ACSIJ Advances in Computer Science: an International Journal, Vol. 5, Issue 1, No.19 , January 2016
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Real-Time Color Coded Object Detection Using a Modular Computer Vision Library

Alina Trifan[1], António J. R. Neves[2], Bernardo Cunha[3] and José Luís Azevedo[4]

[1] IEETA/ DETI, University of Aveiro
Aveiro, Portugal
alina.trifan@ua.pt

[2] IEETA/ DETI, University of Aveiro
Aveiro, Portugal
an@ua.pt

[3] IEETA/ DETI, University of Aveiro
Aveiro, Portugal
mbc@det.ua.pt

[4] IEETA/ DETI, University of Aveiro
Aveiro, Portugal
jla@ua.pt

Abstract

In this paper we present a novel computer vision library called UAVision that provides support for different digital cameras technologies, from image acquisition to camera calibration, and all the necessary software for implementing an artificial vision system for the detection of color-coded objects. The algorithms behind the object detection focus on maintaining a low processing time, thus the library is suited for real-world real-time applications. The library also contains a TCP Communications Module, with broad interest in robotic applications where the robots are performing remotely from a basestation or from an user and there is the need to access the images acquired by the robot, both for processing or debug purposes. Practical results from the implementation of the same software pipeline using different cameras as part of different types of vision systems are presented. The vision system software pipeline that we present is designed to cope with application dependent time constraints. The experimental results show that using the UAVision library it is possible to use digital cameras at frame rates up to 50 frames per second when working with images of size up to 1 megapixel. Moreover, we present experimental results to show the effect of the frame rate in the delay between the perception of the world and the action of an autonomous robot, as well as the use of raw data from the camera sensor and the implications of this in terms of the referred delay.

Keywords: *Image Processing; object detection; real-time processing; color processing*

1. Introduction

Even though digital cameras are quite inexpensive nowadays, thus making artificial vision an affordable and popular sensor in many applications, the research done in this field has still many challenges to overcome. The main challenge when developing an artificial vision system is processing all the information acquired by the sensors within the limit of the frame rate and deciding in the smallest possible amount of time which of this information is relevant for the completion of a given task.

In many applications in the areas of Robotics and Automation the environment is still controlled up to a certain extent in order to allow progressive advancements in the different development directions that stand behind these applications. There are many industrial applications in which semi or fully autonomous robots perform repetitive tasks in controlled environments, where the meaningful universe for such a robot is reduced, for example, to a limited set of objects and locations known apriori. In applications such as industrial inspection, traffic sign detection or robotic soccer, among others, the environment is either reduced to a set of objects of interest that are color-coded (or color-labeled) or the color segmentation of the objects of interest is the first step of the object detection procedure. This is mainly due to the fact that segmenting a region based on colors is less heavy from the point of view of the computational resources involved than the detection of objects based on generic features. In what concerns color object detection there is little work done in a structural manner. There can be found in literature published work on color segmentation for object detection and common aspects in the research papers that approached this problem can be traced. However, to our knowledge, there is no free, open source available library that allows the complete implementation of a vision system software for color coded object detection.

In this paper we present a library for color-coded object detection, named UAVision. The library aims at being a complete collection of software for real-time color object detection. UAVision can be split into several independent modules that will be presented in the following sections. The architecture of our software is of the type "plug and play", meaning that it offers support for different digital cameras technologies and the software created using the library is easily exportable and can be shared between different types of cameras. We call the library modular as each module can be used independently as a link in an image processing chain or several modules can be used for creating a complete pipeline of an artificial vision system.

Another important aspect of the UAVision 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 and allowing the use of digital cameras at the maximum frame rates that their hardware supports. In Autonomous Mobile Robotics, performing in "real-time" is a demand of almost all applications since the main purpose of robots is to imitate humans and we, humans, have the capacity of analyzing the surrounding world in "real-time". Even though there is not a strict definition of real-time, almost always it refers to the amount of time elapsed between the acquisition of two consecutive frames. Real-time processing means processing the information captured by the digital cameras within the limits of the frame rate. There are many applications in which the events occur at a very high pace, such as, for example, industrial production or inspection lines where robots have to repeatedly and continuously perform the same task. If that task involves visual processing, the vision system of such a robot has to keep up with the speed of the movements so that there are the smallest delay as possible between perception and action.

In this paper we present a vision system for color-coded object detection, which has been implemented using the UAVision library and three different types of digital cameras. Three different cameras have been used in order to prove the versatility of the proposed library and to exemplify the range of options that a user has for implementing a vision system using the UAVision library. The vision systems that we present are of two different types, perspective and omnidirectional, and can perform colored object detection in real-time, working at frame rates up to 50fps.

We present the same vision system pipeline implemented with different types of digital cameras, which fundamentally differ one from another. Even though the three vision systems that we present are designed to detect colored objects in the same environment and they share the same software structure, their architectures both in terms of software implementation and hardware are very different.

The first vision system that we present is a catadioptric one and uses an Ethernet camera. Following the same software pipeline, we present also two perspective vision systems implemented using a Firewire camera and a Kinect sensor, respectively. These two perspective vision systems are used with different purposes. Not only the access to these cameras is done differently in each of these cases, but the calibration module of the library provides methods for calibrating them both in terms of camera parameters and in terms of pixel-world coordinates. The calibration for each of these cameras is done by employing different types of algorithms, that will be explained in the next sections.

The UAVision library has been built using some of the features of the OpenCV library, mostly data structures for image manipulation. Although a powerful tool for developers working in image processing, the OpenCV library does not provide all the necessary support for implementing a complete vision system from scratch. To name some of the contributions of our library, it provides support for accessing different types of cameras using the same interface, algorithms for camera calibration (colormetric, catadioptric and perspective calibration) and the possibility of using image masks to process only relevant parts of the image. Moreover, we introduced blobs as data structures, an important aspect in colored object detection.

The authors have not found in literature free, up to date, available computer vision libraries for time constrained applications. We consider this paper an important contribution for the Computer Vision community since it presents in detail a novel computer vision library whose design focuses on maintaining a low processing time, a common constraint of nowadays autonomous systems.

The practical scenario in which the use of the library has been tested were the robotic soccer games, promoted by the RoboCup Federation [1]. The RoboCup Federation is an international organization that encourages research in the area of robotics and artificial intelligence by means of worldwide robotic competitions organized yearly. The RoboCup competitions are organized in four different challenges, one of them being the Soccer League. In this challenge, completely autonomous robots compete in games of soccer, following adjusted FIFA rules. The soccer challenge is divided into four leagues, according to the size and shape of the competing robots: the Middle Size League (MSL), the Standard Platform League (SPL), the Humanoid League (HL) and the Small Size League (SSL) (Fig. 1).

In MSL the games are played between teams of up to five wheeled robots that each participating team is free to build keeping in mind only the maximum dimensions and

weight of the robots. In SPL, all teams compete using the same robotic platform, a humanoid NAO built by Aldebaran [2] and the games take place between teams of up to five NAOs. In HL, the games are played between humanoid robots, that is robots that imitate the human body and the number of players in a game depends on the sub-league in which a team is participating: Kid-Size League, Teen League or Adult League. All the RoboCup games occur in closed, controlled environments. The robots play soccer indoor, on a green with white lines carpet and the color of the ball is known in advance. The color labeling of the environment makes the RoboCup competitions suitable for employing the UAVision library for the vision system of the autonomous robotic soccer players. Based on the library, an omnidirectional vision system has been implemented for the MSL team of robots CAMBADA [3], [4] from University of Aveiro. A perspective vision system that can be used either as a complement of the omnidirectional one or as an external monitoring tool for the soccer games has also been implemented and results obtained with both types of these systems are presented in this paper.

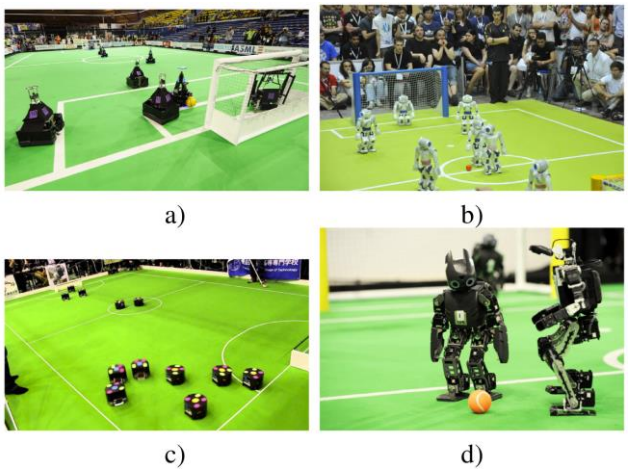


Fig. 1 Game situations in the four soccer leagues of RoboCup: a) MSL, b) SPL, c) SSL and d) HL.

This paper is structured into six sections, first of them being this introduction. Section 2 presents an overview of the most recent work on color coded object detection. Section 3 describes the modules of the vision library, while Section 4 presents the typical pipeline of a vision system developed based on the UAVision modular library. In Section 5 we present the practical scenarios in which the library has been used and the results that have been achieved, as well as the processing time of our algorithms and the proof that the library is indeed suitable for time-constrained applications. Finally, Section 6 concludes the paper.

2. Related Work

The library that we are proposing is an important contribution for the Computer Vision community since so far, there are no free, open source machine vision libraries that take into consideration time constraints. CMVision [5], a machine vision library developed at the Carnegie Mellon University was one of the first approaches to build such a library but it remained quite incomplete and has been discontinued in 2004. Several other machine vision libraries, such as Adaptive Vision [6], CCV [7] or RoboRealm [8], provide machine vision software to be used in industrial and robotic applications but they are not free. UAVision aims at being a free, open-source library that can be used for robotic vision applications that have to deal with time constraints.

In most autonomous robots there is the concern of maintaining an affordable cost of construction and limited power consumption. Therefore, there are many limitations in terms of hardware that have to be overcome by means of software in order to have a functional autonomous robot.

This is the case of most of the robots used by the broad RoboCup community. The most recent innovations done by the RoboCup community are presented next as they are related to the work proposed in this paper. The UAVision library has been used for the implementation of a vision system for a soccer robot and therefore, it is important to highlight the challenges that uprise in this environment and the state of the art available solutions.

The soccer game in the RoboCup Soccer League is a standard real-world test for autonomous multi-robot systems. In the MSL, considered to be the most advanced league in RoboCup at the moment, omnidirectional vision systems have become widely used, allowing a robot to see in all directions at the same time without moving itself or its camera [9]. 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 for the object detection, relaxing thus the detection algorithms. Many teams are currently taking their first steps in 3D ball information retrieving [10], [11], [12]. There are also some teams moving their vision systems algorithms to VHDL based algorithms taking advantage of the FPGAs versatility [10]. Even so, for now, the great majority of the

teams base their image analysis in color search using radial sensors [13], [14], [15].

While in MSL the main focus of the research efforts relies in building robots with fast reactions, good cooperation skills and complex artificial intelligence, able to play soccer as well as humans do from the point of view of game strategy and cooperation, in SPL the focus is achieving a stable biped walk, similar to the human walk. Most teams participating in this league use color segmentation as basis of the object detection algorithms. The first step in most of the color segmentation algorithms is scanning the image either horizontally, vertically, or in both directions [16], [17] while looking for one of the colors of interest. For this process, the image is almost always sampled down by skipping lots of pixels and thus losing information and considering what might be false positives. Another approach is using the methods provided by the CMVision library [5] which provides a base to segment images into regions of interest, to which probabilities are assigned [18]. The information about a color of interest segmented is validated in most cases if there is a given threshold of green color in the proximity of the color of interest previously segmented [16], [17], [19]. This is done in order to guarantee that the objects of interest are only found within the limits of the field. Therefore, when searching for an orange ball, as an example, the simplest validation method is checking if there is as little as one green pixel before and/or after an orange blob that has been found.

Team description papers like [17], [20], [21] validate the colors of interest segmented if they are found under the horizon line. The notion of horizon is calculated based on the pose of the robot's camera with respect to the contact point of the robot with the ground, which is the footprint. The next step in the detection of the objects of interest is the extraction of certain features or calculation of different measures from the blobs of the segmented colors of interest. It is common and useful to compute the bounding box and the mass center of a blob of a given color [16], [22], [23], as well as to maintain a function of the size of the object relative to the robot [23].

3. UAVision: A Real-Time Computer Vision Library

The library that we are presenting is intended, as stated before, for the development of artificial vision systems to be used for the detection of color-coded objects. The library incorporates software for image acquisition from digital cameras supporting different technologies, for camera calibration, blob formation, which stands at the basis of the object detection, and image transfer using TCP communications. The library can be divided into four 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.

3.1 Acquisition Module

UAVision provides the necessary software for accessing and capturing images from three different camera interfaces, so far: USB, Firewire and Ethernet cameras. For this purpose, the Factory Design Pattern [24] has been used in the implementation of the Image Acquisition module of the UAVision library. A factory called “Camera” has been implemented and the user of the library can choose from these three different types of cameras in the moment of the instantiation. This software module 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 library also provides software for accessing and capturing both color and depth images from a Kinect sensor [25].

Apart from the image acquisition software, this module also provides methods for converting 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.

3.2 Camera Calibration Module

The correct calibration of all the parameters related to a vision system is crucial in any application. 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 and definition of regions in the image that do not have to be processed. The process of the vision system calibration handles four main blocks of information: camera settings, mask, map and color ranges.

The camera settings block represents the basic camera information. Among others, this includes: the resolution of the acquired image, the Region of Interest regarding the CCD or CMOS of the camera and the colormetric parameters.

The mask is a binary image representing the areas of the image that do not have to be processed, since it is known apriori that they do not contain relevant information. Using a mask for marking non-interest region is a clever approach especially when dealing with a catadioptric vision system. These systems provide a 360º of the surrounding environment and that includes, most of the times, parts of the robot itself. By using a mask that allows skipping the processing of certain regions, the processing time decreases significantly.

The map, as the name suggests, is a matrix that represents the mapping between pixel coordinates and world coordinates. This mapping between the pixels of the acquired image and real coordinates allows the vision system to characterize the objects of interest, their position or size. The algorithms presented in [9], [27] are being used for calibration of the intrinsic and extrinsic parameters of catadioptric vision systems in order 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 [28].

For the omnidirectional vision system, by exploring a back-propagation ray-tracing approach and the physical properties of the mirror surface [27], the relation between the distances in the image and the distances in the real world is known. For the perspective vision systems, after having calculated the intrinsic and extrinsic parameters, the map calculation is straightforward, based on the camera pinhole model equations.

The color ranges block contains the color regions for each color of interest (at most 8 different colors as it will be explained 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 Camera Calibration module of the library allows the storage of all of the information contained in these four blocks in a binary or text configuration file.

UAVision library contains algorithms for the self-calibration of the parameters described above, including some algorithms developed previously within our research group, namely the algorithm described in [26] for the automatic calibration of the colormetric parameters. These algorithms extract some statistical information from the acquired images, such as the intensity histogram, saturation histogram and information from a black and white area of the image, to then estimate the colormetric parameters of the camera.

3.3 Color-coded Object Segmentation Module

The color-coded object segmentation is composed of three 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 RAW data), but the meaning and position of the color components in the image changes. For example, considering a RGB image, the index of the table to be chosen for a specific triplet is obtained as

```
idx = R << 16 | G << 8 | B,
```

being the << a bitwise shift to the left operation and | the bitwise OR. The RAW data (Bayer pattern) is also RGB information, however, the position where the R, G and B information are picked changes. These positions are also calculated only once by the LUT module in order to obtain a faster access during color classification.

Each bit in the table entries expresses if one of the colors of interest (in this case: 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. If a mask exists, color classification is only applied to the valid pixels defined by the mask.

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 minimization of processing time. In Fig. 2 the three different types of scanlines are illustrated.

A Scanline is a list of pixel positions in the image, whose shape depends on its type. The module Scanlines allows the definition of an object that is a vector of scanlines. The amount and characteristics of these scanlines depend on the parameters (start and end positions, distance - vertical, horizontal or angular between scanlines, etc).

Each Scanline object is used by the RLE module, described next, that produces a list of occurrences of a specific color. Depending on the number of Scanlines objects created in a specific vision system, we will have the same number of RLE lists produced. Then, the Blob module is able to use all these lists to combine RLEs and form blobs, as described next.

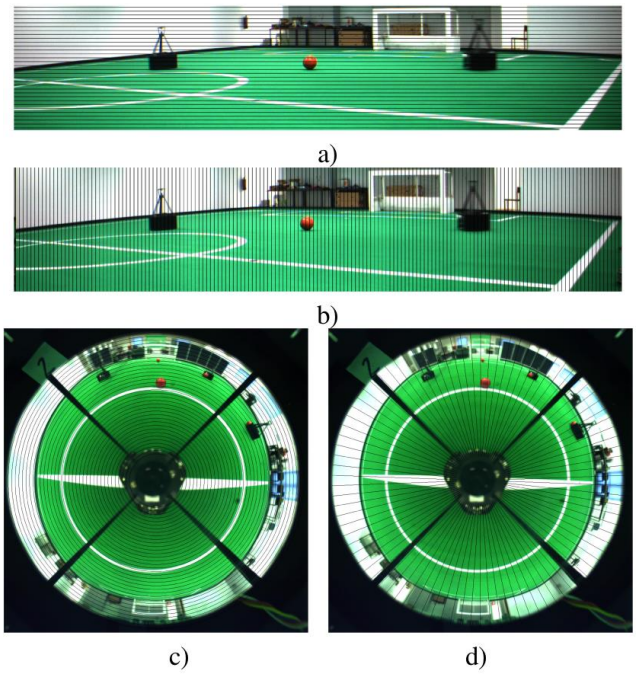


Fig. 2 - Examples of different types of scanlines: a) horizontal scanlines, b) vertical scanlines, c) circular scanlines, d) radial scanlines.

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 (Fig. 3). 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.

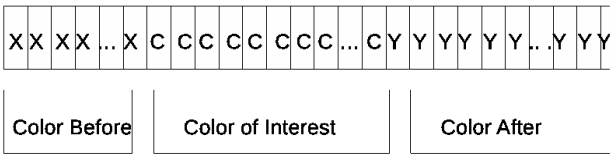


Fig. 3 - RLE illustration.

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. These thresholds are application dependent and should be determined experimentally.

As a result of this module, the user will 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. 4).

A A A C C C C C B B B X X C C C B B...
colorList={(3, 5, 5, 5), (13, 3, 0, 2)...}

Fig. 4 - An example of a result of the RLE module. The first quadruple in the list describes the first detection of the color of interest, the second one the second detection of the color of interest and so on. The values presented in each quadruple represent the following information, by order: position of the first found pixel of the color of interest, number of pixels of the color of interest, number of pixels of the color found before the color of interest and number of pixels of the color found after the color of interest. The number of pixels found before or/and after the color of interest are optional and can be specified when searching for transitions between colors.

Blob formation

To detect objects with a specific color in a scene, one has to detect regions in the image with that color, usually designated as blobs. The blobs can be later on validated as objects of interest after extracting different features of the blobs that are relevant for establishing whether they are more than just blobs of colors. Such features can be, among others, the area of the blob, the bounding box, solidity, skeleton.

In order to construct the blobs, we use information about the position where a specific color occurs based on the Run Length module previously described (a practical example can be seen in Fig. 5). We iterate through all of 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 (Fig. 6).

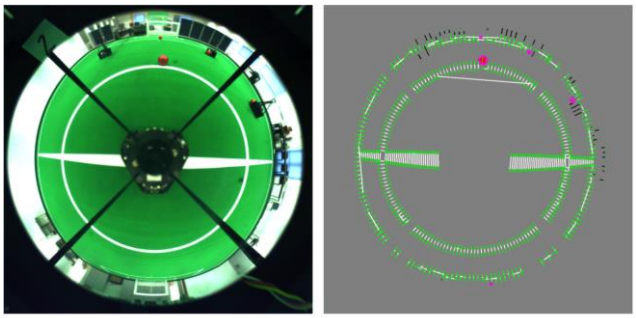


Fig. 5 - On the left, an image captured using the Camera Acquisition module of the UAVision library. On the right, the run length information annotated.

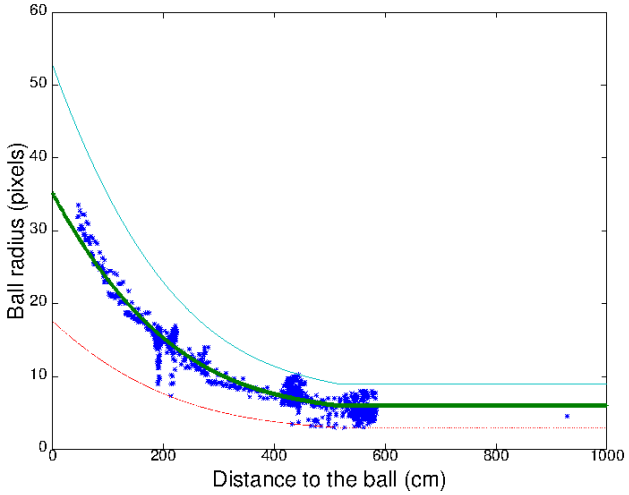


Fig. 6 - An example of a function used for a catadioptric vision system describing the radius (in pixels) of the object of interest (in this case, an orange soccer ball) as a function of the distance (in centimeters) between the robot and the object. The blue marks represent the measures obtained, the green line the fitted function and the cyan and red line the upper and lower bounds considered for validation.

The algorithm for forming blobs that we implemented in the library works as follows: a RLE object contains information about the occurrence of a color of interest on a given scanline. The information contained by the RLE object is: the position of the first found pixel of the color of interest, the number of consecutive pixels of the same color and the position of the last found pixel of the color of interest. The middle point between the first and last found pixels of the color of interest can be calculated for each RLE.

When searching for blobs, the first blob is considered to be the first found RLE and the center of the blob is the center of the RLE. Running through all the found RLE, based on the Euclidean distance between middle points in each run lengths, the color information in adjacent RLE is merged if the calculated Euclidean distance is between a given threshold. This threshold has been experimentally calculated. If the calculated distance is not lower than the threshold, the given RLE is considered to be the starting point of a new blob and the process is repeated.

When joining another RLE to the Blob, the center of the blob, as well as other measurements such as area, width/height relationship, solidity are being calculated. The blobs are then validated as being objects of interest if they respect the size-distance function presented in Fig. 6 and if they are fairly round objects.

TCP Communications Module

Two of the major limitations that have to be considered when working with autonomous robots are the energetic autonomy of the robot and its processing capacities. The broad experience in autonomous robotics within our research group has pointed out the fact that certain calibration tasks, that have to be performed prior to the functioning of a robot, might require too much of the on board resources. Therefore, based on the approach "divide and conquer" we have implemented a TCP Communications module that allows us to remotely communicate with a robot and distribute some of the tasks that are solely related to the calibration processes. Moreover, this module is relevant when developing applications for logging or monitoring, in which the robot performs autonomously and there is the need of having a real-time remote logger/monitor.

The purpose of this module was to simplify and be more intuitive than the POSIX API, while at the same time remain close to the known flow of operations defined by it. The new proposed flow of operations can be seen in Fig. 7. The socket descriptor is no longer the representation of a socket, but rather the object instantiated. Separating the client and server into two distinct sockets, with a single operation to setup both, the "open" operation, the flow of operations is simpler while familiar. On the client side, the socket descriptor request, the socket binding and the connection to the server were all merged into one "open" operation. The rest of the client operations were left unchanged, "send" and "receive" operations to communicate with the server and a "close" operation to close the socket. On the server side, similar to the client, the socket descriptor request, the socket binding and the passive socket marking were merged into one "open" operation, retaining the "accept" operation. The behavior of server and client are now defined in the type of socket instantiated.

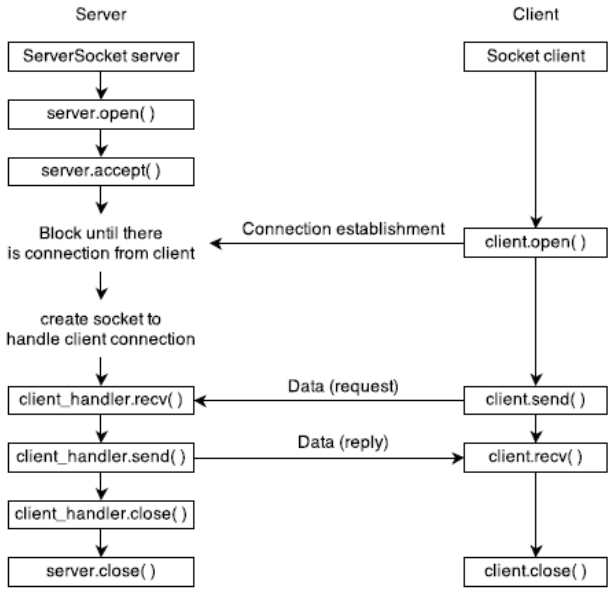


Fig. 7 - Setup of TCP Communications Module sockets for a client-server application.

Performance wise, this library has the same performance as the POSIX API while having the benefit of a simpler API, object oriented and with code legibility.

This module is relevant for the library that we are presenting since it allows the user to have a client and a server that can exchange information via TCP connection. The exchanged information is an image, with or without any annotated information. Based on this module, the vision system that will run on a robot can act as a server and an external application, the client, can request information from the server while it is performing its regular task. By using threads, the communications with the client will not affect the normal flow of operations on the server side. An example of a client application that has been implemented will be presented in Section 5.

4. A Typical Vision System Using the UAVision Library

An example of a typical vision system that can be used when developing a time-constrained robotic vision system is presented in Fig. 8.

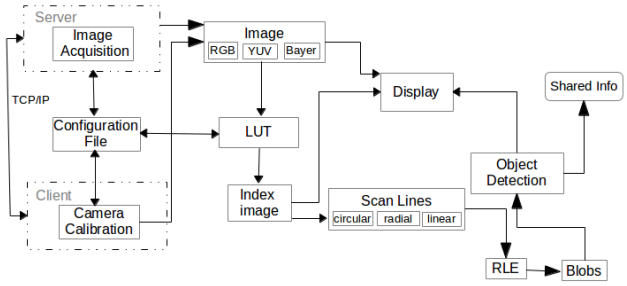


Fig. 8 - Software architecture of the vision system developed based on the UAVision library.

The pipeline of the object detection procedure is the following:

- After having acquired an image, this is transformed into an image of labels using a LUT previously built. This image of color labels, also designated in this paper 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 colors of interest or transitions between colors of interest is run length encoded.
- Blobs are formed by merging adjacent RLEs of a color of interest.
- The blob is then labeled as object of interest if it passes several validation criteria that are application dependent.
- If a given blob passes the validation criteria, its center coordinates will be passed to higher-level processes and they can be shared.

5. Experimental Results

The UAVision library has been used so far in different applications that are being developed at the University of Aveiro, Portugal. In this section we present the scenarios in which time-constrained robotic vision systems that have been built based on UAVision library have been successfully employed.

The code is written in C++ and the main processing unit available on the robots is an Intel Core i5-3340M CPU @ 2.70GHz 4 processor, running Linux (distibution Ubuntu 12.04. LTS Precise Pangolin). In the implementation of the vision system that we present in the paper, we did not use multi-threading, however both image color classification and the next steps can be parallelized if needed.

The parallelization of our software is not done for a basic reason: the Linux kernel installed on our computers is configured to use a 10ms scheduling interval. At this time interval the kernel determines whether the current process continues using the same core, or performs a context switch to another process. Since none of the modules of our vision system has processing time higher than 10ms, using threads would not improve it. In slower systems or in applications that have to deal with much higher resolution images, parallelization can be an option.

5.1 A time constrained catadioptric vision system for robotic soccer players

CAMBADA team of robots from University of Aveiro, Portugal (Fig. 9) is an established team that has been participating, for a decade, in the RoboCup MSL competitions previously described. These robots are completely autonomous, able to perform holonomic motion and are equipped with a catadioptric vision system that allows them to have omnidirectional vision [29]. In order to play soccer, 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 4m/s and the ball can be kicked with a velocity of more than 10m/s, which leads to the need of having fast object detection algorithms. In the RoboCup soccer games, the objects of interest are color coded making thus the soccer games a suitable application for the use of the UAVision library.

Using the modules of the UAVision library, a vision system composed of two different applications has been developed with the purpose of detecting the objects of interest. The two applications are of the type client-server

and have been implemented using the TCP Communication module. 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, camera parameters, etc.) and for debug purposes. The main task of the server application is to perform real time color-coded object detection. The camera used by the CAMBADA robots is an IDS UI-5240CP-C-HQ-50i Color CMOS 1/1.8'' Gigabit Ehernet camera [30], providing images with a resolution of 1280 x 1024 pixels. However, we use a region of interest of 1024 x 1024 pixels.

Fig. 9 - An example of a robot setup during a soccer game.

The architecture of the vision system that has been developed for these robots follows the generic one, presented in Fig. 8. For these robots, the objects of interest are: the ball, usually of a known solid color, the green field and the white lines necessary for the localization of the robot and the black obstacle, i.e. other robots that are on the field.

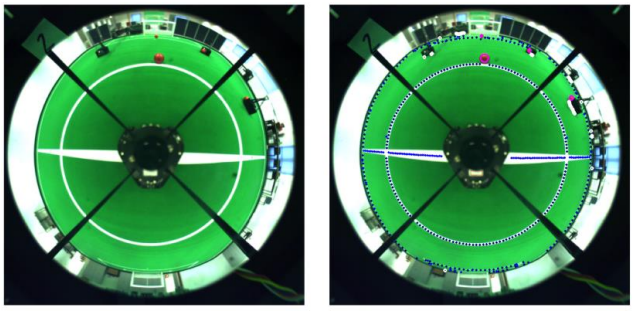
In this vision system, we use the following types and numbers of scanlines:

- 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.

As explained before, 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 labeled 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. 6). 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.

Visual examples of the detected objects in an image acquired by the vision system are presented in Fig. 10 and Fig. 11. As we can see, the objects of interest (balls, lines and obstacles) are correctly detected even when they are far from the robot. The ball can be correctly detected up to 9 meters away from the robot (notice that the robot is in the middle line of the field and the farthest 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.



a) b)

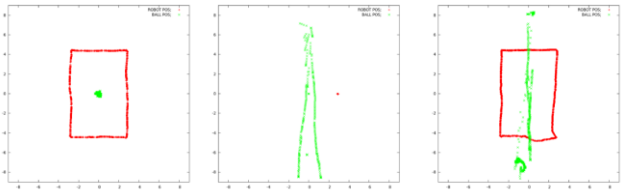
Fig. 10 – On the left, an image acquired by the omnidirectional vision system. On the right, the result of the color-coded object detection. The blue circles mark the white lines, the white circles mark the black obstacles and the magenta circles mark the orange blobs that passed the validation thresholds.

a) b) c)

Fig. 11 – 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.

Several game scenarios have been tested using the CAMBADA autonomous mobile robots. In Fig. 12(a) we present a graphic with the result of the ball 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 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. 12(b). The robot is stopped on the middle line and the ball is sent

across the field. This graphic 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. 12(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 when 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 [31]. 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.



a) b) c)

Fig. 12 – 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 cameras that have been used can provide 50fps at full resolution (1280 x 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 50fps and the experimental results show that the detection performance is not affected. A detailed study about the detection of colored objects in Bayer data can be seen in [33].

5.2 Delay Measurements between Perception and Action

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, we used the setup presented in Fig. 13.

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 to 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. This is the normal working mode of the robots in game play.

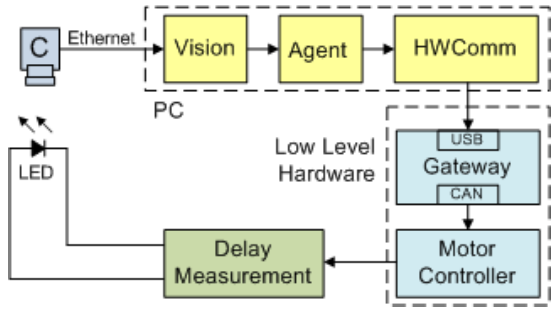
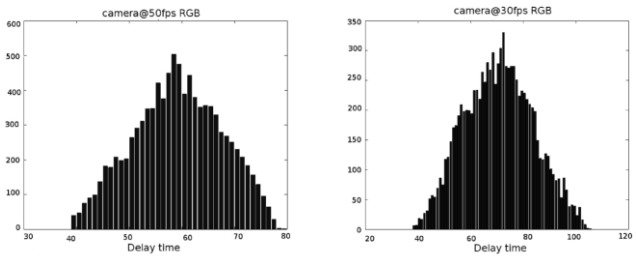


Fig. 13 – The blocks used in our measurement setup. These blocks are used by the robots during game play.

As presented in Fig. 14, the delay time between the perception and reaction of the robot significantly decreases when working at higher frame rates. The average delay at 30fps is 71 ms and at 50fps it is 60ms, which corresponds to an improvement of 16%.



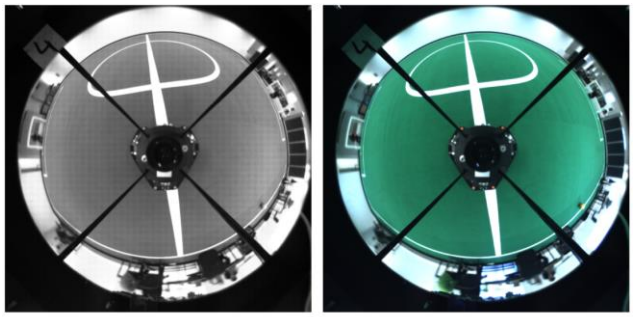
a) b)

Fig. 14 – Histograms showing the delay between perception and action on the CAMBADA robots. On the left, the camera is acquiring RGB frames and is working at 50fps (average = 60ms, max = 80ms, min = 39ms). On the right, the camera working at 30fps (average = 71ms, max = 106ms, min = 38ms).

The jitter verified reflects the normal function of the several modules involved, mainly because there is no synchronism between the camera, the processes running on the computer and the microcontrollers at the hardware level. If our vision system would have been synchronized with all the other processes running in the computational units of the robot (laptop and microcontrollers), for example using an external trigger signal, we would expect a similar delay in all the acquired frames. However, in the application described, we just guarantee that the other processes running in the laptop (decision and hardware communication processes) run after the vision system but

without synchronism with the camera and the software running in the microcontrollers.

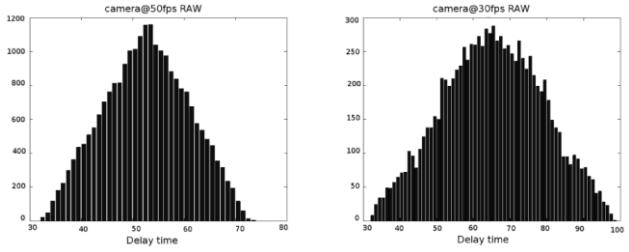
Since some digital cameras do not provide frame rates greater than 30fps in RGB, but allow the access to the raw data of the sensor at higher frame rates, and also because it is expected that the conversion between the raw sensor data and a specific color space in the camera will take some time, we performed some experiments for measuring the delay between perception and reaction of the robot accessing to the raw data. An example of a raw image acquired by the camera and the corresponding interpolated RGB version are presented in Fig. 15.



a) b)

Fig. 15 – In a) an example of a RAW image acquired by the camera (grayscale image containing the RGB information directly from the sensor - Bayer pattern. In b) the image a) after interpolation of the missing information for each pixel in order to obtain a complete RGB image.

We present the time results of these experiments in Fig.16. Similar to the results obtained when capturing RGB frames, the time between perception and reaction decreased when the frame rate increased. In this case, we have an improvement of 20% when using the camera at 50fps.



a) b)

Fig. 16 – Histograms showing the delay between perception and action on the CAMBADA robots. The camera is acquiring single channel frames, returning the raw data of the sensor (Bayer pattern). On the left, the camera is working at 50fps (average = 53ms, max = 74ms, min = 32ms). On the right, the camera working at 30fps (average = 66ms, max = 99ms, min = 32ms).

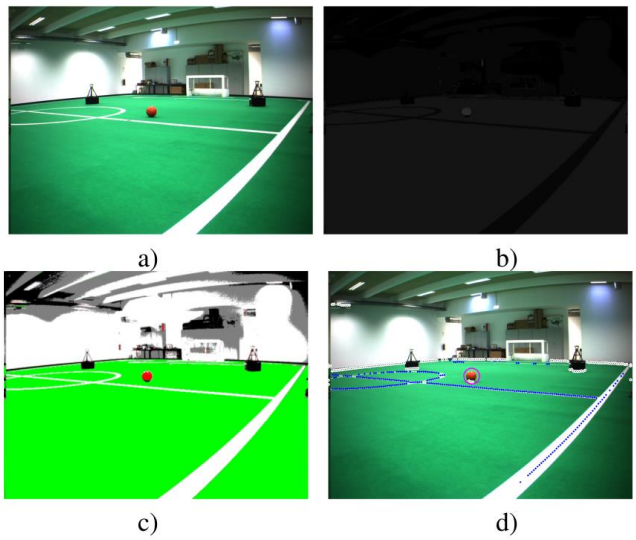
Another important result is the comparison between the delay time when acquiring frames in RGB and when accessing the raw data (Bayer pattern). Based on the graphics that we presented, we measured an improvement of 10% of the reaction time at 50fps and 7% at 30fps when using raw data. If the camera used by the robot could deliver raw data, these results prove that it is more advantageous. Moreover, we repeated the experiments with the robots detecting the ball using similar scenarios as the ones presented in Fig. 12. The experimental results obtained show that there is no loss in the detection performance of the robots and we observed an even better control in terms of motion. We provide in the library website [31] the sequences used in these experiments, as well as the binaries of the vision system used by our robots.

5.3 Perspective vision systems using the UAVision library

These results are complemented with another set of images acquired using a perspective camera that is fixed pointing to the soccer field (Fig. 17). The perspective camera is used for detecting the same objects of interest: the ball, the field lines and the obstacles/other robots and as future work will be used as ground truth for the team. The perspective camera used is a Firewire Point Grey Flea 2, 1 FL2-08S2C with a 1/3" CCD Sony ICX204 [32]. The images that we have used for the following experiments have a resolution of 1280 x 960 pixels. We present results achieved by using the UAVision library for two different cameras, but the architecture of the vision system is the same (the one presented in Fig. 8).

In this vision system, we use the following types and numbers of scanlines:

- 480 horizontal scanlines.
- 640 vertical scanlines.



a) b)

c) d)

Fig. 17 – Results obtained using a perspective camera: (a) - original image, (b) - index image, (c) - color classified image and (d) - image annotated with detected objects of interest.

The UAVision library has also been successfully employed for the detection of aerial balls, based on color information, using a Kinect sensor. This new implemented vision system, following the already presented architecture, was integrated in the software of the robots and a Kinect has been mounted on the goalie of the robotic soccer team. In this application, the depth information from the Kinect sensor has been used for discarding the objects of the color of the ball that are found farther than a certain distance (in this case, 7m were considered) in order to filter possible similar objects outside the field. An example of a ball detection by this system is presented in Fig. 18.

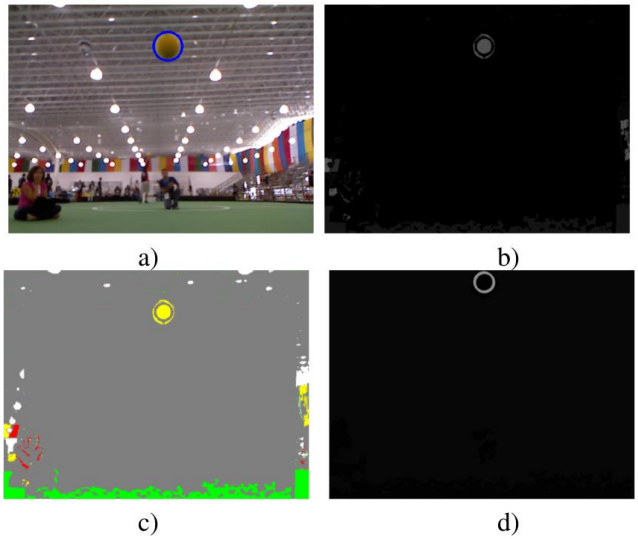


Fig. 18 – Results obtained with a Kinect sensor: a) original image with the ball detected, b) index image, c) color classified image, d)depth image.

5.4 Processing time

The processing time shown in Table. 1 proves that the vision systems built using the UAVision library are extremely fast. Taking as example the omnidirectional vision software of the CAMBADA robots, the full execution of the pipeline only takes on average a total of 15ms, allowing thus a framerate greater than 50fps if the digital camera supports it. Moreover, the maximum processing time that we measured was 18ms, 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 on a computer with a Intel Core i5-3340M CPU @ 2.70GHz 4 processor, processing images with resolutions from 640 x 480 pixels to 1280 x 960 pixels. In the implementation of these vision systems we did not use multi-threading. However, both image classification and the next steps can be parallelized if needed.

Table 1: Average processing times (in milliseconds - ms) measured for the vision systems developed using the UAVision library and presented in this section. The column "Omni" refers to the omnidirectional vision system used by the CAMBADA robots, using a resolution of 1024 x 1024 pixels. Column "Perpective"' refers to the perspective vision system developed with a Firewire Camera and a resolution of 1280 x 960 pixels. Column "Kinect"' refers to the vision system developed for the goalie using a Kinect camera, working with a resolution of 640 x 480 pixels. The filtering operation only exists for the Kinect vision system, as described in Section 5.3.

Operation	Vision Systems		
	Omni	Perspective	Kinect
Acquisition	1	1	1
Color Classification	5	10	12
Filtering	-	-	16
RLE	4	1	0
Blob Creation	2	0	0
Blob Validation	3	0	0
Total	**15**	**12**	**19**

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.

The time spent in the color classification step is greater in the perspective vision system, considering a resolution similar to the omnidirectional vision system, due to the fact that in the omnidirectional vision there is a mask defined and not all the pixels are color classified, as described before. This is one important feature of the proposed library.

The time spent by the omnidirectional vision system regarding RLE and Blobs modules are greater than the ones in the perspective vision system due to the existence of more scanlines to process, as presented in the description of the vision systems.

6. Conclusions

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 50fps when processing images with more than 1 mega pixels.

The experimental results in this paper show that the library can be easily used in different vision systems. The

similar vision pipeline was successfully used with three different cameras (Ethernet, Firewire and Kinect) in three different applications (omnidirectional and perspective vision systems for ground object detection and a vision system for aerial balls detection).

The use of this library in other applications is straightforward. The user only has to configure the colors of interest for the application and the validation procedure for the detected blobs.

The algorithms developed for this library take into consideration applications in which the computing resources are limited. As an example, the vision system of the soccer robot presented in this paper can be used in another robot with more limited processing capabilities after a small number of adjustments. These adjustments are related to the sparsity of the scanlines.

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.

Acknowledgments

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] RoboCup. http://www.robocup.org. Last visited - September, 2014.

[2] Aldebaran Robotics oficial website. http://www.aldebaran-robotics.com. Last visited - September, 2014.

[3] A. Neves, J. Azevedo, N. Lau B. Cunha, J. Silva, F. Santos, G. Corrente, D. A. Martins, N. Figueiredo, A. Pereira, L. Almeida, L. S. Lopes, and P. Pedreiras. CAMBADA soccer team: from robot architecture to multiagent coordination, chapter 2. I-Tech Education and Publishing, Vienna, Austria, In Vladan Papic (Ed.), Robot Soccer, 2010.

[4] R. Dias, F. Amaral, J. L. Azevedo, R. Castro, B. Cunha, J. Cunha, P. Dias, N. Lau, C. Magalhaes, A. J. R. Neves, A. Nunes, E. Pedrosa, A. Pereira, J. Santos, J. Silva, and A. Trifan. CAMBADA Team Description. RoboCup 2014, Joao Pessoa, Brazil, 2014.

[5] CMVision. http://www.cs.cmu.edu/~jbruce/cmvision/. Last visited - September, 2014.

[6] Adaptive Vision. https://www.adaptive-vision.com/en/home/. Last vis- ited - September, 2014.ADVANCES IN COMPUTER SCIENCE : AN INTERNATIONAL JOURNAL 12

[7] CCV. http://libccv.org/. Last visited - September, 2014.

[8] Roborealm. http://www.roborealm.com/index.php. Last visited - September, 2014.

[9] Antnio J. R. Neves, Armando J. Pinho, Daniel A. Martins, and Bernardo Cunha. An eficient omnidirectional vision system for soccer robots: from calibration to object detection. Mechatronics, 21(2):399–410, mar 2011.

[10] F.M.W Kanters, R. Hoogendijk, R.J.M. Janssen, K.J. Meessen, J.J.T.H. Best, D.J.H Bruijnen, G.J.L. Naus, W.H.T.M Aangenent, R.B.M. van der Berg, H.C.T. van de Loo, G.M. Heldes, R.P.A. Vugts, G.A. Harkema, P.E.J. van Brakel, B.H.M Bukkums, R.P.T. Soetens, R.J.E. Merry, and M.J.G. can de Molengraft. Tech United Eindhoven Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[11] U. P. Kappeler, O. Zweigle, H. Rajaie, K. Hausserman, A. Tamke, A. Koch, B. Eckstein, F. Aichele, D. DiMarco, A. Berthelot, T. Walter, and P. Levi. RFC Stuttgart Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[12] A. Ahmad, J. Xavier, J. Santos-Victor, and P. Lima. 3d to 2d bijection for spherical objects under equidistant fisheye projection. Computer Vision and Image Understanding, 125:172–183, 2014.

[13] M. Huang, X. Ge, S. Hui, X. Wang, S. Chen, X. Xu, W. Zhang, Y. Lu, X. Liu, L. Zhao, M. Wang, Z. Zhu, C. Wang, B. Huang, L. Ma, B. Qin, F. Zhou, and C. Wang. Water Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[14] H. Lu, Z. Zeng, X. Dong, D. Xiong, and S. Tang. Nubot Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[15] A.A.F Nassiraei, S. Ishida, N. Shinpuku, M. Hayashi, N. Hirao, K. Fu- jimoto, K. Fukuda, K. Takanaka, I. Godler, K. Ishii, and H. Miyamoto. Hibikino-Musashi Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[16] A. Trifan, A. J. R. Neves, B. Cunha, and N. Lau. A modular real- time vision system for humanoid robots. In Proceedings of SPIE IS&T Electronic Imaging 2012, January 2012.

[17] T. Rofer, T. Laue, C. Graf, T. Kastner, A. Fabisch, and C. Thedieck. B-Human Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[18] leader@austrian kangaroos.com. Austrian Kangaroos Team Description. RoboCup 2011, Istanbul, Turkey, 2011.

[19] TJArk.ofice@gmail.com. TJArk Team Description. RoboCup 2014, Joao Pessoa, Brazil, 2014.

[20] H. L. Akin, T. Mericli, E. Ozukur, C. Kavaklioglu, and B. Gokce. Cerberus Team Description. RoboCup 2014, Joao Pessoa, Brazil, 2014.

[21] D. Garcia, E. Carbajal, C Quintana, E. Torres, I. Gonzalez, C. Busta- mante, and L. Garrido. Borregos Team Description. RoboCup 2012, Mexico City, Mexico, 2012.

[22] S. Liemhetcharat, B. Coltin, C. Mericli, and M. Veloso. CMurfs Team Description. RoboCup 2014, Joao Pessoa, Brazil, 2014.

[23] E. Hashemi, O. A. Ghiasvand, M. G. Jadidi, A. Karimi, R. Hashemifard, M. Lashgarian, M. Shafiei, S. Mashhadt, K. Zarei, F. Faraji, M. A. Z. Harandi, and E. Mousavi. MRL

Team Description. RoboCup 2014, Joao Pessoa, Brazil, 2014.
[24] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. De- sign Patterns: Elements of Reusable Object-oriented Software. Addison- Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.
[25] Microsoft Kinect oficial website. http://www.microsoft.com/en-us/ kinectforwindows/. Last visited - September, 2014.
[26] Alina Trifan A. J. R. Neves and Bernardo Cunha. Self-calibration of colormetric parameters in vision systems for autonomous soccer robots. In in Proc. of RoboCup 2014 Symposium, 2014.
[27] B. Cunha, J. L. Azevedo, N. Lau, and L. Almeida. Obtaining the inverse distance map from a non-svp hyperbolic catadioptric robotic vision system. In Proc. of the RoboCup 2007, Atlanta, USA, 2007.
[28] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In in ICCV, pages 666–673, 1999.
[29] A. J. R. Neves, G. Corrente, and A. J. Pinho. An omnidirectional vision system for soccer robots. In Proc. of the EPIA 2007, volume 4874 of Lecture Notes in Artificial Inteligence, pages 499–507. Springer, 2007.
[30] IDS. https://en.ids-imaging.com. Last visited - September, 2014.
[31] UAVision. http://sweet.ua.pt/an/uavision/. Last visited - September, 2014.
[32] Pointgrey. http://www.ptgrey.com/products/flea2/flea2 firewire camera. asp. Last visited - September, 2014.
[33] António J. R. Neves, Alina Trifan, José Luís Azevedo. Time-constrained detection of colored objects on raw Bayer data. Proceedings of the 5th Eccomas Thematic Conference on Computational Vision and Medical Image Processing, VipIMAGE 2015, Tenerife, Spain, p. 301-306, October 2015.

Alina Trifan obtained her bachelor's degree from Universitatea Tehnica Cluj-Napoca, Romania in 2009 and the MSc degree from University of Aveiro, Portugal in 2011. She is currently a Ph.D student working on computer vision algorithms for robotic applications, under the supervision of Prof. Antonio Neves and Prof. Bernardo Cunha. Her research is focused on time constrained algorithms for vision systems of autonomous mobile robots.

António J. R. Neves received the Electronics and Telecommunications Engineering degree from the University of Aveiro, Portugal, in 2002 and the Ph.D. in Electrical Engineering from the University of Aveiro, in 2007. Currently, he is an Assistant Professor at the Department of Electronics, Telecommunications and Informatics of the University of Aveiro, and a researcher at the Intelligent Robotics and Inteligent Systems group of the Institute of Electronics and Telematics Engineering of Aveiro - IEETA. His main research interests are robotics (computer vision and multi-agent robotics), image and video coding (lossless coding, pre-processing techniques, images with special characteristics) and bioinformatics (microarray images and DNA coding). He published more than one hundred papers including several book chapters, journal articles and conference papers in the areas described above.

Bernardo Cunha received the Electronics and Telecommunications Engineering degree from the University of Aveiro, Portugal, in 1982 and the Ph.D. in Electrical Engineering from the University of Aveiro, in 1991. Currently, he is an Assistant Professor at the Department of Electronics, Telecommunications and Informatics of the University of Aveiro, and a researcher at the Intelligent Robotics and Inteligent Systems group of the Institute of Electronics and Telematics Engineering of Aveiro - IEETA. His main research interests are robotics (hardware design, computer vision and multi-agent robotics). He published several papers in the areas described above.

José Luís Azevedo received the Electronics and Telecommunications Engineering degree from the University of Aveiro, Portugal, in 1985 and the Ph.D. in Electrical Engineering from the University of Aveiro, in 1998. Currently, he is an Assistant Professor at the Department of Electronics, Telecommunications and Informatics of the University of Aveiro, and a researcher at the Intelligent Robotics and Inteligent Systems group of the Institute of Electronics and Telematics Engineering of Aveiro - IEETA. His main research interests are robotics (hardware design, electronics and multi-agent robotics). He published several papers in the areas described above.