Ivo dos Santos Pinheiro

Calibração Automática do Sistema de Visão da Equipa CAMBADA

Automatic Calibration of the CAMBADA Team Vision System

Ivo dos Santos Pinheiro

Calibração Automática do Sistema de Visão da Equipa CAMBADA

Automatic Calibration of the CAMBADA Team Vision System

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Manuel Bernardo Salvador Cunha, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor António José Ribeiro Neves, Professor Auxiliar Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

À minha família, namorada e amigos ...

o júri

presidente

Doutor Paulo Jorge dos Santos Gonçalves Ferreira
professor catedrático da Universidade de Aveiro

Doutor Luis Paulo Reis
professor auxiliar da Faculdade de Engenharia da Universidade do Porto

Doutor Manuel Bernardo Salvador Cunha
professor auxiliar da Universidade de Aveiro

Doutor António José Ribeiro Neves
professor auxiliar convidado da Universidade de Aveiro

agradecimentos

Aproveito esta oportunidade para agradecer a todas as pessoas que me ajudaram a levar esta dissertação a bom termo. Quero agradecer especialmente ao professor António José Neves, co-orientador da dissertação, pela sua constante disponibilidade, paciência e auxílio imprescindível na resolução de todos os problemas que fui encontrando ao longo desta dissertação. Quero agradecer também ao professor Manuel Bernardo Cunha, orientador da dissertação, com quem discuti alguns algoritmos de calibração de cores e de câmaras. Um agradecimento a todos os elementos do projecto CAMBADA, em especial ao grupo da visão, os quais durante a preparação, e durante as competições mantiveram um constante espírito de camaradagem e de equipa.

palavras-chave Visão robótica, processamento de imagem, câmaras digitais, câmaras omnidireccionais, calibração de câmaras digitais, calibração de cores

resumo O recurso a camâras digitais em aplicações robóticas tem vindo a crescer significativamente nos últimos anos. As principais áreas de aplicação destes robôs são a indústria e as aplicações militares, nos quais estas camâras são usadas como sensor que permite ao robô retirar a informação relevante do ambiente envolvente, a qual lhe permite tomar decisões. Para extrair informação de uma imagem, como a forma ou a cor de objectos, é importante que os parâmetros da câmara se encontrem bem calibrados. Esta dissertação está inserida no projecto CAMBADA (Cooperative Autonomous roBots w/ Advanced Distributed Architecture), desenvolvido pelo Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e apresenta um conjunto de algoritmos para calibração dos principais parâmetros de uma câmara digital, bem como a calibração de cores de interesse associadas a uma determinada aplicação das câmaras. Esta dissertação propõe dois algoritmos que permitem aos robôs, de forma automática, calibrar os parâmetros das câmaras do seu sistema de visão. Estes algoritmos baseiam-se em informação extraída das imagens adquiridas pelas câmaras e na utilização de compensadores. São propostos ainda três métodos para calibrar as cores, dois deles de forma manual mas intuitivos para o utilizador, e por último um algoritmo automático baseado em detecção de contornos e crescimento de regiões.

keywords Robotic vision, image processing, digital cameras, omnidirectional cameras, camera calibration, color calibration

Abstract In the past few years the use of digital cameras in robotic applications has been increasing significantly. The main areas of application of this robots are the industry and military applications, where these cameras are used as a sensor that allows the robot to take the relevant information of the surrounding environment, allowing it to make decisions. To extract information from the acquired image, such as shapes or colors, the camera calibration procedure is very important. This thesis is inserted in the CAMBADA (Cooperative Autonomous Mobile roBots w/ Advanced Distributed Architecture) project, developed by the Department of Electronics, Telecommunication and Informatics of the University of Aveiro, and presents several algorithms to calibrate the most important parameters of a digital camera and the colors of interest for a specific application. This thesis proposes two algorithms that allow the robots to calibrate autonomously the parameters of their camera systems. These algorithms are based on the information extracted from the acquired image and in the use of controllers. It also proposes three methods for color calibration, two of them are manual but intuitive to the user, and the last one an automatic algorithm based on edge detection and region growing algorithms.

Contents

1 Introduction **1**

1.1 The RoboCup MSL case 2

1.2 Thesis contributions 4

2 Digital Cameras **5**

2.1 CCD sensor 5

2.2 Camera parameters 5

2.2.1 White-balance 6

2.2.2 Brightness 7

2.2.3 Contrast 7

2.2.4 Gain 8

2.2.5 Exposure 8

2.2.6 Shutter 9

2.2.7 Gamma 9

2.2.8 Saturation 9

2.2.9 Hue 10

2.3 Color spaces 11

2.3.1 RGB 11

2.3.2 YUV 11

2.3.3 HSV 14

3 Autocalibration of camera parameters **17**

3.1 Camera calibration algorithm 18

3.2 Autocalibration of camera parameters using PI controllers 21

3.2.1 Run-time auto-calibration 26

4 Color calibration **31**

4.1 Automated algorithm for color calibration 33

5 Conclusion **37**

A AutoCalibration manual **39**

A.1 Introduction 39

A.2 How to use the application 40

B LibAutoCalib **43**

Bibliography **45**

Chapter 1

Introduction

In the past few years the use of digital cameras in robotic applications has been increasing significantly. The main areas of application of this robots are the industry and military applications, where these cameras are used as a sensor that allows the robot to take the relevant information of the surrounding environment, allowing it to make decisions.

To extract information from the acquired image, such as shapes or colors, the camera calibration procedure is very important. If the camera is wrongly calibrated, the image details are lost and it is impossible to recognize anything based on shape or color, as can be seen in Fig. 1.1.

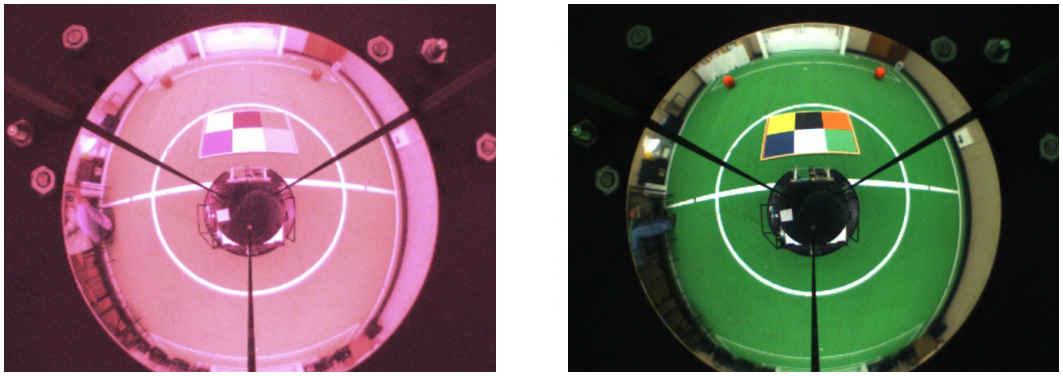


Figure 1.1: On the left, an example of an image acquired by one of the CAMBADA team robots with its camera wrongly calibrated. On the right, an image correctly calibrated.

The major applications of cameras in robotics vision occur within controlled environments, which means that the intensity and the type of light is always constant. In this case the parameters of the cameras only need to be adjusted once because the environment luminance will be almost constant.

This thesis is inserted in the CAMBADA (Cooperative Autonomous Mobile roBots w/ Advanced Distributed Architecture) project, developed by the Department of Electronics, Telecommunication and Informatics of the University of Aveiro [1]. CAMBADA is a team of soccer robots that participates in the Middle Size League (MSL) of ROBOCUP [2]. It is a multidisciplinary project, that involves knowledge of mechanics, artificial intelligence, power electronics, instrumentation, computer architecture, telecommunications, software engineer, artificial vision and cooperation, among others.

CAMBADA robots have a hybrid vision system integrating an omnidirectional and a perspective camera. The omnidirectional part of the vision system [3] is based on a catadioptric configuration implemented with a firewire camera (with a 1/3" CCD sensor and a 4.0mm focal distance lens) and a hyperbolic mirror. The perspective camera uses a low cost firewire camera (BCL 1.2 Unibrain camera with a 1/4" CCD sensor and a 3.6mm focal distance lens). The omnidirectional camera works at 30 frames per second (fps) in YUV4:2:2 mode with a resolution of 640×480 pixels. The front camera works at 30 fps in YUV4:1:1 mode with a resolution of 640×480 pixels.

The omnidirectional vision system is used to find the ball, to detect the presence of obstacles and white lines. In the case of the MSL competition of RoboCup, most of the teams adopt this approach for the vision system of the robots. The perspective vision is used to find the ball and obstacles in front of the robot at larger distances, which are difficult to detect using the omnidirectional vision system, due to its very limited spacial resolution at distances greater than 5 to 6 meters.

A set of algorithms has been developed to extract the color information of the acquired images and, in a second phase, extract the information of all objects of interest. The vision system architecture uses a distributed paradigm where the main tasks, namely, image acquisition, color extraction, object detection and image visualization, are separated in several processes. Efficient color extraction algorithms have been developed based on lookup tables and a radial model for object detection. The vision system is fast and accurate having a constant processing time independent from the environment around the robot [4].

1.1 The RoboCup MSL case

The ultimate goal of the RoboCup project is, by 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer [2]. It means in a near future, the robots will have to play under natural light conditions and in outdoor fields. This introduces many obstacles to the robots because they must be able to play either

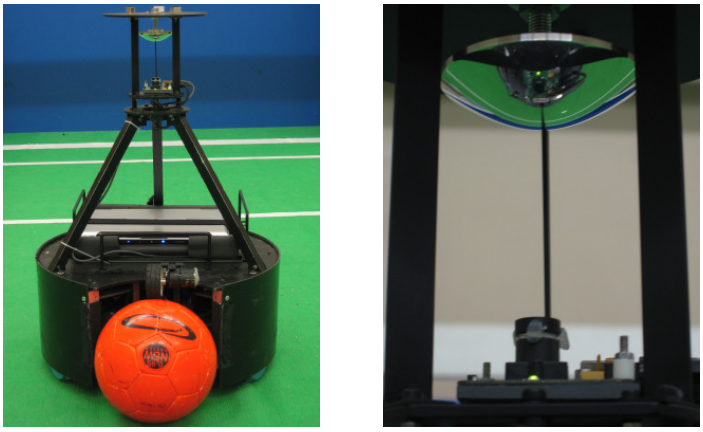


Figure 1.2: One of the robots used by the CAMBADA MSL robotic soccer team and its vision system.

under controlled lighting conditions, as is the case of artificial illumination, as well in non-controlled lighting conditions, such as in outdoor fields. In outdoor fields the illumination can change slowly during the day, due to the movement of the sun, as well as fast in short periods of time due to a partial and temporally covering of the sun by clouds. It means that the robots, have to adjust the colors, in real time, its color segmentation values as well as its camera parameters to adapt to new lighting conditions [5].

Image analysis in the RoboCup domain is simplified, since objects are color coded. Black robots play with an orange ball on a green field that has white lines. Thus, the color of a pixel is a strong hint for object detection. This fact can be exploited by defining color classes, using a look-up table (LUT) for fast color classification. The table consists of 16 777 216 entries (one of each 2^{24} possible color combinations - 8 bits for red, 8 bits for green and 8 bits for blue). Each entry is 8 bits wide, which means that the LUT occupies 16 MBytes in total. If another color space is used, the table size is the same, changing only the "meaning" of each component. Each bit expresses whether the color is within the corresponding class or not. This means that a certain color can be assigned to several classes at the same time. To classify a pixel, we first read the pixel's color and then use the color as an index into the table.

The color calibration is performed in HSV (Hue, Saturation and Value) color space due to its special characteristics. In the current setup, the image is acquired in RGB or YUV format and then is converted to an image of labels using the appropriate LUT [4].

As far as we can understand from the published work made by the other teams of RoboCup MSL, most of them don't have any software to autocalibrate the camera, which means their cameras are only adjusted manually at the beginning of each game. Some of the teams,

however, claim to have an automatic process for color calibration running offline over a pre-acquired video.

In the team description papers of Brainstormers Tribots [6], NuBot [7], Tech United [8] and Robofoot ÉPM [9], some autocalibration algorithms are mentioned, although they don't present any details about them.

1.2 Thesis contributions

Until the beginning of this work, the CAMBADA team didn't have an automatic application to calibrate the vision system. The camera parameters were adjusted manually and this process usually took much time and required an expert person. The color calibration was made by acquiring a video and, in offline mode, the video was processed and the color range for each color was adjusted.

This thesis proposes two algorithms for camera parameters calibration, both automatically performed by the robots. The first one obtains satisfactory results, but presents some control problems along the time. Moreover, it doesn't use any measure of quality to ensure the correct calibration and it doesn't evaluate the whole image.

The second proposed algorithm attains better results and has solved the problems referred to in the first algorithm. It's based on a PI controller, responsible for the camera parameters update, and uses a set of measures, namely ACM, Average, Entropy and MSV to evaluate the quality of the acquired images. This improves the obtained results and guarantees that the images have nearly the same characteristics. Experimental results obtained by this algorithm show that the quality measures of the image acquired always converge to the same values. Moreover, it is possible to apply this algorithm in run-time in order to guarantee that, even while varying the illumination of the environment, the relevant colors of the image will remain the same.

A method to calibrate colors using a HSV histogram is also proposed. It allows to visualize easily what is the color range of a certain color class and if a pixel is in or out of the selected color range. This method uses the histogram of the three components, hue, saturation and value to select the desired color range for each color class.

Finally an automatic method to calibrate the color classes using a canny edge detector is also proposed. For each region obtained after applying the edge detector, a region growing algorithm is applied in order to obtain the color range of that region. Each color class has a value of hue, saturation and value that is used as seed for the region growing algorithm.

Chapter 2

Digital Cameras

A digital camera is a device that captures video or photographs digitally by recording the images captured by a light sensitive sensor. It is formed basically by a lens, a color sensor, software for image treatment and hardware capable of process and transmit the captured images, usually designated by frames [10, 11].

2.1 CCD sensor

Most of the colors of the visible spectrum could be reproduced by adding distinct parts of red, green and blue light. This property is used by most of CCD sensors to capture color images. It consists in a square grid of capacitors, which are charged by a photosensitive element covered by a filter that only allows one component of light (red, green and blue) to reach the sensor (see Fig. 2.1). The image is formed measuring the charge of each capacitor by a control circuit.

For each pixel, the CCD doesn't acquire the information of the three components. Instead, a Bayer pattern is used. It is a repeating 2×2 mosaic pattern of light filters, with green ones at opposite corners and red and blue in the other two positions. The predominance of green takes advantage of properties of the human visual system, which determines brightness mostly from green information and is far more sensitive to brightness than to hue or saturation.

2.2 Camera parameters

For any particular application the quality of an image acquired by a camera is dependent of many factors, such as illumination, camera lens and, the most important, the camera

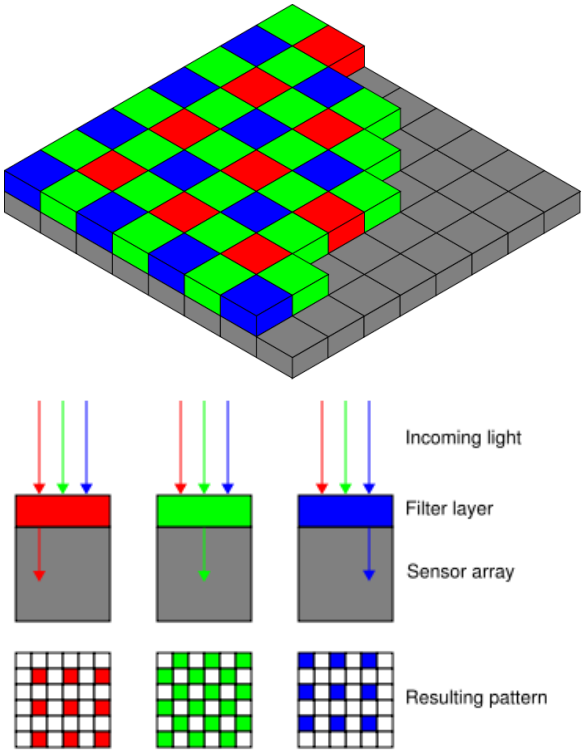


Figure 2.1: The Bayer arrangement of color filters on the pixel array of an image sensor [12].

parameters. In the following sections it will be explained the common parameters of a camera and their effects in the image.

2.2.1 White-balance

This parameter is one of the most important configurable parameters of the camera. The image colors appear different depending on the illumination under which the image was taken. This is due to the fact that different light sources have different color temperatures. The adjustment of this parameter should be performed in order to make a white source to appear white on the image under different light conditions. Usually, when this parameter hasn't been adjusted, the images have a red or blue tonality, as presented in Fig. 2.2. The compensation of this effect can either be automatically performed by the camera, or can be made manually. The adjustment is performer by correcting the red and blue channels gain.

Figure 2.2: Examples of images acquired in a robotic soccer field using different values for the white-balance. On the left, an image acquired with a high value of the blue gain. In the center, an image with the white-balance correctly calibrated. On the right, an image acquired with a high value of the red gain.

2.2.2 Brightness

This parameter adjusts the black level of an image, which means that an offset is added or subtracted for each pixel. This parameter, in the limit, could lead to images brighter or darker (see Fig. 2.3). Usually, this parameter must be set manually.

Figure 2.3: Examples of images acquired in the robotic soccer field with different values of brightness. On the left, an image acquired with a low value of brightness. In the center, an image correctly acquired. On the right, an image acquired with a high value of brightness.

2.2.3 Contrast

This parameter is responsible to turn the bright colors more bright and the dark colors more dark. In a high contrast image, edges can be seen more clearly and the different elements of an image are accented. In a low contrast image, the brightness of different elements is nearly the same and it's hard to make out detail (see Fig. 2.4).

Figure 2.4: Examples of images acquired in the robotic soccer field with different values of contrast. On the left, an image acquired with a low contrast. In the center, an image correctly acquired. On the right, an image acquired with a high contrast.

2.2.4 Gain

This parameter is usually implemented by hardware. When the charge of each CCD capacitor is measured, its value is amplified by hardware, by a "gain" factor before the quantization step. Increasing this factor makes the image brighter and increases the contrast but adds noise to the image, due to the fact that the original noise of the image is also amplified. Usually, this parameter can be set to the automatic mode, which means that the camera automatically control the value of the gain by an algorithm that evaluates the brightness of the last frame (see Fig. 2.5).

Figure 2.5: Examples of images acquired in the robotic soccer field with different values of gain. On the left, an image acquired with a low value of gain. In the center, an image correctly acquired. On the right, an image acquired with a high value of gain.

2.2.5 Exposure

Exposure is the total amount of light allowed to fall on the image, which means that it is related with the time that the CCD sensor is exposed to the light in each frame. A high value of exposure will lead to images more bright, which is the same effect of the gain but without adding noise. This time is limited by the frame rate. For example, if the camera is acquiring

images at 15 frames per second, the CCD sensor could only be exposed 1/15 seconds (see Fig. 2.6).

Figure 2.6: Examples of images acquired in the robotic soccer field with different values of exposure. On the left, an image acquired with a low value of exposure. In the center, an image correctly acquired. On the right, an image acquired with a high value of exposure.

2.2.6 Shutter

The shutter speed determines the amount of time that the shutter of a camera is opened and is therefore similar to the exposure parameter. The adjustment of this parameter allows to control the movement of an object in a scene. A low shutter will freeze the object and a high shutter will make it looks blurred as the object moves (see Fig. 2.6).

2.2.7 Gamma

This parameter can be implemented by hardware or software in the camera. It is usually used to force an image to have the same visual aspect on different monitors. For each pixel, the operation $Pc = Po^{\frac{1}{gamma}}$ is performed, where Pc and Po are the corrected and the original pixel values respectively. The effect of this parameter is to turn darker or brighter the dark pixels related to the bright pixels. The adjustment of this parameter is usually set manually (see Fig. 2.7).

2.2.8 Saturation

The saturation of a color refers to how vibrant a color is. A low saturated color is near to gray, leading to an image where the colors look "washed out". On the opposite, a high saturated color becomes clear, resulting in image where the colors are very intense (see Fig. 2.8).

Figure 2.7: Examples of images acquired in the robotic soccer field with different values of gamma. On the left, an image acquired with a low value of gamma. On the right, an image acquired with a higher value of gamma.

Figure 2.8: Examples of images acquired in the robotic soccer field with different values of saturation. On the left, an image acquired with a low value of saturation. In the center, an image correctly acquired. On the right, an image acquired with a high value of saturation.

2.2.9 Hue

This parameter allows to introduce an offset in the colors spectrum. Usually, the camera doesn't use this parameter by default and doesn't have an automatic mode (see Fig. 2.9).

Figure 2.9: Examples of images acquired in the robotic soccer field with different values of hue On the left, an image acquired with a low value of hue. In the center, an image correctly acquired. On the right, an image acquired with a high value of hue.

2.3 Color spaces

A color space is a mathematical model to digitally represent the colors. There exist many color spaces. Each one has different characteristics and is suitable for different applications. Some of them use the three primary colors (red, green and blue) while others use different concepts, such as the case of hue, saturation and value, related to the human visual system, or luminance and two chromatic components, such as the one used for television [13, 14].

2.3.1 RGB

The name RGB comes from Red, Green and Blue initials, that are the three emissive primary colors. This color space uses the additive properties of the colors allowing to obtain almost all possible colors of the visible spectrum. This color space is used mainly on computer graphics, as well in TV and video. This color space isn't intuitive to human perception, but have the advantage to be very simple to implement and compute. The RGB could be visualized as a cube with three axis, each one corresponding to red, green and blue (see Fig. 2.10). The bottom corner, when $red = green = blue = 0$ is black, while on the opposite corner, where $red = green = blue = 255$ (for a 8 bit per channel display system) is white. In the diagonal vector from black to white, are represented the gray levels, characterized by having the same amount of each primary color.

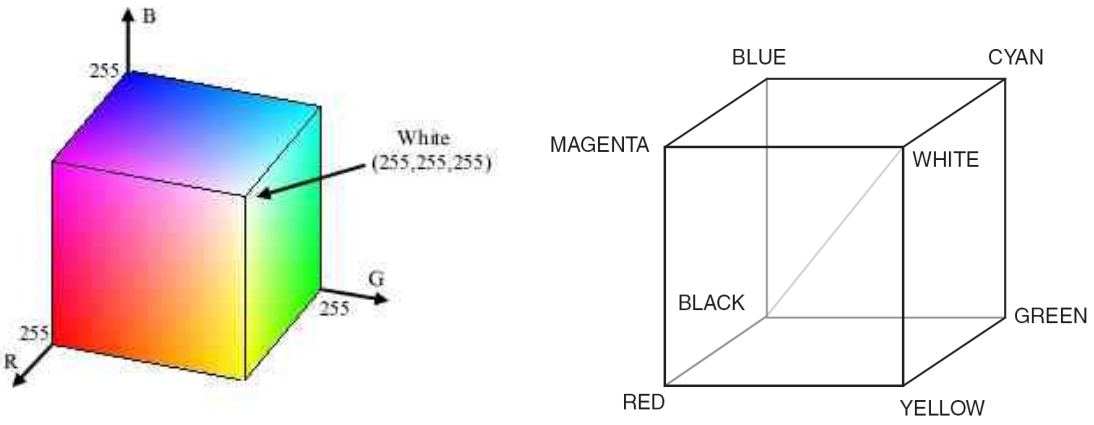


Figure 2.10: The RGB color space representation [15, 13].

2.3.2 YUV

The YUV color space separates the information of a color into its luminance (Y) and two chromatic components, namely U and V (see Fig. 2.11). It is used by the Phase Alternation

Line (PAL), National Television System Committee (NTSC), and Sequential Couleur Avec Mémoire (SECAM) composite color video standards. The biggest advantage of this color space relatively to the RGB is that the human visual perception (in terms of both intensity and spacial resolution) of the Y component is greater than the perception of the U and V components. Due to this knowledge, the U and V components usually are sub-sampled (Fig. 2.12) and the image requires a smaller bandwidth to be transmitted an it is more efficiently stored. This process introduces some loss of quality in the reconstructed image but it is assumed that the human eye can't distinguish the differences. This is used as a first step for image compression.

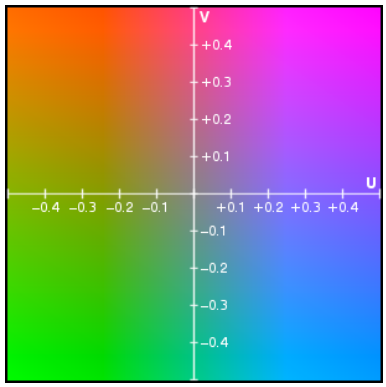


Figure 2.11: The U-V color plane considering Y=127 [16].

To convert RGB to YUV the following equations are used [13]:

$$
\begin{aligned}
Y &= (0.257 * R) + (0.504 * G) + (0.098 * B) + 16 \\
V &= (0.439 * R) - (0.368 * G) - (0.071 * B) + 128 \\
U &= -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128
\end{aligned}
$$

To convert YUV to RGB the following equations are used [13]:

$$
\begin{aligned}
R &= 1.164(Y - 16) + 1.596(V - 128) \\
G &= 1.164(Y - 16) - 0.813(V - 128) - 0.391(U - 128) \\
B &= 1.164(Y - 16) + 2.018(U - 128)
\end{aligned}
$$

Sub-sampling schemes

Because the human eye is less sensitive to the chrominance than the luminance, bandwidth could be optimized by storing more information about the luminance than about chrominance. In following, the various sub-sampling schemes will be explained. The modes YUV444,

YUV422 and YUV411 are acquired by digital cameras using a packet representation. The mode YUV420 is acquired using a planar representation, i. e. the camera returns each image component (the full image) at a time.

YUV444: each component has the same resolution without loss of information. The information acquired by the camera has the following mapping:

$$Y_0U_0V_0Y_1U_1V_1Y_2U_2V_2Y_3U_3V_3$$

Each pixel will be mapped according to the following:

$$[Y_0U_0V_0]\,[Y_1U_1V_1]\,[Y_2U_2V_2]\,[Y_3U_3V_3]$$

YUV422: the U and V components are spatially sampled half times relatively to the Y component, which means that the horizontal resolution is an half than the YUV444. The information acquired by the camera has the following mapping:

$$U_0Y_0V_1Y_1U_2Y_2V_3Y_3$$

Each pixel will be mapped according to the following:

$$[Y_0U_0V_1]\,[Y_1U_0V_1]\,[Y_2U_2V_3]\,[Y_3U_2V_3]$$

YUV411: the horizontal resolution of U and V components is divided by four. The video in this format uses 6 bytes to store each macropixel (one rectangle with 1×4 pixels).

$$U_0Y_0Y_1V_2Y_2Y_3$$

Each pixel will be mapped according to the following:

$$[Y_0U_0V_2]\,[Y_1U_0V_2]\,[Y_2U_0V_2]\,[Y_3U_0V_2]$$

YUV420: the horizontal and the vertical resolution of U and V components are divided by two. The video in this format uses 6 pixels to store each macropixel (one square with 2×2 pixels).

In Fig. 2.12, it is presented a graphical representation of the several YUV sub-sampling schemes described above.

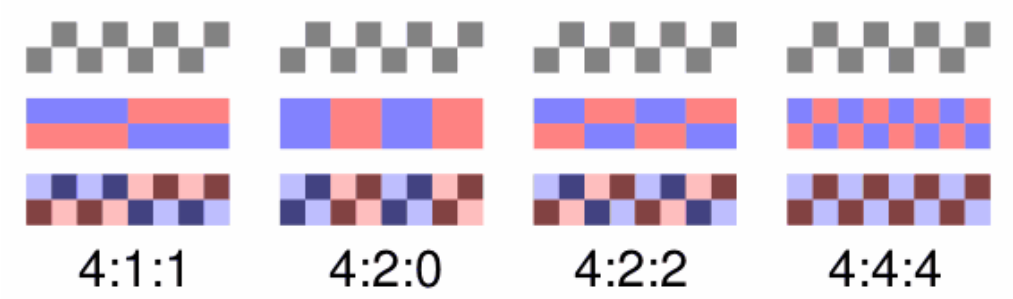


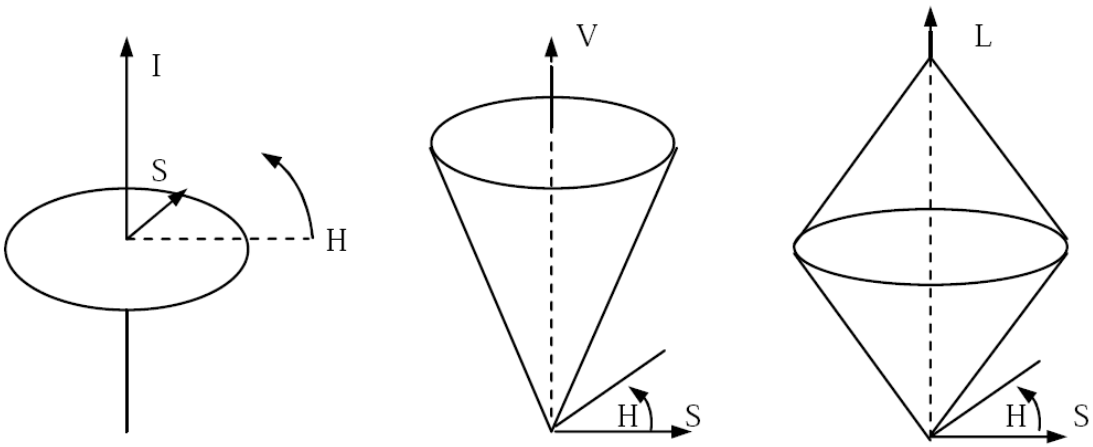
Figure 2.12: Representation of the different sub-sampling schemes of the YUV color space [17].

2.3.3 HSV

The color receptors in the human eye, known as cones, from different areas of the spectrum, with the greatest sensitivity in the blue, green and red part of it. The color information signals acquired by the cones are then further processed in the visual system. Nevertheless, a person cannot make intuitive estimates of the blue, green and red components of any particular color. On the other hand, in the perception process, a human can easily recognize basic attributes of color such as intensity (brightness, lightness) I, saturation S and hue H. The hue represents the impression related to the dominant wavelength of the color stimulus. The saturation corresponds to relative color purity. Colors with zero saturation are gray levels. Maximum intensity is sensed as pure white, minimum intensity as pure black. The H, S, I components of the HSI color model are calculated from formula expressing approximately the psychophysical sense of these notions from the RGB coordinate system to a cylindrical model of perceptions (see Fig. 2.13a)) [18].

Two other derivations of the generic HSI color space are applied usually in computer graphics and image processing: HSV and HLS. Figures 2.13b) and 2.13c) illustrate the geometric interpretation of these models. They differ of the original HSI model in the expression of the intensity and saturation values. For the HSV model the colors become less saturated when the intensity approaches minimal level. In HLS the colors become less saturated when the intensity approaches minimal or maximal levels. Important advantages of the HSI, HSV and HLS models over other color spaces are good compatibility with the human intuition of color. They are intuitive color spaces and allow an easy separation between chromatic values and achromatic values.

For image processing, a color system based on the human perception of color (HSI, HLS or HSV) may be more beneficial namely when human interaction is necessary. On the other hand



a) b) c)

Figure 2.13: Graphical representation of a) HSI, b) HSV and c) HLS color spaces [18].

if the image source is acquired using a YUV sampling scheme with lower resolution for the chromatic signals these models also allow an easy preservation of the luminance component in the form of the I,V or L signals. Due to this fact, in the field of robotic applications, HSV is the color space most used for color classification which simplifies and makes more accurate the selection of color ranges for each color of interest [14]. This color space is represented in Fig. 2.14. .

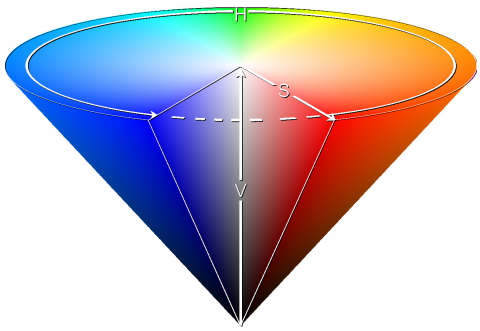


Figure 2.14: Conical representation of the HSV color space [19].

In this work it is used the HSV color space. To convert RGB to HSV it used the following equations:

$h \epsilon [0, 359]$
$s \epsilon [0, 1]$
$v \epsilon [0, 1]$

$max = max(r, g, b)$

$min = min(r, g, b)$

$$h = \begin{cases} 0 & \text{, if max=min} \\ 60° \times \frac{g-b}{max-min} + 0° & \text{, if max=r and g} \geq \text{b} \\ 60° \times \frac{g-b}{max-min} + 360° & \text{, if max=r and g} < \text{b} \\ 60° \times \frac{g-b}{max-min} + 120° & \text{, if max=g} \\ 60° \times \frac{g-b}{max-min} + 240° & \text{, if max=b} \end{cases}$$

$$s = \begin{cases} 0 & \text{, if max=0} \\ \frac{max-min}{min} & \text{, otherwise} \end{cases}$$

$v = max$

To convert HSV to RGB it used the following equations:

$h_i = \lfloor \frac{h}{60} \rfloor \, mod \, 6$

$f = \frac{h}{60} - \lfloor \frac{h}{60} \rfloor$

$p = v \times (1 - s)$

$q = v \times (1 - f \times s$

$t = v \times (1 - (1 - f) \times s)$

Compute color vector (r, g, b)

$$(r, g, b) = \begin{cases} (v, t, p) & \text{, if } h_i = 0 \\ (q, v, p) & \text{, if } h_i = 1 \\ (p, v, t) & \text{, if } h_i = 2 \\ (p, q, v) & \text{, if } h_i = 3 \\ (t, p, v) & \text{, if } h_i = 4 \\ (v, p, q) & \text{, if } h_i = 5 \end{cases}$$

Chapter 3

Autocalibration of camera parameters

The calibration of the camera is crucial for color-based object detection and has to be performed when environmental conditions change, for example when using different soccer fields or when lighting conditions changes.

Using the camera in auto mode has several problems and cannot be used is some robotic applications. In the case of the CAMBADA robots and due to the fact that the environment around these robots has specific characteristics, such as the green field and the black body of the robots that fills the most part of the image (see Fig. 3.1), the auto mode of the cameras do not perform well.

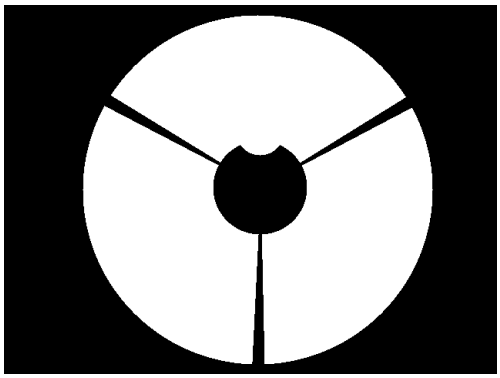


Figure 3.1: An example of a mask used to select the pixels to be processed by the omnidirectional vision system. White points represents the area that will be processed.

The image acquired by the camera in auto mode is overexposed due to the large black areas existing in the image and the white-balance is not correctly calibrated, leading to erroneous colors in the image. This is due to the fact that most of cameras use algorithms that assume that the pixels of an image are distributed along the visible color spectrum.

Until the beginning of this work, the camera calibration was performed manually by a specialized person. This calibration process usually took about 10 minutes for the full configuration of each robot's vision system. So an algorithm that calibrates automatically the camera is a great help to optimize the process. In the following it will be explained the developed algorithms and discussed the experiments results obtained.

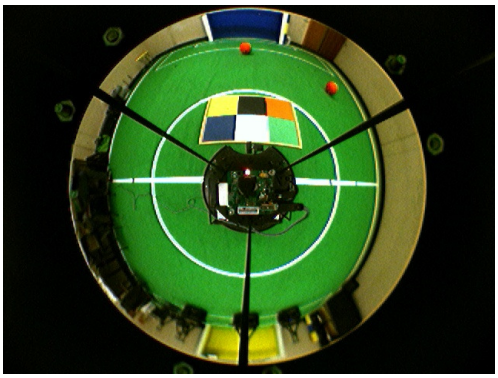


Figure 3.2: An example of an image acquired by the camera of one robot of the CAMBADA team.

3.1 Camera calibration algorithm

The first developed algorithm doesn't analyze the entire image but uses a selection of two areas with specific characteristics, namely a black and a white areas. It only adjusts the white-balance, the gain and brightness parameters. It is assumed that, when the camera is calibrated, the selected areas have the following characteristics:

- the white area should be white:
  - in the YUV color space this means that the average value of U and V should be 127. If the white-balance is not correctly configured, these values are different from 127 and the image does not have the correct colors;
  - in the RGB color space the average value should equal for the three components and near (255, 255, 255). If not, the entire image is either too dark and gain parameter needs to be adjusted, or there is a predominance of one of the primary colors and the white-balance needs to be adjusted;
- the black area should be black – in the RGB color space this means that the average values of R, G and B should be close to zero. If the brightness is too high we observe that the black area becomes blue, resulting in a degradation of the image.

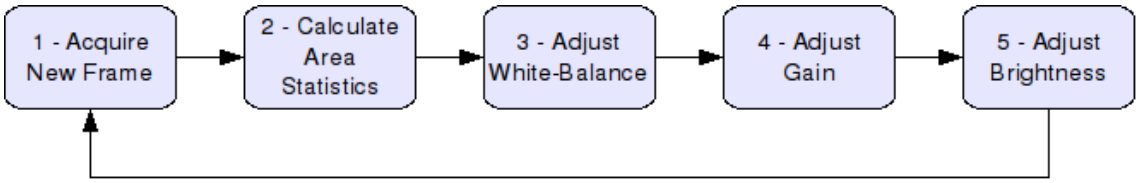


Figure 3.3: Diagram of the autocalibration algorithm.

The algorithm performs as follows:

1. A new frame is acquired by the camera.
2. For the selected black and white areas, it is calculated the maximum, minimum and the average values of Y, U and V in the YUV color space and R, G and B in the RGB color space.
3. The adjustment of white-balance is made based on the average value of white area in the YUV color space. If the U value is smaller than 127 the blue gain of white-balance is incremented, else if it is greater than 127 the blue gain is decremented. The same is made for red gain of white-balance, but using as reference the mean value of V component of the white area.
4. The adjustment of the gain parameter of the camera is based by the maximum and minimum values of the R, G and B values in the white area. If any of the maximum values is equal to 255 the gain will be decremented. On the other case, if any of the minimum values is smaller than 200 the gain is incremented.
5. The adjustment of brightness is performed using the black area and it is adjusted in order to guaranty that the blue value in the RGB color space is smaller than 5 and, at the same time, guaranty that any of the mean values of R, G and B never is greater than or equal to 2.

The changes in the parameters, as described above, are made using some predefined constants obtained experimentally. This is a possible drawback of the algorithm.

The experimental results shows that this algorithm presents a satisfactory performance. However it also shows some problems, namely it takes some time to converge and only partial evaluation of the image to obtain useful information. Moreover, it depends on the initial parameters set to the camera. Some results obtained with this algorithm are presented in Fig. 3.4.

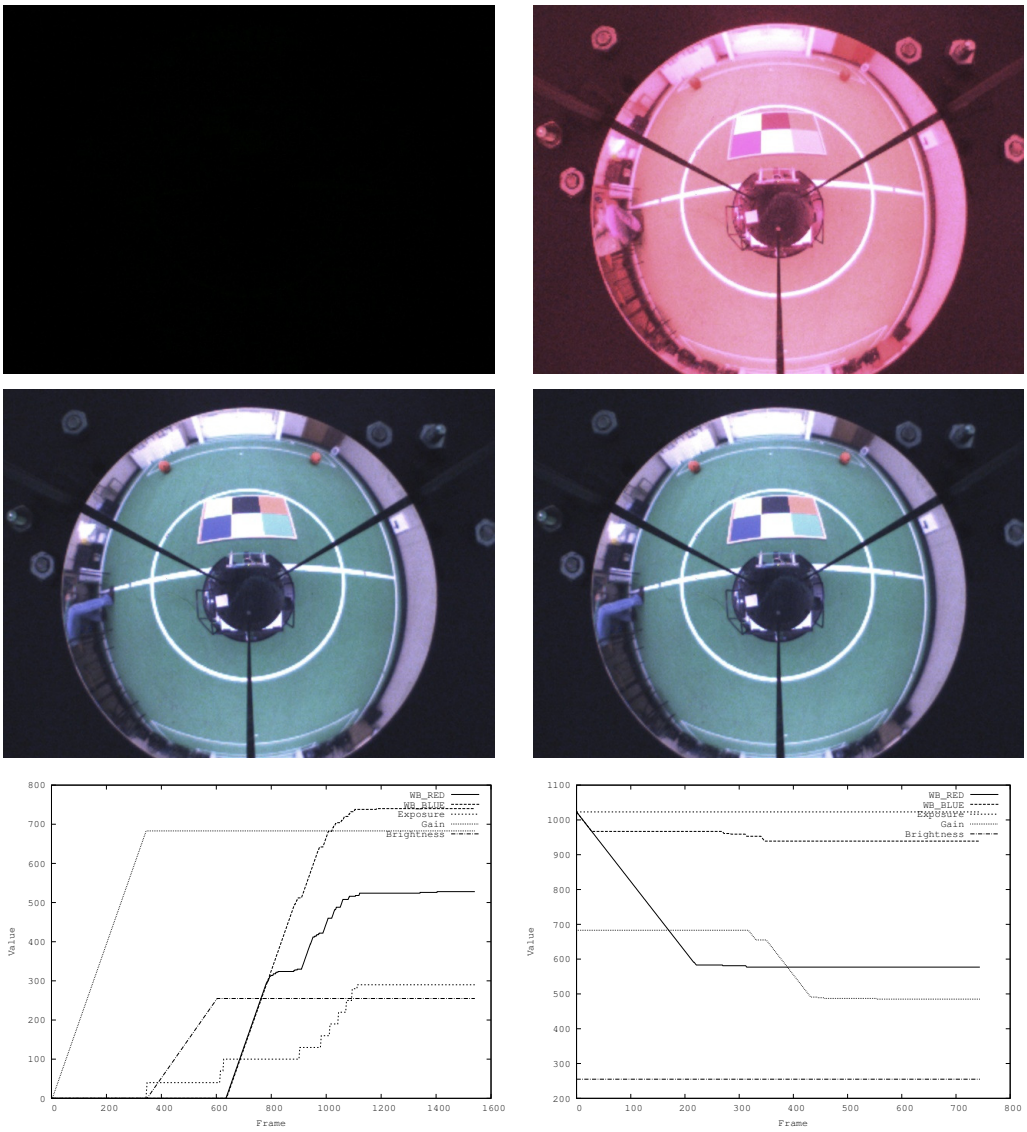


Figure 3.4: Experimental results using the automated calibration procedure described in this section. On the left, results obtained starting with all the parameters of the camera set to zero. On the right, results obtained with all the parameters set to the maximum value. On the top, the initial image acquired. In the middle, the image obtained after applying the automated calibration procedure. The last row contains graphics showing the evolution of the parameters along the time.

3.2 Autocalibration of camera parameters using PI controllers

This section presents an evolution of the algorithm described in the previous section. It also uses two reference areas in the image, namely a white area to calibrate the white-balance and a black area to calibrate the brightness. However, the histogram of the intensities of the image is used to calibrate the exposure and the gain of the camera.

An histogram of the intensities of an image is a graphical representation of the intensity pixel value shown as bars corresponding to the count of the occurrences of all possible values in the image. For an image represented using 8 bits per pixel, the possible values are between 0 and 255. Image histograms can indicate the nature of the light conditions, the exposure of the image and whether it is underexposed or overexposed.

The assumptions used by the proposed algorithm are:

- the white area should be white – in the YUV color space this means that the average value of U and V should be 127. If the white-balance is not correctly configured, these values are different from 127 and the image does not have the correct colors;
- the black area should be black – in the RGB color space, this means that the average values of R, G and B should be close to zero. If the brightness is too high, it is observed that the black region becomes blue, resulting in a degradation of the image;
- the histogram of intensities should be centered around 127 and should span all intensity values. Dividing the histogram into regions, the left regions represent dark colors, while the right regions represent light colors. An underexposed image will be leaning to the left, while an overexposed image will be leaning to the right in the histogram.

Statistical measures can be extracted from digital images to quantify the image quality [20, 21]. A number of typical measures used in the literature can be computed from the image gray level histogram, namely:

- Mean:

$$\mu = \sum_{i=0}^{N-1} iP(i); \tag{3.1}$$

- Entropy:

$$E = -\sum_{i=0}^{N-1} P(i)log(P(i)); \tag{3.2}$$

- Absolute Central Moment (ACM):

$$ACM = \sum_{i=0}^{N-1} |i - \mu| P(i); \tag{3.3}$$

- Mean Sample Value (MSV):

$$MSV = \frac{\sum_{j=0}^{4} (j+1) x_j}{\sum_{j=0}^{4} x_j}; \tag{3.4}$$

where N is the number of possible gray values in the histogram (typically, 256), $P(i)$ is the probability of each gray value, $x(j)$ is the sum of the gray values in region j of the histogram (in the proposed approach we divided the histogram into five regions). The image is correct when $\mu \approx 127$, $E \approx 8$, $ACM \approx 50$ and $MSV \approx 2.5$. These measures allow to analyze the performance of the automated calibration algorithm. Moreover, the information of MSV will also be used to calibrate the exposure and the gain of the camera.

The algorithm configures the most important parameters of the camera: exposure, gain, white-balance and brightness. For the Unibrain Fire-i cameras, the dynamic range of these parameters are:

- exposure: 0-511;
- gain: 0-255;
- white-balance:0-255, both for the red and the blue channels;
- brightness: 128-383.

For the Point Grey Flea 2 camera, the dynamic range of the parameters are:

- exposure: 1-1023;
- gain: 0-683;
- white-balance: 1-1023, both for the red and the blue channels;
- brightness: 0-255.

For each one of these parameters, a PI controller was implemented. PI controllers are used instead of proportional controllers as they result in better control having no stationary error. The constants of the controller have been obtained experimentally for both cameras, guaranteeing the stability of the system and an acceptable time to reach the desired reference [22].

The algorithm configures one parameter at a time, iterating between them when the convergence of the parameter under analysis has been attained. The algorithm stops when all the parameters have converged. The algorithm is outlined in Fig. 3.5.

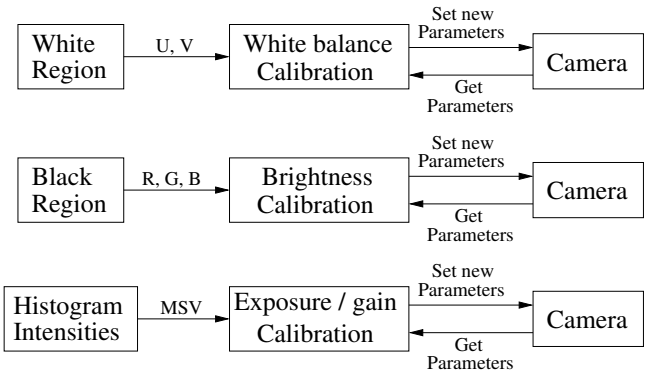


Figure 3.5: Overview of the automated calibration procedure. The algorithm executes one module at a time, changing between them when the convergence of the parameter under analysis has been attained.

To measure the performance of this calibration algorithm, tests have been conducted using the camera with different initial configurations. In Fig. 3.6, the experimental results are presented both when the algorithm starts with the parameters of the camera set to zero and set to the maximum value. As it can seen, the configuration obtained after using the proposed algorithm is approximately the same independently of the initial configuration of the camera. Moreover, the algorithm converges fast, between 60 and 70 frames to converge.

In Fig. 3.9 is presented an image acquired with the camera in auto mode. The results obtained using the camera with the parameters in auto mode are overexposed and the white balance is not correctly configured. This is due to the fact that the camera analyzes the entire image and, as we can see in Fig. 3.9, there are large black regions corresponding to the robot itself. The implemented algorithm uses a mask to select the region of interest to calibrate the camera using only the valid pixels. Moreover, and due to the changes in the environment around the robot as it moves, leaving the camera in auto mode leads to undesirable changes in the parameters of the camera, causing problems to the correct color classification.

Table 3.1 presents the value of the statistical measures described to evaluate the quality of digital images, regarding the experimental results presented in Fig. 3.6. These results confirms that the camera is correctly configured after applying the automated calibration procedure, since the results obtained are near the optimal. Moreover, independently of the initial configuration, we obtain images with the same characteristics.

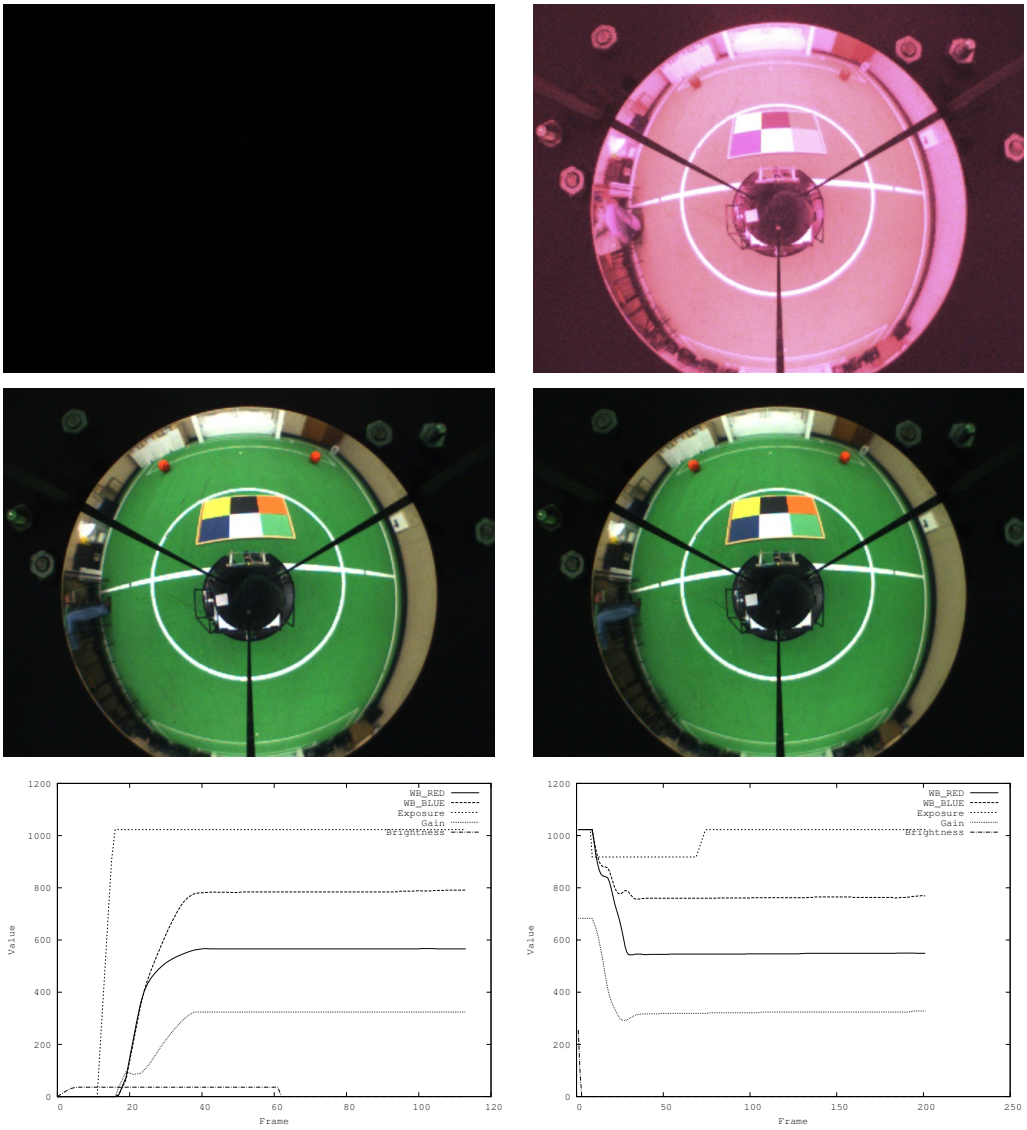


Figure 3.6: Some experiments using the automated calibration procedure. On the left, results obtained starting with all the parameters of the camera set to zero. On the right, results obtained with all the parameters set to the maximum value. On the top, the initial image acquired. In the middle, the image obtained after applying the automated calibration procedure. The last row contains the graphics showing the evolution of the parameters along the time.

According to the experimental results presented in Table 3.1, we conclude that the MSV measure is the best one in classifying the quality of an image. Although important, the other measures cannot distinguish between two images that have close characteristics.

The good results of the automated calibration procedure can also be confirmed in the his-

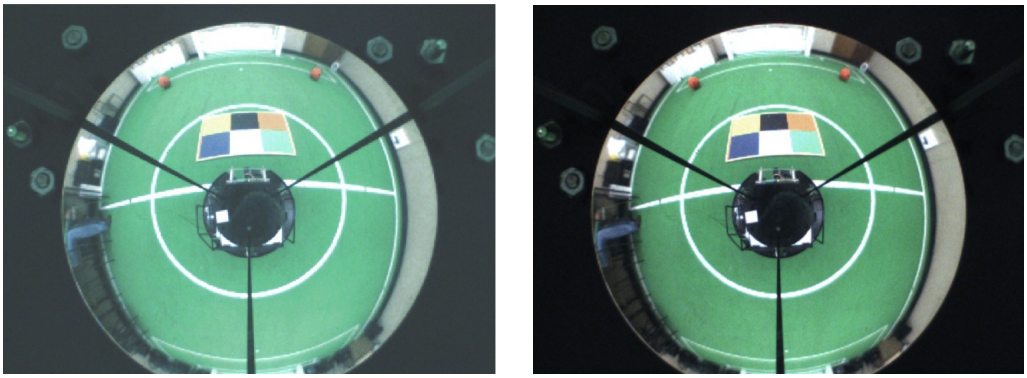


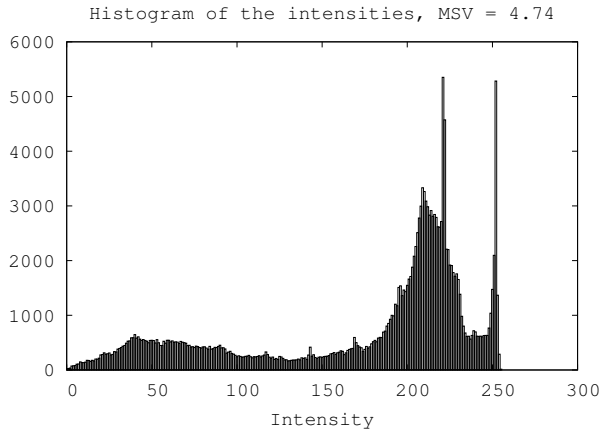
Figure 3.7: On the left, an example of an image acquired with the camera parameters in auto mode. On the right, an image acquired after applying the automated calibration algorithm.

Experiment	—	ACM	Average	Entropy	MSV
Parameters	Initial	111.00	16.00	0.00	1.00
set to zero	Final	39.18	101.95	6.88	2.56
Parameters.	Initial	92.29	219.03	2.35	4.74
set to maximum	Final	42.19	98.59	6.85	2.47
Camera	Initial	68.22	173.73	6.87	3.88
Auto Mode	Final	40.00	101.14	6.85	2.54

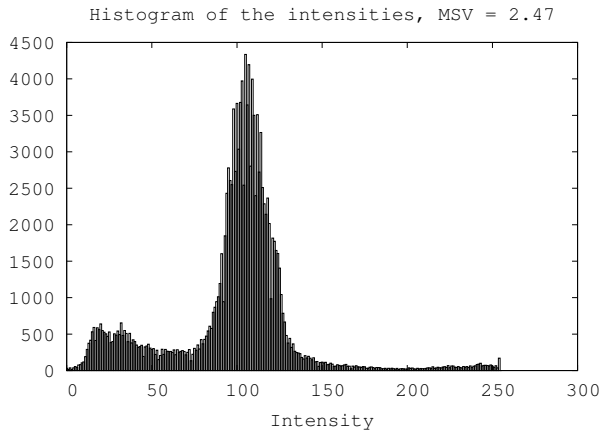
Table 3.1: Statistical measures obtained for the images presented in Figs. 3.6 and 3.7. The initial values refer to the images obtained with the camera before applying the proposed automated calibration procedure. The final values refer to the images acquired with the cameras configured with the proposed algorithm.

tograms presented in Fig. 3.8. The histogram of the image obtained after applying the proposed automated calibration procedure (Fig. 3.8b) is centered near the intensity 127, which is a desirable property, as visually confirmed in Fig. 3.6. The histogram of the image acquired using the camera in auto mode (Fig. 3.8a) shows that the image is overexposed, leading to the majority of the pixels to have bright colors.

This algorithm have also been tested outdoor, under natural light. Figure 3.9 shows that the algorithm works well even with different light conditions. It confirms that the algorithm could be used in non-controlled lighting conditions and under different environments.



a)



b)

Figure 3.8: The histogram of the intensities of the two images presented in Fig. 3.6. a) shows the histogram of the image obtained with the camera in auto mode. b) shows the histogram of the image obtained after applying the automated calibration procedure.

3.2.1 Run-time auto-calibration

Sometimes it is necessary to adjust the camera parameters during the game because the illumination along the field isn't constant and the nature of the light and its intensity changes during the game. That usually makes the colors become out of the range where they were supposed to be. Consequently the robot can't identify where the ball is or identify its own location on the field. In order to overcome this problem, it is crucial to keep a process running able to adjust the camera parameters during the game, and insuring that the image colors can be kept constant independently of the light. In this section, it will be discussed an implementation of an algorithm to perform this task. The algorithm must be fast because

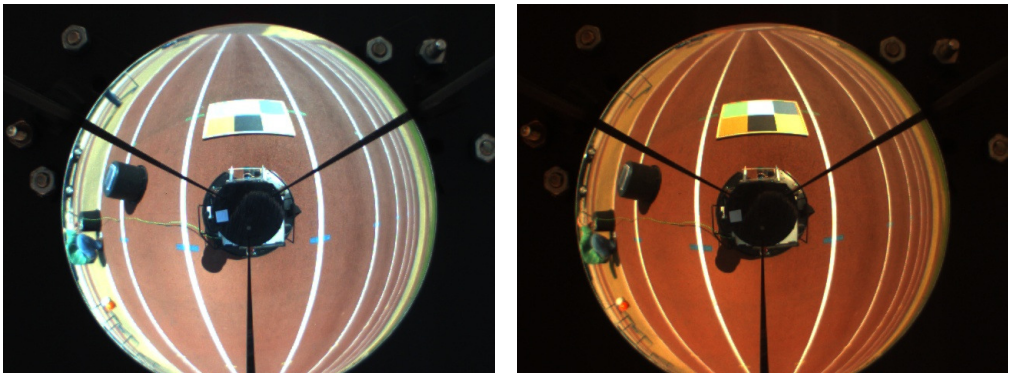


Figure 3.9: On the left, an image acquired outdoor using the camera in auto mode. As it is possible to observe, the colors are washed out. That happens because the camera's auto-exposure algorithm tries to compensate the black around the mirror. On the right, the same image with the camera calibrated using the implemented algorithm. As it is possible to observe, the colors and their contours are much more defined.

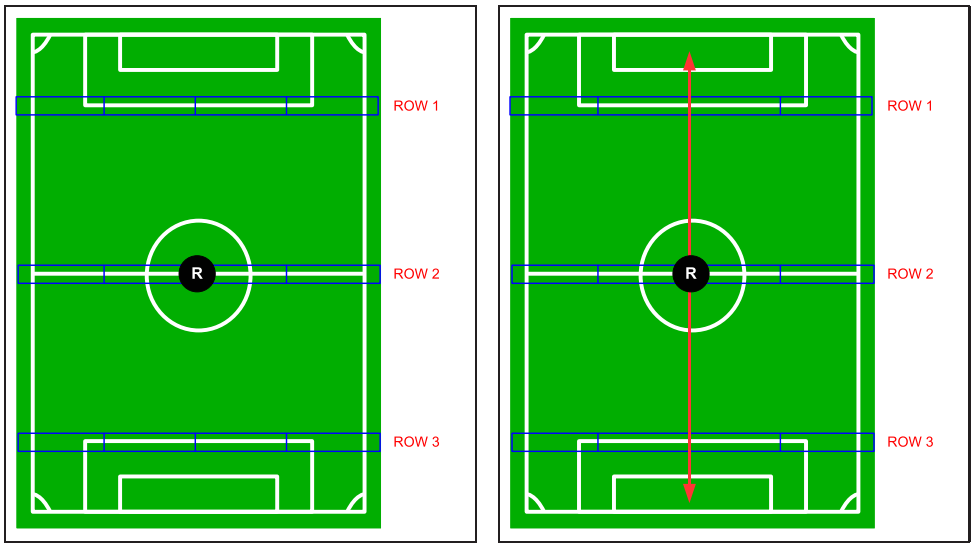
it could affect the other processes running in the computer of the robots, for example the acquisition and image processing program [23] and the agent software.

The developed algorithm is based on the assumptions described in the previous section but only adjusts the gain and exposure. The only measure made for each frame is the MSV (Equation 3.5). As mentioned before, it allows to know if the image is underexposed or overexposed. This measure is fast to be calculated, in particular when the image is acquired on the YUV color space, due to the fact that the luminance of each pixel is acquired directly from the camera without any further arithmetic operation. This algorithm is included in the image acquisition and processing application, which is the application running on the robot's computer responsible for acquiring, process and analyze the relevant information of each frame [23], and takes about 2ms per frame to execute. After the frame is acquired, the algorithm is called to calculate the MSV of the frame, then the gain and the exposure are adjusted in such way that the MSV of the next frame is near 2.5.

Tests have been performed on CAMBADA field, where it is possible to switch on and off three rows of fluorescent lamps. In the first test, the robot is placed in the middle of the field and the test starts with all the lamps on (see Fig. 3.10 a)). Then, one by one, all the rows of lamps are switched off. After that, the second row is switched on, followed by the first row and then the third row.

An example of an image acquired in the experiment described above is presented in Fig. 3.11.

The experimental results are presented in the Fig. 3.12, where the graphics present the evolu-



a) b)

Figure 3.10: In a) the representation of CAMBADA's field, for the first run-time experiment. The robot position is represented as a black circle, and the rows with the lamps are represented with the blue color, showing the arrangement of the lamps on CAMBADA's field. The test has started with all rows of lamps on. Then, from the row one to the row three, the rows lamps were shut down one by one. After that, the second row is switched on, followed by the first row and then by the third row. On the right, the red arrows represents the movement along the field performed by the robot during the second run-time test, starting with the rows one and three on. Then, in the middle of the test, the first and the third rows are switched off, and the second row is switched on at the same time.

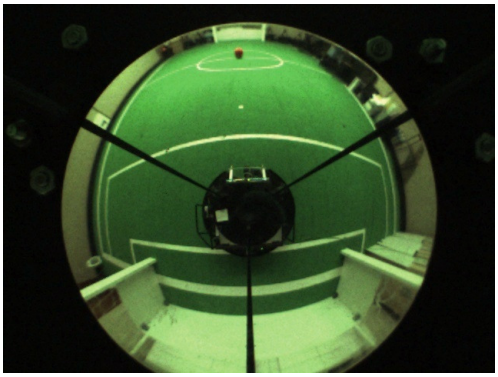


Figure 3.11: An image of the CAMBADA's field with only the second row of lamps on, causing a non-uniform light over the field. As it is possible to observe, it causes a great variation of the green along the field, which is impossible to calibrate.

tion of the exposure and gain parameters of the camera and the MSV value. As can be seen, the algorithm tries to adjust the camera parameters, in such way that the MSV is always 2.5. When both parameters, gain and exposure, hits the maximum, there is no way to maintain the MSV at 2.5 and the residual error is accumulated by the integral controller. The consequence is that the algorithm doesn't react immediately when the MSV decreases. In a real situation, this doesn't cause any problem because the illumination changes slowly along the time.

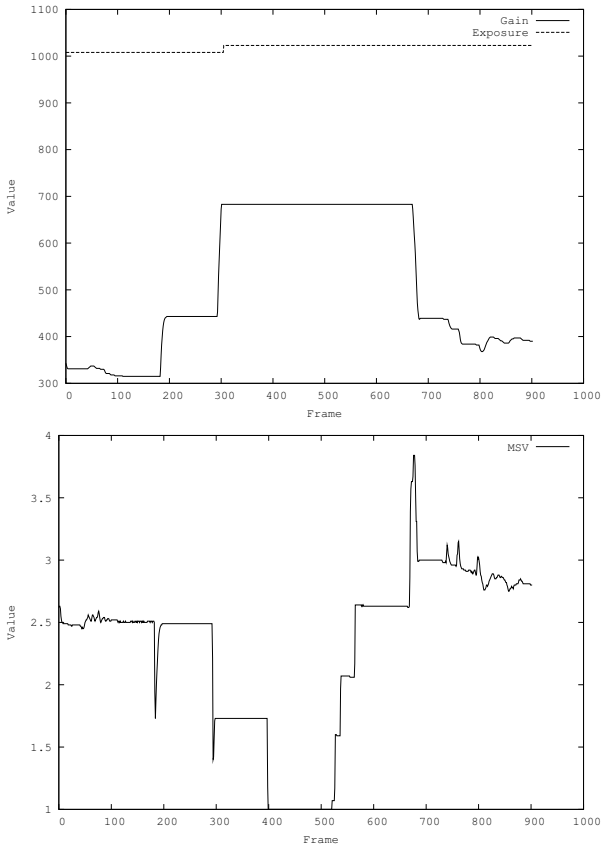


Figure 3.12: Graphics of first run-time test. It shows that the algorithm adapts quickly to small changes in light conditions maintaining the MSV value at 2.5.

The second experiment performed starts with the robot placed in the middle of the field. Then it moves in the field making the trajectory marked in the Fig. 3.10 b) with the red arrow. In the beginning of the test, only the first and third rows of lamps are switched on. Then, in the middle of the test, the first and the third rows are switched off, and the second row is switched on at the same time.

The experimental results are presented in Fig. 3.13, where the graphics present the evolution of the exposure and gain parameters of the camera and the MSV value. As can be observed, the algorithm adapts to gradual changes in light. Moreover, the test has shown that the robot

can always classify the relevant colors of the acquired image and always knows where it is placed on the field, meaning that the objects of interest, such as the white lines, are correctly detected.

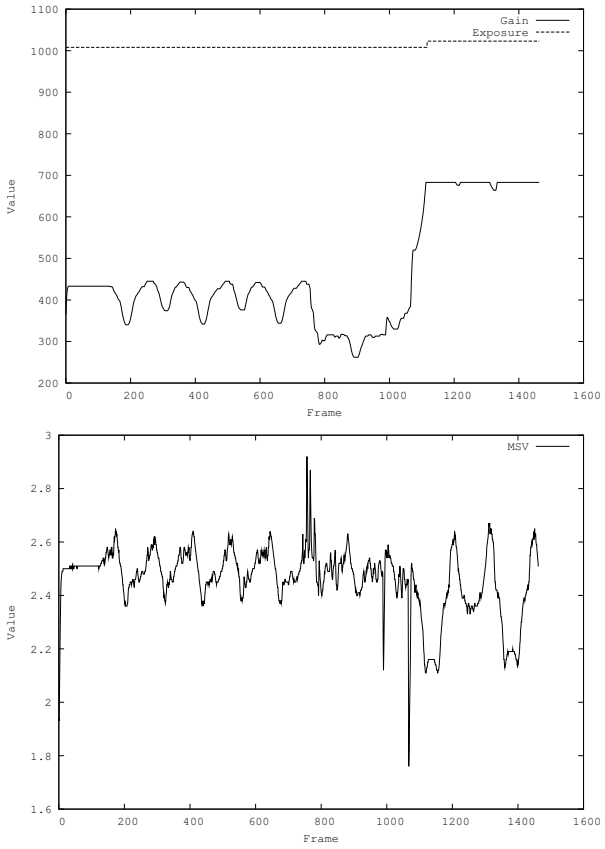


Figure 3.13: Graphics of the second run-time test. It shows that the algorithm adapts quickly to different light conditions maintaining the MSV value at 2.5.

Regarding the experimental results obtained, it is possible to conclude that the adjustment of the gain and exposure parameters contribute significantly to the color classification. It makes the robot to be able to always classify the colors, even under light variations along the field. Consequently, the robot always knows where it is placed on the field and can make the right decisions and perform the right behaviors. The effectiveness of the proposed algorithm was confirmed on ROBOTICA 2008 where this algorithm was used and the team CAMBADA has won the first place.

Chapter 4

Color calibration

In the RoboCup domain image analysis is simplified, since objects are color coded. Black robots play with an orange ball on a green field that has white lines. Thus, the color of a pixel is a strong hint for object detection. Due to this fact, the correct object detection may have a significant dependence on the correct color classification.

The calibration of the color range associated to each color class is another process that has to be performed before each game because the colors of an image acquired by a camera are dependent of the scene and also of the camera calibration. These two procedures have to match completely in order to allow the robot to recognize the relevant objects existing in the environment around it.

Until now, the color calibration procedure has been made offline, which means it was necessary to previously acquire a video sequence. Then, with an offline application, the user selects some pixels and according to the their color value a particular color class is created. This procedure has to be repeated for each relevant color. Usually, the color calibration is performed in HSV (Hue, Saturation and Value) color space due to its special characteristics (see Fig. 4.1) [14].

The main drawback of this procedure is the fact that it has to be performed offline, meaning that when the camera calibration or the colors of the scene change, it is necessary to acquire a new video and then process it again. These steps usually requires several minutes to adjust the information for each color. Another problem is that, using an offline procedure, there is no way to test the color calibration result in real-time and in all positions of the field.

In this chapter three developed methods will be presented allowing online color calibration. These methods make possible to calibrate the colors directly on the robot.

The first implemented method is based on the statistical information about a specific selected

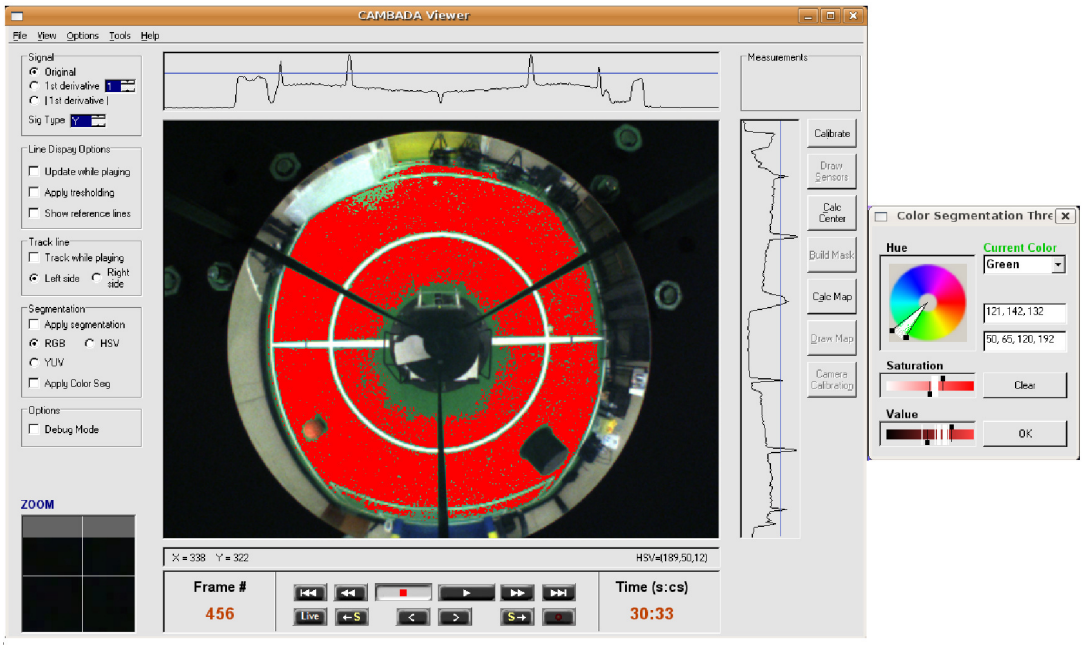


Figure 4.1: A screenshot of CambadaViewer application, that is the application used to calibrate the colors in offline mode [24].

area for each color class. For each area, it is calculated the maximum and minimum values in HSV color space. These values are used to select the upper and lower boundaries of the color class. If the color range wasn't correctly calibrated, a manual adjustment has to be made for maximum and minimum values of hue, saturation and value using a sliders interface. This procedure can present some problems particularly when the selected area hasn't an homogeneous color or, sometimes, when there is a pixel with a quite different color from the desired one, leading to colors wrongly calibrated or somewhat less accurate. Furthermore the use of sliders to select the color range is not much intuitive.

In order to improve the method described above, the second method is an evolution from the first one. It was developed an interface that uses the histogram of the three color components (hue, saturation and value) to select the desired color range for each color class. For each color, a set of pixels are selected corresponding to the color class that is under calibration. The value of each pixel, in HSV color space, is marked on the histogram of hue, saturation and value. Then, it is selected the range of hue for the selected color and the saturation and value histograms are updated with the image pixels included in this range of hue. This procedure is easier, faster and more accurate than the method using sliders.

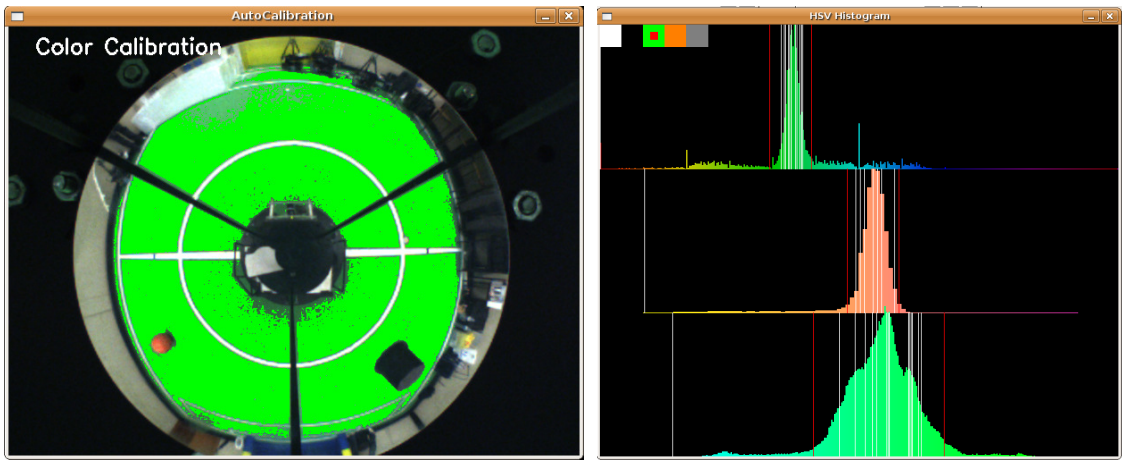


Figure 4.2: On the left, an image acquired on the CAMBADA field with the green color segmented. On the right, the corresponding histogram. From top to bottom: hue, saturation and value of the image.

4.1 Automated algorithm for color calibration

A third method was implemented using HSV histograms and a canny edge detector [25, 26, 27, 28]. The canny edge detector finds the edges in areas with strong intensity contrasts. This is the most used edge detector in image processing. Applying the canny edge detector to the image acquired, all the transitions between colors with high contrast are marked, as shown in Fig. 4.3.

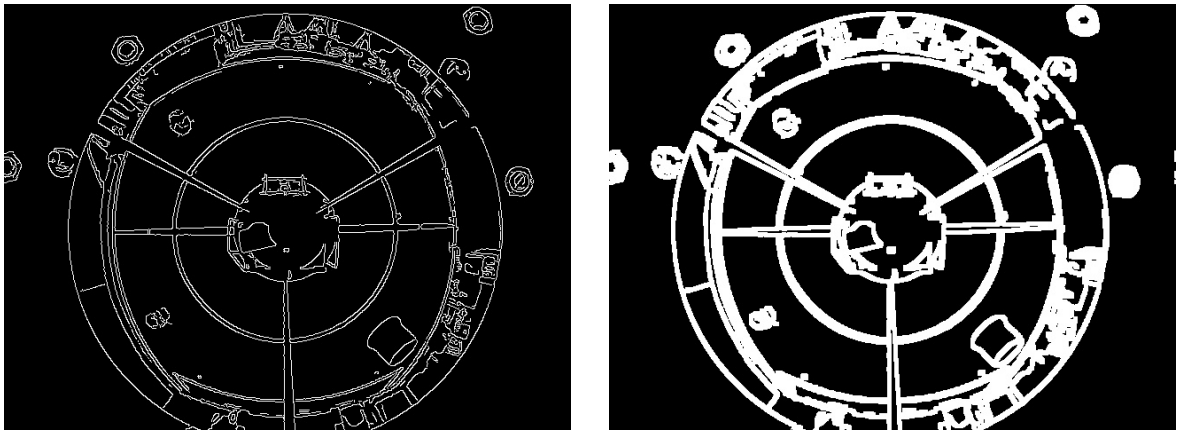


Figure 4.3: On the left, an image acquired on the CAMBADA's field, to which the canny edge detector was applied. On the right, the same image with strong edges after considering as edge the neighboring pixels of the original edges obtained by the canny edge detector.

As it can be seen in the left side of the Fig. 4.3, some areas are not correctly delimited. To

	Minimum	Maximum
Hue	79	177
Saturation	26	64
Value	84	207

Table 4.1: Color ranges obtained for the green color class using the described algorithm according to the Fig. 4.4.

.

correct this effect, it is considered as edge the neighboring pixels of the original edges obtained by the canny edge detector. With this transformed image, where areas with the same color are well delimited, it is possible to calculate the maximum and minimum boundaries in HSV color space for color classification. First, it is necessary to find at least one pixel of the intended color in each one of the delimited areas. For that, using the assumption that most of the image pixels are green, the hue, saturation and value of the color with the higher level of ocurrence in the HSV histogram are determined. Using the HSV histogram, it is possible to know which are the values, in HSV color space, of the pixel that appears more often. Then, with a threshold of 10, it is selected at least one pixel in each delimited area. Now, using a region growing algorithm, all pixels delimited are selected. With all green pixels selected, it is calculated the maximum and minimum values in HSV color space for the color under analysis.

The right side of the Fig. 4.4, shows the field image with the green color segmented. The results obtained with this algorithm are presented on Table 4.1. As it can be seen, all the green pixels are segmented confirming that this algorithm works well.

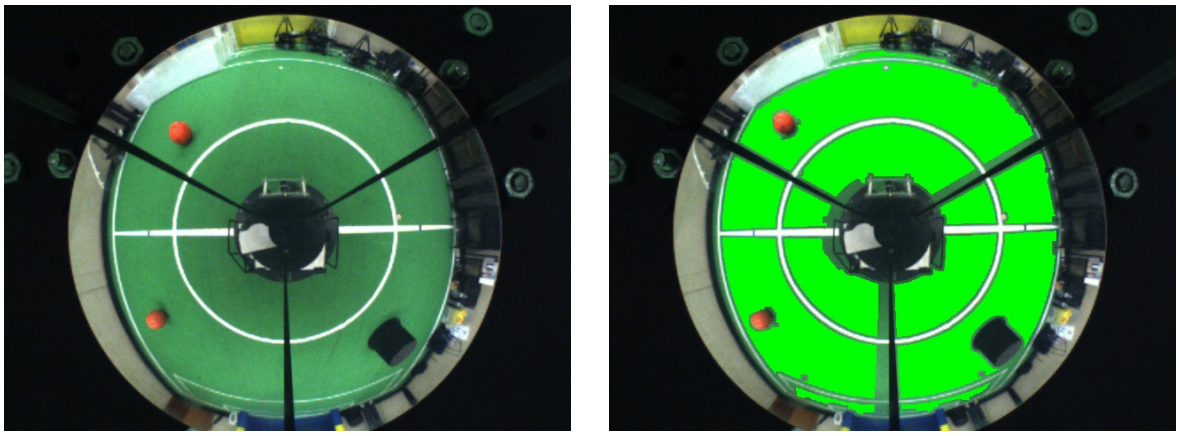


Figure 4.4: On the left, an image acquired on the CAMBADA's field. On the right, the same image with the green color segmented.

The experimental results obtained with the described algorithm are promising. However, the current implementation has a high computational complexity (above 100ms to process a single image) which means it can not be used in real-time.

Chapter 5

Conclusion

There are many problems behind the use of cameras in robotic applications. One of them, and the most difficult to overcome, is the use of cameras in robotic applications under natural light. To achieve the first goal of RoboCup MSL competition, it is necessary to overcome this problem.

In Chapter 1, it was discussed the most common problems behind the use of cameras in RoboCup MSL competition. It was also discussed how CAMBADA team and the other RoboCup MSL teams solve this problems.

In Chapter 2, it was introduced some notions about camera parameters and the RGB, YUV and HSV color spaces, to better understand the algorithms discussed in the following chapters.

In Chapter 3, it was discussed the two implemented algorithms to autocalibrate the cameras at the beginning of each game. This doesn't solve the whole problem but minimizes the effects of the light variations during the tournament. By comparing the two developed algorithms, it is easy to conclude that the second algorithm, using PI controllers and the MSV measure, has better results. That happens mainly because it uses the PI controllers to approximate the camera parameters to the ideal parameters and the MSV measure that evaluates the luminance of the image. Comparing to the second algorithm, the first developed algorithm doesn't work so well. One of the problems is that it takes much more time to converge. Another problem is that it only uses the information of two selected areas, and not the whole image.

It was also presented a run-time algorithm that can be used with the image acquisition and processing application. It minimizes the effects of lighting variations along the field and during the game. That is a typical problem that CAMBADA team has faced, until now.

In Chapter 4, it was presented some color calibration algorithms and it was also discussed

the methods used by CAMBADA team until now. The first developed algorithm, using the analysis of a selected area for each color, isn't the most accurate due to the fact that if one pixel of the selected area is quite different of the selected color, that will cause an incorrect color calibration. It is possible to adjust the color ranges using an interface with sliders.

The second method, using HSV histograms, makes the process of manual color calibration easier, because selecting some pixels on the image allows to visualize their position on HSV histogram. Selecting the maximum and minimum boundaries of hue, saturation and value histograms are updated only with the image pixels included on the selected range of hue.

The last developed method use an edge detection and a region growing algorithm. The experimental results obtained with the described algorithm are promising. However, the current implementation has a high computational complexity and can not be used in real-time. In the future, with some optimization this drawback could be solved. This method with the camera calibration running on robot's computer, will probably allow the robot to have its vision system always optimized for the best performance on color classification and object detection.

Now with the proposed algorithm for camera parameters and color calibration methods, it is possible to calibrate the vision system of the robots without the need for either a previous video acquisition or for the presence of an expert person. This can now be made directly on the robot, and visualized in real-time if the colors are correctly calibrated.

This thesis has contributed significantly to the good results obtained by the CAMBADA team in the Portuguese Robotics Open, ROBOTICA 2008 [29]. The work presented in this thesis was also used in other robotic applications, as is the case of the Rota2008 and Ratozinger teams, of University of Aveiro , in the Autonomous driving contest. This work has contributed to their color and cameras calibration.

The results obtained prove that the algorithms presented in this thesis will contribute to the main goals of RoboCup league.

As future work, the following lines can be pointed out:

- Improvement of the color calibration performance in such way that it can be made automatically in offline and also in run-time mode.
- Improvement of the white-balancing algorithm, in such way that it doesn't need a white reference or, if needed, the reference must be searched in the image.
- Development of an algorithm to automatically calibrate the camera saturation.
- Implementation of an algorithm integrated into the software of the robot to calibrate colors and camera parameters in run-time.

Appendix A

AutoCalibration manual

A.1 Introduction

This manual intends to explain how to use the calibration tool designed for CAMBADA team, to calibrate their firewire cameras. This application is used to adjust automatically the parameters of the camera. To use it, the application only needs to know where the black and white references are. These are the necessary references used by the calibration algorithm. The camera's parameters are adjusted by the analyses of the selected black and white areas and the whole valid image, until the "the white is really white and the black is really black".

The algorithm could be explained in a simple way such as:

- The white-balance (the gain of blue and red channels respectively), is adjusted until the average values of U and V (in YUV color space), of the selected white zone are approximately 127.

- The gain and exposure are adjusted in such way that the luminance histogram is centered around 127.

- The brightness is adjusted from the zone selected as black. It is adjusted until the average value of each RGB channel is around 2.0.

This application also allows the user to calibrate the color range, using the information obtained by the HSV histograms.

A.2 How to use the application

The use of this application is simple. To start the application the following command line format is used: *./AutoCalibration -cf # {options}* The only mandatory parameter is the configuration file (option *-cf* r?.conf), where the information of the initial parameters of the camera, the color range for each color, the mask that selects the valid pixels, and the distance map are stored. This application was developed only for the cameras used by CAMBADA team, which are a Point Grey Flea 2 and a Unibrain Fire-i BCL. To use one of these cameras, the parameter *-cam* must be initialized (option *-cam* # { Unibr: Unibrain Fire-i BCL, Point: Point Grey Flea 2}). The parameter fps (frames-per-second) also has to be configured (option *-fps* # {3, 7, 15, 30}).

When the application starts, the image acquired by the camera is diplayed in the computer screen. To calibrate the camera parameters, first it is necessary to select the white and black areas. For this, press the **C** key and in the screen is displayed the number of each color. Then, press the key number of the desired color. When the color is selected, with the mouse select the area for that color. The key **S** allows to see the selected zones for each color in the screen.

To start the calibration algorithm press the **A** key. In the screen will be displayed the message "AutoCalibrating ..." and the current values of gain, exposure, brightness and white-balance. After a few seconds, when the parameters had stabilyzed, you should press again the **A** key to stop the calibration algorithm. To store the correct parameters into the configuration file press **U**. It is possible to use the camera's autocalibration algorithm at any time, pressing the **K** key. To desactivate the camera's algorithm press again the **K** key.

To calibrate the colors, first press the **E** key. It will be displayed in the screen the HSV histogram. To select the desired color to be calibrated, press in the "HSV - Histogram Window", with the mouse's left key, the respective square color. Then press the key **I**. It will be displayed, in a new window, an enlarged image of the mouse's pointed area. To know the values, in HSV color space, of the selected pixel, press the mouse's left key. The values of the current pixel will be displayed on the image and in the HSV-Histogram window, the current pixel position will be displayed with white lines, in the HSV-Histogram. To display the segmented image, press the **Y** key. If a certain pixel is not segmented, for the respective color, select it with the mouse. On the HSV-Histogram window, the red lines are the upper and lower boundaries of the selected color. If a certain white line is out of the range, to move the lower boundarie, in the HSV-Histogram window, press on the white line with the mouse's left button. To move the upper boundarie, in the HSV-Histogram window, select the white line with the mouse's right button. To clear the white lines on the HSV-Histogram, press with

the mouse's left button on the gray square.

Now will be explained the function of each key and the meaning of the arguments that is possible to pass to the application:

Application:	
-h	Help
-cf	Configuration file (Obligatory)
Camera:	
-mode # 4	4:RGB, 1:YUV 411, 2:YUV 422
-cam # {Point}	Point: Point Gray Flea2, Unibr: Unibrain Fire-i BCL
-fps # {15}	3:3.75, 7:7.5, 15, 30
-wb #:# {-1:-1}	
-sht # {0}	
-exposure # -1	
-gain # {-1}	
-brightness # {-1}	
-gamma # {-1}	
-saturation # {-1}	
-sharpness # {-1}	

During the application some keys have specific functionalities, which are:

Key	Description
H	Visualize on the screen the function of each key
C	Allows to select a specific color area
A	Activate/deactivate the calibration algorithm
S	Shows the selected color areas
K	Activate/deactivate the camera's calibration algorithm
U	Update the configuration file
P	Pause
I	Allows to know the HSV information of a certain pixel
	Shows the enlarged area around the mouse pointer
Y	Segment the image
E	Displays the HSV histogram
R	Displays the RGB histogram
G	Displays the Luminance Histogram
Q	Quit

Appendix B

LibAutoCalib

- void Rgb2Yuv (double *RGB, color_yuv& yuvPix) - Convert a RGB pixel to YUV
- void Rgb2Yuv2(double *RGB, color_yuv& yuvPix) - Convert a RGB pixel to YUV
- void ConvertYuv2Rgb(unsigned char buffin[], unsigned char buffout[], int mode) - Convert a YUV frame to RGB
- void AverageAreaRGB(unsigned char* imgBuf, int y1, int y2, int x1, int x2, Stats* RGB) - Calculates the average value in RGB mode of the selected area for each channel
- void StdAreaRGB(unsigned char* imgBuf, int y1, int y2, int x1, int x2, Stats* statsvalues) - Calculates the standard deviation value in RGB mode of the selected area for each channel;
- void MaxMinAreaRGB(unsigned char* imgBuf, int y1, int y2, int x1, int x2, Stats* statsvalues) - Calculates the maximum and minimum values in RGB mode of the selected area for each channel
- void AverageAreaHSV(unsigned char* imgBuf, int y1, int y2, int x1, int x2, Stats* HSV) - Calculates the average value in HSV mode of the selected area for each channel
- void StdAreaHSV(unsigned char* imgBuf, int y1, int y2, int x1, int x2, Stats* statsvalues) - Calculates the standard deviation value in RGB mode of the selected area for each channel
- void MaxMinAreaHSV(unsigned char* imgBuf, int y1, int y2, int x1, int x2, Stats* statsvalues) - Calculates the maximum and minimum values in RGB mode of the selected area for each channel

- void acmCalc(unsigned char* buf, unsigned char* mask, ImageStats* stats) - Calculates image statistics of the valid pixels
- void SegmentateImage (unsigned char* SegImage, unsigned char* mask, ColorRange *cr) - Segment the image
- void CalcHistogram(unsigned char* imgBuf, unsigned char* mask, int* histogram, ImageStats* stats) - Calculates the histogram
- ImageStats GetHistParam (const ImageHolder& image, unsigned char* mask) - Calculates histogram parameters
- void CalibrateGainExposure(ImageStats stats, CameraSettings* cs) - Calculates the Gain and Exposure values to set to the camera according to theinformation passed by stats.
- Class ImageAnalyzer
  - ImageAnalyzer() - Default constructor
  - ImageAnalyzer(enum iMode Mode, unsigned Rows, unsigned Cols, unsigned char *ImgBuf, unsigned char *Mask) - Contructor
  - ĩmageAnalyzer() - Destructor
  - void CalcImgStats() - Calculates image satistics
  - void DrawHistogram() - Draw histogram
  - void SetColourCoord(unsigned color, int x1, int y1, int x2, int y2) - Set color coordinates
  - void CalcImgAreaStats() - Calculates area satistics
  - void AverageArea(unsigned color) - Calculates the average value of the selected area for each channel
  - void MaxMinArea(unsigned color) - Calculates the average value of the selected area for each channel
  - void ZoomPoint(CvPoint ptCenter) - Zooms the selected point
  - void SegmentateImg (ColorRange *cr) - Segmentates the image
  - void DrawRgbHistogram() - Draws the RGB histogram
  - void DrawHsvHistogram() - Draws the HSV histogram

Bibliography

[1] CAMBADA official homepage, April 2008. http://www.ieeta.pt/atri/cambada.

[2] RoboCup official homepage, April 2008. http://www.robocup.org.

[3] 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.

[4] A. J. R. Neves, D. A. Martins, and A. J. Pinho. A hybrid vision system for soccer robots using radial search lines. In *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBÓTICA'2008*, pages 51–55, Aveiro, Portugal, april 2008.

[5] G. Mayer, H. Utz, and G. Kraetzschmar. Playing robot soccer under natural light: A case study. In *Proc. of the RoboCup 2003*, volume 3020 of *Lecture Notes on Artificial Inteligence*. Springer, 2003.

[6] R. Hafner, S. Lange, M. Lauer, and M. Riedmiller. Brainstormers tribots team description. Technical report, Institute of Computer Science, Institute of Cognitive Science, 2008. University of Osnabru, Germany.

[7] H. Zhang, H. Lu, X. Wang, F. Sun, X. Ji, D. Hai, F. Liu, L. Cui, and Z. Zheng. Nubot team description paper 2008. Technical report, College of Mechatronics and Automation, 2008. National University of Defense Technology, China.

[8] J. J. M. Lunenburg and G. v.d. Ven. Tech united team description. Technical report, Control Systems Technology Group, 2008. Eindhoven University of Technology, Netherlands.

[9] B. Bouchard, D. Lapensée, M. Lauzon, S. Pelletier-Thibault, Jean-Christophe Roy, and G. Scott. Robofoot Épm team description paper 2008. Technical report, Mechatronics Laboratory, 2008. École Polytechnique de Montréal, Canada.

[10] S. Battiato and M. Mancuso. An introduction to the digital still camera technology. *ST Journal of System Research - Special Issue on Image Processing for Digital Still Camera*, 2(2), December 2001.

[11] P. M. R. Caleiro. Aplicação gráfica para configuração e monitorização de sistemas de visão robótica. Technical report, Universidade de Aveiro, 2006.

[12] Bayer filter, Wikipedia Article, April 2008. http://en.wikipedia.org/wiki/Bayer_filter.

[13] Keith Jack. *Video Demystified.* Elsevier, 4 edition, 2005.

[14] P. M. R. Caleiro, A. J. R. Neves, and A. J. Pinho. Color-spaces and color segmentation for real-time object recognition in robotic applications. *Revista do DETUA*, 4(8):940–945, June 2007.

[15] Using rgb2ind, matlab toolbox, April 2008. http://www.mathworks.com/access/helpdesk_r13/help/toolbox/images/colorcube.jpg V.

[16] YUV, wikipedia article, April 2008. http://en.wikipedia.org/wiki/Y%27UV.

[17] Chroma subsampling, Wikipedia Article, April 2008. http://en.wikipedia.org/wiki/Chroma_sub sampling.

[18] F. Ortiz, F. Torres, J. Angulo, and S. Puente. Comparative study of vectorial morphological operations in different color spaces. In *Proc. of the Int. Conf. on Intelligent robots and computer vision*, volume 4572, pages 259–268, San Diego, CA, oct 2001.

[19] HSL and HSV, Wikipedia Article, April 2008. http://en.wikipedia.org/wiki/HSL_and_HSV.

[20] M. V. Shirvaikar. An optimal measure for camera focus and exposure. In *Proc. of the IEEE Southeastern Symposium on System Theory*, Atlanta, USA, March 2004.

[21] N. Nourani-Vatani and J. Roberts. Automatic camera exposure control. In *Proc. of the 2007 Australasian Conference on Robotics and Automation*, Brisbane, Australia, December 2007.

[22] J. J. D'Azzo, C. H. Houpins, and S. N. Sheldon. *Linear Control System Analysis and Design with Matlab.* CRC Press, 2003.

[23] D. A. Martins. Image processing system for robotic applications. Master's thesis, Universidade de Aveiro, 2008.

[24] 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.

[25] R. Boyle and R. Thomas. *Computer Vision: A First Course*. Blackwell Scientific Publications, 1988.

[26] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), November 1986.

[27] E. Davies. *Machine Vision: Theory, Algorithms and Practicalities*. Academic Press, 1990.

[28] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[29] Robotica 2008 - festival nacional de robotica, May 2008. http://robotica.ua.pt/robotica2008.