent coordination. In: Papic, V. (ed.) Robot Soccer, ch. 2. I-Tech Education and Publishing, Vienna (2010)