The Color and the Shape

Automatic On-line Color Calibration for Autonomous Robots

Diplomarbeit
bei Prof. Dr. Raúl Rojas

Vorgelegt von
Ketill Gunnarsson
ketill@inf.fu-berlin.de

am Fachbereich Mathematik und Informatik der
Freien Universität Berlin,
Institut für Informatik,
Takustraße 9, 14195 Berlin
Berlin, den 1. November 2005

Summary

This thesis presents a method for automatic on-line color calibration of soccer-playing robots. It was developed to automate the color calibration process of the FU-Fighters Mid-Size robots, which play in the RoboCup Middle-Size League. Although the method was implemented on soccer-playing robots, it could also be used in other applications where colors need to be mapped to symbolic colors (calibrated).

The method requires geometrical models of the objects whose colors are to be calibrated. In the implementation presented here, a model of the field-lines in world coordinates, and one of the ball in image coordinates was used. This made calibration of 4 distinct symbolic colors possible: Field (green), ball (orange), yellow goal (yellow) and blue goal (blue). The method accomplishes this without making any specific assumption about the color of the field, ball, or goals except that they are of roughly homogeneous distinct colors, and that the field-lines are bright relative to the field. The calibration works by localizing the robot(without using color information), then growing homogeneously colored regions and matching their size and shape with those of the expected regions. If a region matches the expected one, its color is added to the respective symbolic color class. This method can be run in a background thread thus enabling the robot to quickly recalibrate in response to changes in illumination.

The method was tested in a real-world scenario during the RoboCup World Championship in Osaka 2005, where it was used extensively by the FU-Fighters Mid-Size team which attained the second place.

Contents

Chapter 1

Introduction

This work describes a method for automatic on-line color calibration used by the soccer-playing robots of the FU-Fighters Mid-Size team. The robots compete in the RoboCup Middle-Size League, and are equipped with an omni-directional vision system and use conventional sub-notebooks for processing (see [4] and Fig.1.1).

A short introduction to the RoboCup domain follows. After that the importance of color calibration is discussed as well as the advantages of automating such a calibration. Next, the automatic calibration method is described step-by-step and finally, experimental results are presented.

1.1 The RoboCup Initiative

Robotic soccer provides a highly dynamic environment where one of the main challenges is simply finding out what is going on on the field. It is different from other "standard-problems" of artificial intelligence where an agent often has complete knowledge over a mostly static world.

The RoboCup Federation (www.robocup.org) was founded in 1997 as an international organization consisting of around 150 universities and research institutes. The purpose of RoboCup is to proliferate research by using soccer-playing robots as a benchmark for new algorithms and methods developed in artificial intelligence and robotics.

Computers have been competing with man since the early days of their creation. Traditionally the competitions were biased in favor of the computers and the tests emphasized areas like number crunching (e.g. multiplying large numbers together) or searching for the best action in a static, completely known world (e.g. chess). The drawback of these problem is that

Figure 1.1: FU-Fighters Mid-Size Robots used in the RoboCup World Championship in Osaka 2005.

they do not promote skills which robots vitally lack in order to deal with everyday situations. Simply exploring a busy office-building without bumping into something or someone is an example of a task where computers are largely inferior to humans. In fact, one can say that as the environment becomes more dynamic, the harder it is for a computer to master the situation.

For these reasons soccer was proposed as the next computer-human competition, and the RoboCup Federation has set itself the ultimate goal of winning the human world champion of soccer by the year 2050(!). Since this goal is very ambitious and not achievable in the near future, the RoboCup community currently concentrates on solving many of the subproblems needed to realize this dream.

Playing soccer is a very challenging task for a robot because it presents a highly dynamic environment with many objects moving at high speeds (players, ball). It is actually so challenging that several simplifications are made to the game environment in RoboCup in order for the robots to be able to perform something which can be considered to be a game of soccer. These simplifications include limiting the competition to indoor environments with constant lighting, as well as strictly color-coding the field and the robots. The field is also a lot smaller than a normal soccer field. Furthermore,

the robots only compete internally since they are still perform abominable compared with any human player.

It is the intention of the RoboCup Federation to abandon these simplifications gradually and bring the game closer to an official outdoor game of soccer. In the process one hopes to solve problems common to many modern and future robotic tasks. This way, progress made in robotic soccer should expedite advance in other areas of robotics and AI.

A problem common to RoboCup as well as other areas of robotics is the problem of dynamic lighting and the mapping of color values to symbolic colors. This problem is the theme of this thesis. Other problems in RoboCup include multi-agent collaboration, strategy acquisition, real-time reasoning, and sensor-fusion to name a few.

An advantage of robotic soccer which is worth mentioning is that most people understand what the game is about (get the ball into the opponents goal!), which makes it easier for non-experts to get an idea about how fast (or slow) progress in AI and robotics is.

Teams in RoboCup compete in several different leagues allowing researchers to focus on different sub-problems of robotic soccer, from straightforward sensing and acting, to high-level inter-robot cooperation and planing. At the RoboCup World Championship held each year, teams compete in 5 different leagues:

- Small-Size League: Size up-to 18cm diameter, 5 robots in each team on a field of approx. 3x4m. The robots (which carry color-markers) and an orange golf-ball are tracked by fixed overhead camera(s) connected to an off-board computer. The computer processes all information and radios commands to each robot in the team.

- Mid-Size League: Size approx. 50-60cm diameter, 4-6 robots in each team on a field of approx. 8x12m. The robots do most processing on-board with their own camera and other sensors.

- Humanoid League: Robots in humanoid form of various sizes, performing challenges such as penalty-shooting and, recently, play 2 against 2. Processing is done on-board.

- 4-Legged League: Teams of 4 sony-aibo robot dogs play against each other. Processing is done on-board.

- Simulation League: Teams of 11 simulated software agents play against each other in a virtual world.

Figure 1.2: Many different robots compete in several leagues at the RoboCup World Championship held each year (source: [11], [12] and [13]).

1.2 The Importance of Color Calibration

One of the most challenging subproblem in RoboCup is a real-time vision system delivering information about the state of the game comparable to that available to a human player. Today, most such vision systems rely on color calibration, and their performance is directly affected by the quality of the calibration. Color calibration is the process of mapping color values (e.g. RGB or YUV values) to symbolic colors (orange, green, yellow, etc.). It enables the vision system to "look-up" the symbolic color of a pixel when processing the image. This, for example, makes it possible to quickly find the ball by simply looking for an orange blob in the image.

In the RoboCup Mid-Size League most teams go through tedious calibration procedures to obtain as accurate color calibration as possible. A typical approach is to manually define which parts of the color space correspond to a color class using some kind of a GUI tool. This involves capturing images from different positions on the field, defining the color-boundaries between color classes, or classifying individual color pixels into one of the color classes. This method is error-prone and time consuming. Furthermore, a classification obtained at one point can fail at another, if the lighting conditions are different. For this reason, all objects in the RoboCup-environment are strictly color-coded and the organizers try to provide lighting that is as steady, bright and uniform as possible.

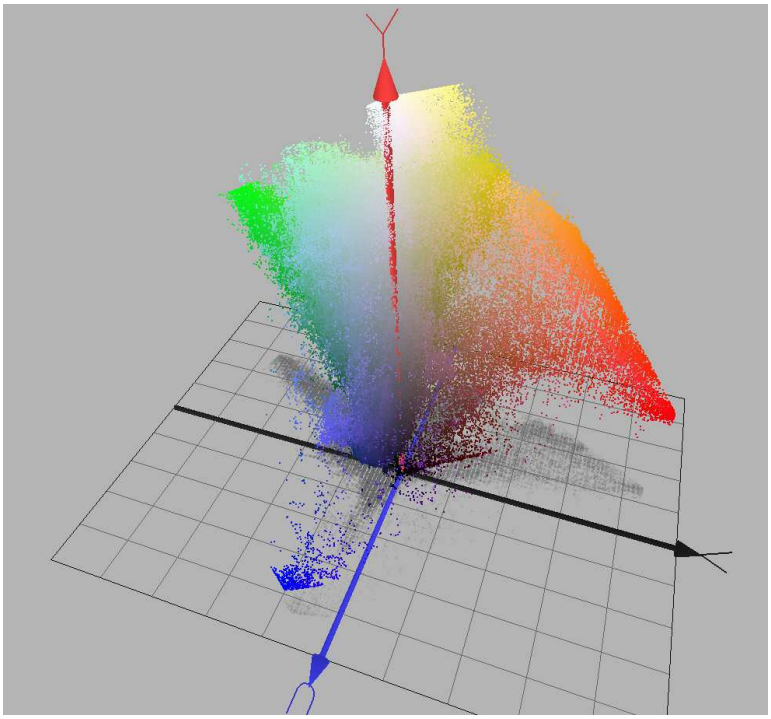

Figure 1.3: 3D color visualization of the colors appearing in the image seen in figure 3.3, captured from a Mid-Size robot's camera. Image obtained with "ColorSpace" (www.couleur.org)

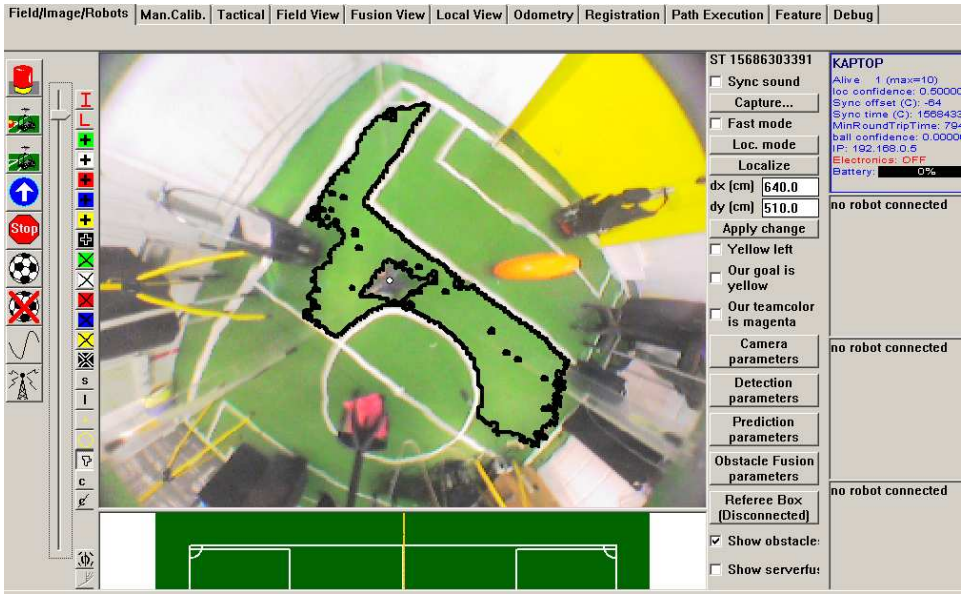

Figure 1.4: The manual color calibration tool of the FU-Fighters system. The operator clicks a point in the image where a color growing should be started with a manually chosen threshold. The pixels are compared to the starting point and added to the region until they exceed the given threshold. The operator then adds the region to the corresponding color class or chooses another region.

The method presented here remedies the problem by automatically classifying regions of homogeneous color into the following four color classes: field, ball, yellow goal, and blue goal. Regions that do not fit the criteria of any of the classes are not classified and can be considered obstacles. The white field-lines are detected without the use of color information, and can be identified as non-obstacles. The output of the method is essentially a separate list of color values for each color class. These lists grow over time, as more and more colors are classified. In the FU-Fighters system, these lists are stored as a look-up table using the full 24-bit YUV color depth.

The method can run on-line during a game to compensate for changes in lighting, and is able to calibrate a whole image from scratch and error-free in 1-2 seconds. It is robust against false classification even with robots, humans and other objects cluttering the field.

The calibration method previously used in the FU-Fighters system was

completely manual. The operator would get an image over the network from the robots camera using a debug tool running on a remote server (see Fig1.4). With it he would repeatedly choose a color region with a manually tuned threshold and add those regions' colors to the corresponding color class. Since light varies depending on where the robot is standing on the field, the operator needs to drive the robot to numerous different positions and calibrate colors at each one of them. In figures 1.5 and 1.6 the process is illustrated for one position.

Calibrating a robot in this manner could take up to 15 minutes. Since the robots cameras are all slightly different it is not possible to transfer the color look-up table of a calibrated robot to an uncalibrated one, therefore making individual calibration of each robot necessary.

Manual color calibration is one of the main contributor towards long setup times in RoboCup. Reducing setup time is a major concern for most teams since they are often required to play many games with short intervals at different fields. Furthermore, long setup times make any robot less useful or even completely useless, regardless the task.

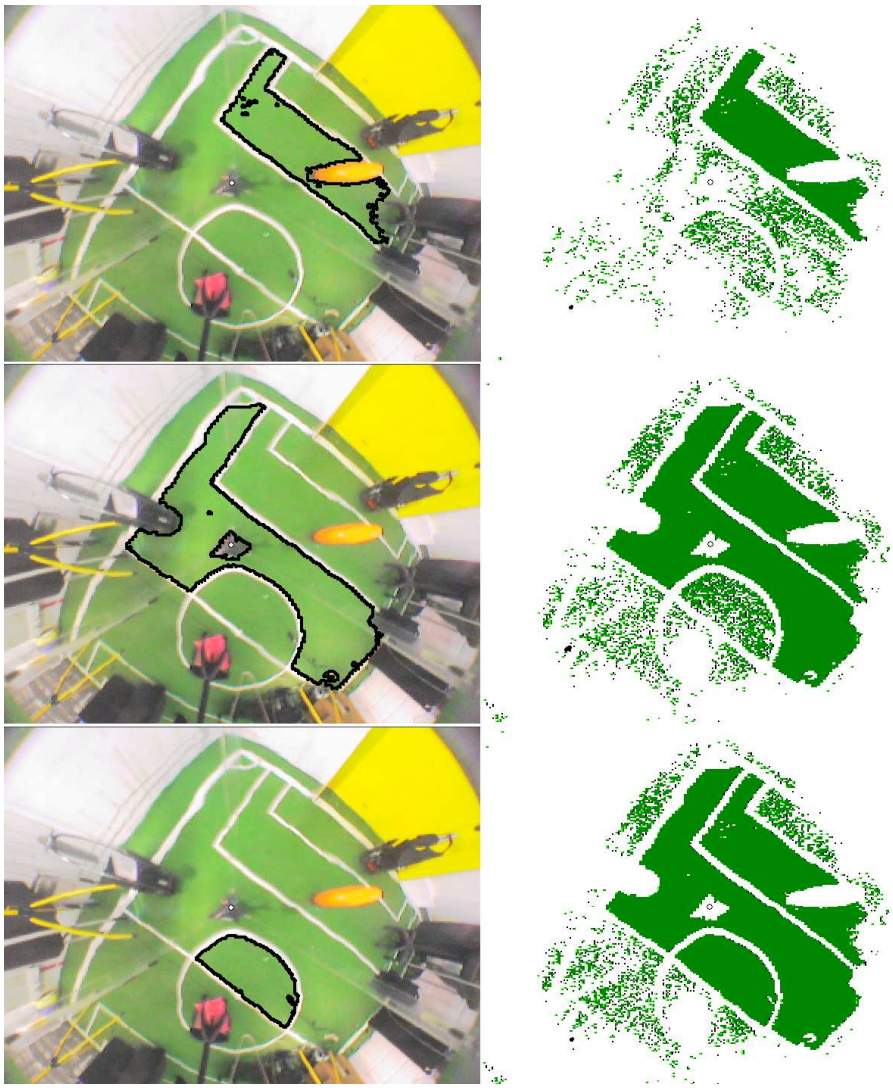

Figure 1.5: The manual calibration process. Here the field is being calibrated. As the operator repeatedly adds colors to the color classes the segmented image improves. Left: Image with a manually grown region. Right: Segmented image after the colors in this and previous regions were added to the corresponding color classes.

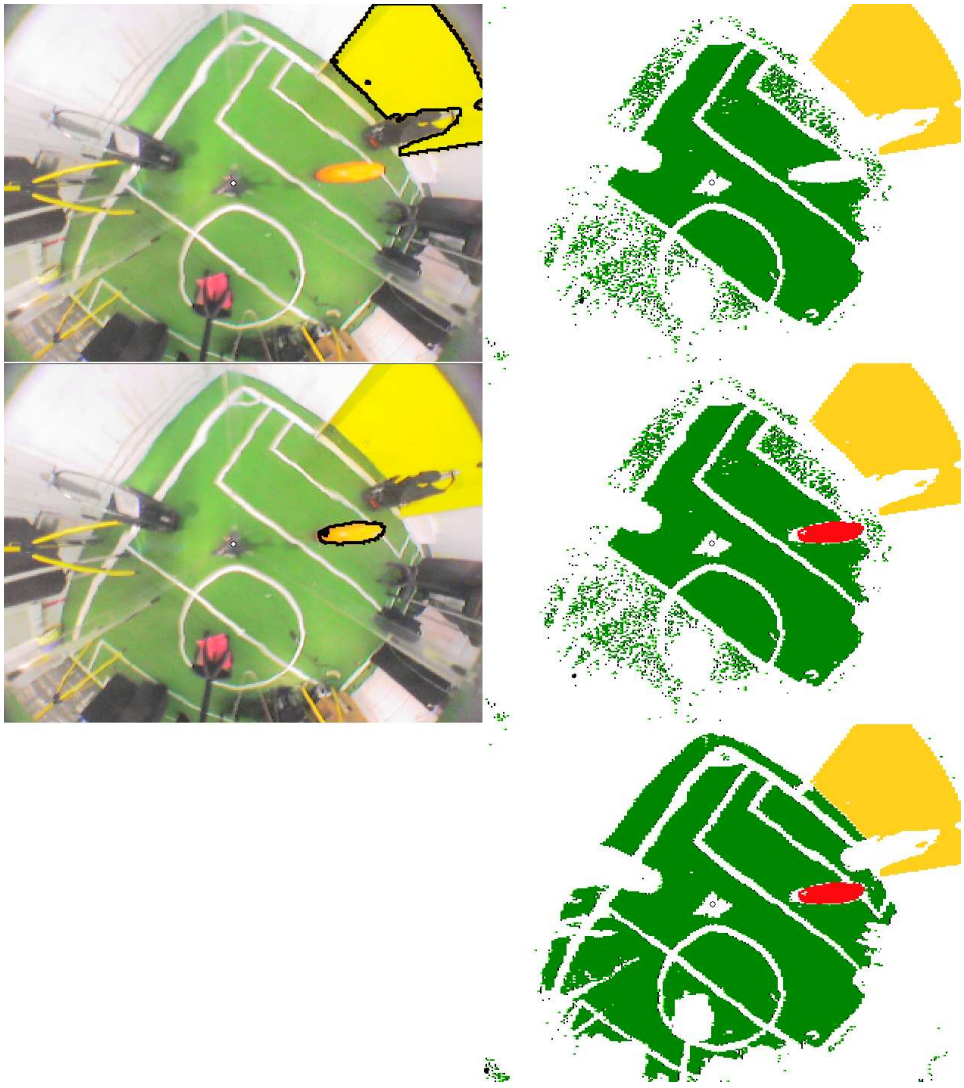

Figure 1.6: Continuation of figure 1.5. Here the yellow goal and the ball are being calibrated. The bottom right image is the final results after adding numerous other regions to the color classes.

Chapter 2

Related Work

Most of the work done on color calibration in RoboCup has been focused on assigning every point in the color-space to a specific symbolic color. The method presented in this work falls in this category, and the next section reviews some of these methods. Thereafter, other approaches that rely more on relative color differences than on accurate calibration are reviewed. Finally, color constancy methods are discussed.

2.1 Automatic Calibration For Look-Up Table Systems

Most vision-systems in RoboCup rely on color-labels to quickly segment an image and detect objects of interest (ball, opponents, field-lines, goal, etc.). They require a table which allows a look-up of the symbolic color (green, orange, yellow, etc.) of any color value in a particular color-space (RGB, YUV, etc.). Therefore, a lot of work has been aimed at filling these look-up tables automatically.

Related work includes [5], which presents a method for off-line, semi-autonomous color-calibration, implemented in the Mid-Size League. Retinex is used for improving color constancy, and k-means clustering is used for the adaptation of color classes. HSV thresholds are found from the clusters that determine each color class, which area then manually mapped to symbolic colors. This method analyzes the vicinity of colors in the color-space and then forms clusters that represent color-classes. In contrast, the method presented here relies on the expected geometrical shape of the objects belonging to a color-class and does not rely on color-space analysis. Furthermore, it is fully automatic, not requiring manual mapping to symbolic colors.

A method called KADC (Knowledge-based Autonomous Dynamic Colour Calibration) is presented in [9]. KADC is a method for autonomous on-line color classification, implemented in the Sony Legged League. KADC also utilizes the geometric information of the field to define color classes, and then updates them with the help of a color cluster similarity metric called EMD. The method presented here is also based on geometric knowledge of the field, but this is combined with the color-less detection of the robot's position. Then the classification is updated using only geometric criteria without having to incorporate any color similarity metrics to previously established classification. This enables handling of abrupt increase/decrease in illumination, which is reported to be troublesome when applying KADC(by [9]). What further differentiates the method presented here from [9] is that the ball color class is also handled.

2.2 Other Approaches

Some vision-systems in RoboCup utilize either relative differences between color-areas solemnly, or in combination with a rough color calibration. In these systems the calibration is integrated into the object-detection step or no calibration is required.

In [15], Hanek et al. propose a method for detecting the ball without the use of color labeling. It is further extended by the same author [16] to run in real-time. The method is based on using local image statistics and fitting a model of the desired object to the image data. A model of the ball is initialized with some values for position and radius, and then iteratively corrected by estimating to which side the pixels near the edge of the model belong. Although the test cases are quite impressive, there is always a strong overlap between the initial position of the model and the final corrected one. It is also interesting to know how fast the method finds the ball from scratch, especially if it is not big in the camera's image. Under these circumstances, color-labeled methods have a big advantage because of the regulated orange colored-ball. Nonetheless, these kinds of methods will probably become more important if RoboCup decides to drop the color-coded ball.

In [6], Juengel et al. present an efficient object detection system (also implemented in the Sony Legged League) which only requires a linear division of the UV-color space. This system is extended in [7] to obtain a more fine-grained classification. The division is calibrated automatically, and the objects are heuristically detected. However, such a linear division and the

use of heuristics may be inadequate for more demanding situations, for example when color classes are not linearly separable, or when numerous color classes are required.

2.3 Color Constancy Methods

Another area of research important for this discussion are color constancy algorithms. If a scene under standard illumination is given, the same scene will experience a shift of color-values in color-space when subjected to a different unknown illumination. The purpose of color-constancy methods in computer vision is to solve this problem by converting every color-space value (e.g. RGB) to what it would be under the standard illumination. Thus, color constancy methods can make vision-systems in RoboCup robust against illumination changes, but the color calibration still has to be made for the standard illumination (manually or automatically). Since the color of objects in RoboCup varies from one competition site to the other, manual calibration is still very inconvenient. Therefore, there has been more interest in auto-calibrating methods in RoboCup.

At least two approaches to color constancy have been used in RoboCup; an implementation of the Retinex algorithm (mentioned earlier, see [5]) and an algorithm using self-organizing feature maps (see [10]). Retinex is a biologically-inspired algorithm which was first proposed by [19]. Retinex algorithms are based on calculating the so called *lightness* of a pixel. Lightness, in Retinex theory, is a relative measurement of a pixel's brightness, measured in each of it's color channels separately. It is calculated by taking the average of a pixel's intensity ratio to many other pixels in the image. The usual approach is to choose a number of short paths and compare the pixels on a path to the path's first pixel. At each pixel the ratio is recorded, and it's lightness is obtained by averaging these ratios.

Retinex algorithms seem to be too slow for on-line processing, although real-time implementations exist with the use of specialized hardware (see [18]).

Austermeier et al. ([10]) find the color-correspondence between two illumination settings by using self-organizing feature maps (SOM) in color-space. Two SOMs are built, one for the cloud of color points under the initial reference illumination and another one for the new illumination settings. The distance between corresponding grid locations in the two maps is then used as a correction vector for the set of color points belonging to that volume element. Unfortunately, the method is very computationally

expensive.

Another class of color constancy methods, used in many consumer cameras, are based on the "gray-world assumption" of G. Buchsbaum[14] and referred to as gray world methods. They are based on the assumption that the average of the surface reflection of a typical scene is gray (or some other constant color). This method can be implemented fast, but has undesired effects in a RoboCup scenario. For example when a large dark object is present in the image such as the blue goal or a robot, the algorithm will change all colors in the image. This can result in the green field or the orange ball to have colors others than previously defined, therefore causing a problem in the object-detection algorithms.

An alternative to gray world methods is to place a colored or white patch on your robot which is visible in the camera's image at a fixed position. The image is then corrected globally until the image area occupied by the patch has reached it's target value. This method, in contrast to gray world methods, is not affected by large dark or bright objects in the image. However, non-uniform lighting such as when a shadow or a bright light falls on the patch, will bias the image globally. An implementation of this method was used in the FU-Fighters Mid-Size team, where a piece of white paper was attached around the camera's lens. White-balance was achieved by adjusting the camera's parameters with the help of two PID-controllers (see [17]).

Instead of placing a white patch on the robot, one could drill a hole in the middle of the omni-directional mirror and use the incoming light to adjust the camera. Since only the robot can be seen in the mirror's center and is not an object of interest for the vision-system, this would not be a loss of an important image region.

Chapter 3

Automatic Color Calibration

3.1 Overview

The method presented here automatically classifies regions of homogeneous color into the following four color classes: field, ball, yellow goal, and blue goal. It essentially works by mimicking what the operator does when calibrating manually. After acquiring an image from the robots camera, a color region is grown and its colors assigned to the corresponding color class. In order to find the right color class, available geometric information about the field and ball is utilized, as well as the robot's localization. Thus, a grown color region must have a shape which fits the geometrical criteria of one of the color classes. Regions that do not fit the criteria of any of the classes are not classified and can be considered obstacles.

For each color class the method consists of the following steps:

- localize the robot on the field using edge detection (see chapter 3.2).
- loop:
  - Grow a homogeneous color region (see chapter 3.3).
  - Compare the grown region's size and shape to the size and shape of the corresponding expected region (see chapter 3.4).
  - **if** the grown region is within the boundaries of the expected region, and fills more than a certain percentage of the expected size:

    add all the colors in the grown region to the corresponding color class.

- **else**

  add no colors to the color class.

The quality of the calibration improves as more and more iterations of this loop are executed. Good calibration is usually obtained after 1-2 seconds as illustrated in the results chapter (see chapter 4). The homogeneity thresholds for the color growing are computed automatically (see chapter 3.5).

3.2 Color-Less Localization

In order to allow image coordinates to be transformed into world coordinates, the robot needs to localize itself in an initialization step. This is done by using a region tracker (described in [2]), which stops growing a region when a possible field-line is encountered. The stopping criterion is based on the assumption that a field-line is bright compared to the neighboring points. In the FU-Fighters implementation 4 pixels are used, to the left, right, above and below the actual point p, which have a pixel distance from it corresponding to the expected field-line width (see Fig. 3.1). A field-line is considered to be found if p is one standard deviation σ brighter than at least two of its neighbors, where σ is the standard deviation of the brightness of those pixels in the image which correspond to points on the field. The value of σ is calculated on-line by sampling a certain number of random pixels in the image.

Subsequent processing requires extraction of the pixels that coincide with the points of the white field-lines. To accomplish this, a search is performed for dark-white-dark transitions on short scan-lines perpendicular to the edges of each of the tracked regions. This is done in the following manner: find the brightest point p on the scan-line. Inspect the endpoints s and e of an extended scan-line centered on p and having length twice the expected field-line width (the length was tuned experimentally, the idea is that s and e do not lie on the field-line, see Fig. 3.1). If p is σ-brighter than s and e, declare it a white pixel corresponding to a point on a field-line.

The set of points obtained in this way is the input for a special localization algorithm. The localization exploits the presence of certain features in the field's line-model (center circle, corners, etc. see Fig. 3.2) and localizes the robot using them and a force field matrix (Hundelshausen et al. describe this localization technique in [1] and [3]). The localization obtained this way is uniquely determined up to the symmetry of the field because

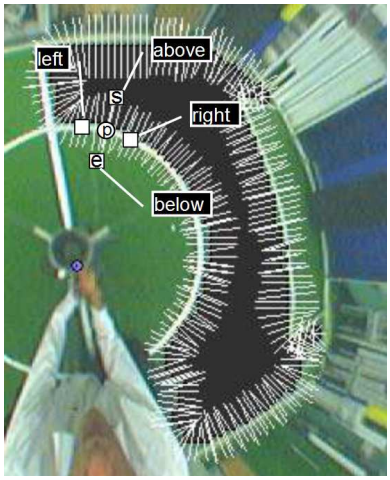

Figure 3.1: A tracked region (painted black) and white scan-lines along its edges. Here a green region is being tracked. s and e are the start and end-points of the scan-line. p is the actual point to be checked for edge criteria, and the points marked above, below, left and right, are its neighbors used to determine the brightness around p.

no information about the two goal box colors is available. Nonetheless, the method can proceed without it, as will be explained in the next chapter.

A drawback of this localization is that more false field-line points are found than with the previously used localization, which tracks green regions. It is also potentially slower since more pixels are processed. Even though it would be possible to localize the robot by calibrating the goal colors only (to break the field's symmetry), there is still a need for calibrating the color of the field. Without it, it would be impossible to identify obstacles on the field.

3.3 Choosing Regions by Color Growing

The second step is to grow homogeneous color regions. It is not important to start growing a region at a specific pixel, but choosing them intelligently can accelerate the classification. This is achieved by building different sets of starting pixels for each expected region. Subsequently, one pixel is chosen at random from each set and separate color region growing processes are started (see figure 3.3). The grown color regions are then passed along for

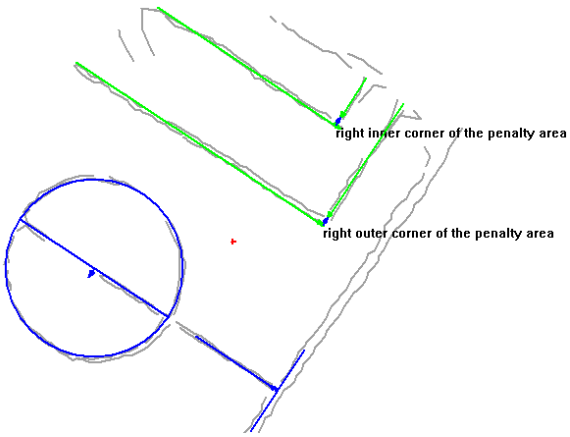

Figure 3.2: Example of features found in the field-line contours. Here the center circle, a T-junction and the inner and outer right penalty area corners have been successfully detected. With these features the robot can localize itself on the field.

further validation (see chapter 3.4). Different pixel criteria are required to obtain the various pixel-sets used for the expected regions of the field, the ball, and the goals.

Since the green field color usually covers a large area of the image, a possible method to obtain the field pixel-sets would be to pick a certain amount of randomly chosen pixels and assign them to the pixel-set of the expected region they correspond to. Instead, pixels are gathered from the field-line detection procedure which provides pixels that are close to a line, but not on one. These pixels - excluding the ones corresponding to points lying outside the field, are then assigned to the pixel-set of the expected world region they correspond to (there are 10 such field-regions, see Fig.3.4).

A goal-box pixel-set consists of pixels corresponding to points lying behind one of the two goal lines in the field-model. The two goal-box pixel-sets are separated by the spatial location of the goals. It is decided later to which goal-box the two sets belong (see chapter 3.4).

Since the ball can be small in the image and its position in general unknown (even if the robot's position is known), it pays off to pick the pixel-set for the ball color-growing carefully in order to make its classification faster. The procedure used therefore only considers pixels corresponding to field-points (because the ball is on the field). It then checks if a pixel has

Figure 3.3: Regions grown successfully for an expected ball region, one of the expected field regions, and an expected goal region. The grown goal region is enclosed by a white curved line, the ball's by a red curved line and the field's by a green curved line.

either been classified as the ball color in a previous iteration of the calibration procedure, or if it could be the center of an expected ball at that point. If this is the case, the pixel is added to the ball pixel-set. Essentially this is a form of pre-validation that verifies if a region starting from this pixel could ever grow into the expected ball at this pixel. It does this by checking if pixels along the axes of the expected ball ellipse are unclassified.

Growing a homogeneous color region works in the following manner: Starting with a pixel p, neighboring pixels are inspected. If their color is within a certain homogeneity threshold with respect to the color at p, they are added to the color region. The neighbors of the newly added pixels are inspected in the same manner, always comparing them to the color of the first pixel p. The homogeneity thresholds are adapted automatically (see chapter 3.5).

3.4 Validating Grown Color Regions

After picking one point from each pixel-set and growing separate color regions (one region for the ball, ten regions for the field, and two regions for the goals), it has to be verified that they belong to the corresponding expected regions. To ensure this, the regions have to pass through validation

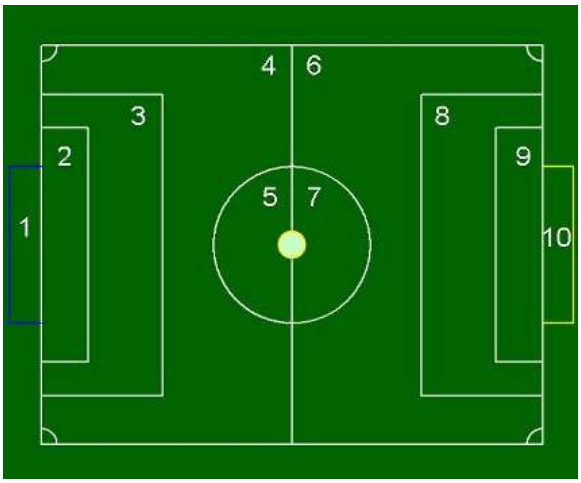

Figure 3.4: The 10 expected field regions.

criteria. The criteria are similar for each color class, and are based on the following observation: if a grown color region r is totally inside and covers an expected region of a color class C, then the colors in r are members of C. The expected regions are defined assuming ideal conditions such as an obstacle-free field and perfect self-localization.

In the case of the field color class, 10 expected regions were defined in world coordinates using the current field-model (see Fig. 3.4). In accordance with the general guidelines, a grown field-color region should lie entirely inside one of the expected regions. After checking that the region fulfills this criterion, it is verified that it covers enough of the expected region. In the FU-Fighters implementation it is required that a field-color region covers 70% of the corresponding expected region, and that all of its points lie inside it. If the criteria are not fulfilled, no colors from the region are added to the field color class in this iteration.

In the case of the goal-box color classes, it is possible to use the field line-model and the robot's position to calculate at which angle in the image a grown goal-box-color region is expected to appear. Furthermore, it is known that this region cannot be inside any of the expected field regions. In the FU-Fighters implementation it is required that a goal-color region lies between the angle defined by the goal-box's posts, and that it covers 70% of the angle. Furthermore, the goal-box has to be clearly visible given the current position of the robot, e.g. the expected angle to the left and right

posts has to be sufficiently large. Also, no point of the region can lie in any of the expected field regions. If the criteria are not fulfilled, nocolors from this region are added to the respective goal-box color class in this iteration.

Once a grown goal-color region has been successfully validated, and its colors have been associated with one of the arbitrarily chosen goal-boxes, the symmetry of the field has been broken, and the sides can be labeled and recognized. Future validated goal-color regions will therefore be assigned to the correct goal-box color class. This is based on the assumption that the robot does not de-localize and flip sides, while the illumination simultaneously changes to prevent goal-identification. However, since there is a convention in RoboCup to paint one of the goals yellow, and the other one blue, the FU-Fighters implementation compares a grown goal-color region to a blue and a yellow reference color. It is then added to the class whose reference color is closer to the region's mean color. This automates the setup of the robot and also increases the robustness of the classification.

In the case of the ball color class, an expected ball region in the image has an elliptic form where the size of the minor and major axis depends on the distance from ball to robot. The expected ball is represented by storing ball-fitting ellipses at different distances from ball to robot. One ellipse data entry consist of the pixel distance from the robot to the center of the ellipse (the robot being in the center of the image), as well as the minor and major axis of the ellipse, measured in pixels. In the FU-Fighters implementation is is required that all pixels in a ball-color region be inside the expected ball region, and that the area be more than 40% of the expected area. Furthermore, the ball has to be encircled by pixels classified as the field color class. This extra criterium was added to prevent human hands from beeing classified as the ball. If the criteria are not fulfilled, no colors from the region are added to the ball color class in this iteration.

3.5 Adaptive Thresholds for Color Growing

The thresholds for color growing are in general not the same for each color class, and vary with lighting conditions. Furthermore, it is advantageous to use various thresholds for the same color class in one and the same scene. This is especially advantageous in the case of the field class, because it is large and can therefore have wide variations in color homogeneity. Accordingly, three separate sets of thresholds are deployed, one for each expected region of the field, the ball and the goals. These sets are initialized with a constant amount of random thresholds. The thresholds are then adjusted

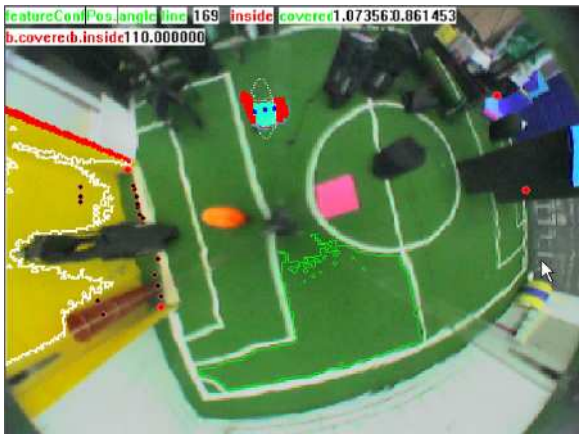

Figure 3.5: Grown regions which fail to meet the validation criteria. Pixels of the goal-box and ball regions are outside the expected region, or the field region does not cover a required percentage of the expected region.

with the help of the validation criteria outlined previously in chapter 3.4. Essentially, this means decreasing the threshold if the region grown using it was too big, and increasing it, if it was too small.

Before a region is grown, a threshold is picked at random from the corresponding set. If a certain threshold was involved in a successful growing, it "survives" and is still part of the set in the next iteration of the calibration procedure. If a region growing has failed a certain number of times using the same threshold, the threshold "dies" and a new randomly initialized threshold takes its place in the set. Each time a region grown with a threshold is too big, the threshold is decreased by a small random amount. If the region is too small, the threshold is increased, and another try to grow a region at the same point is made. The increasing of the threshold is continued until the region is successfully grown, or grows outside the expected region.

3.6 Handling of Multiple Color Class Membership

Using the method presented in this work, it is possible for the same color to be assigned to multiple color classes. This happens when colors belonging to a color class are present in regions not belonging to the class. This is for example the case when the blue goal is so dark that some colors in it also appear in robots. In this case, a very dark-blue or black region is grown inside the goal, which is found to correspond to the expected goal region. The method then defines these colors as belonging to the blue goal color-class even though they are encountered in robots or other dark objects as well. This problem can occur with other color-classes as well, for example between the orange ball color and the yellow goal color (although this is not common).

The traditional solution, used by many teams in RoboCup, is to solve these conflicts manually with a GUI-tool that allows you to exclude certain color values from a color class. An automatic solution is to mark a color as not belonging to a class if it occurs in an unexpected region. This has been implemented in the FU-Fighters system with mixed results; the aforementioned effect of color values belonging to multiple color classes is reduced, but this comes at the expense of "banning" a color value forever from a color class. This would be unfortunate if the membership of the color values needed to be changed later, for example when the lightning changes. Another drawback is the computational cost of searching for wrongly classified color values.

The following implementation is used in the FU-Fighters system. First, the membership of the color values appearing in the actual image are checked. Then, the corresponding pixels are projected into the world. Finally, the pixels are checked for correct membership given the position. Those memberships which do not fit the position are deleted from that color value and marked as unsuitable.

A possible extension to this solution is to delete this markings after a certain time. This requires a book-keeping system where each color value is updated with a time stamp. When a high color resolution is used, such a system needs a lot of memory and processing time. It has not been implemented in the FU-Fighters system.

Another way of handling multiple membership is to ignore it in the calibration system and deal with it in later processing steps. This is the approach used in Juengel et al (see [6]). There, the yellow and the orange

Figure 3.6: In this bright image the same color values are found on the field, in the ball, and in the yellow goal. Therefore the same color value can be assigned to multiple color classes.

color classes, which are sometimes hard to separate, were simply combined into one color class. The object detection algorithms then analyse a color transition and can determine if it is a green-orange or a green-yellow edge, for example.

Chapter 4

Results

The method presented in this work is able to adapt to changes in illumination in a few seconds. The method was tested on a FU-Fighters Mid-Size robot which are equipped with a lightweight laptop having a Pentium III M 933 MHz processor, and 256 MB RAM. For the run-time tests of the method the time it took to adapt to an abrupt change in illumination was recorded.

Fig.4.1 illustrates the performance of the method where the robot is close to a border line and sees a large area outside the field. The classification (on the right) was obtained after a few second with no previous classification. Here the ball and the goals are too far away to be calibrated.

The scene used for the main run-time tests can be considered a standard RoboCup scene with the ball in view, and a robot in the goal, except that a foreign pink piece of paper is present in the field. Note that it will not be classified since it does not fit any expected region. The results of the classification can be seen in Fig.4.2. The first column shows the original images. The second column shows a segmented image using the automatic classification from the previous illumination setting without adjusting to the new illumination (the first row has no previous classification). As can be seen, under the new illumination the color segmentation is poor. The third column shows a segmented image after adapting the classification from the previous scene with the automatic calibration method. The runtime for the adaptation for row 1-4 was: 0.5, 0.8, 2.0, and 2.9 seconds, respectively. The current implementation does not try to match the field-regions inside the goal (regions 1 and 10 in Fig. 3.4), and therefore its colors are not classified in any of the scenes. The regions on the other side of the field are also not classified since they are obscured, and hence can not be matched to the corresponding expected regions. The first row demonstrates the classification

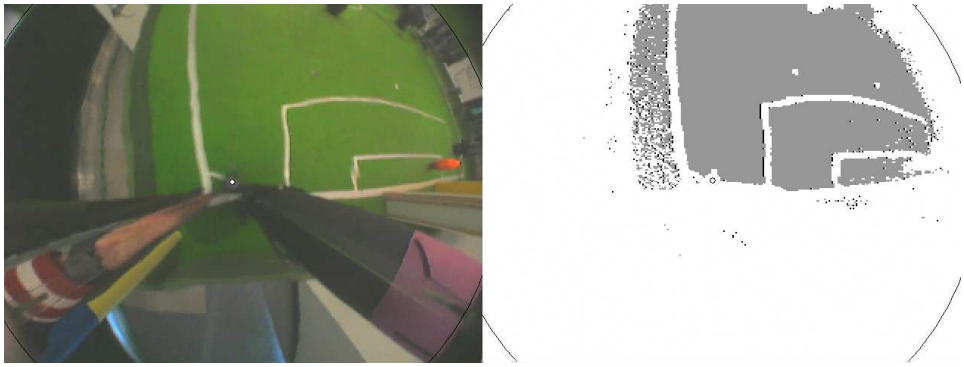

Figure 4.1: The classification (on the right) was obtained after a few second with no previous classification. Here the ball and the goals are too far away to be calibrated so only the field can be calibrated. The color code is: white = unclassified, gray = field.

under a mixed neon - and indirect floodlight. All regions that are clearly visible have been successfully classified. The same goes for the second row, which displays a darker scene with neon lighting only. The third row shows a classification under neon lighting, and with one floodlight directed down on the field. Here the method fails to classify some of the brightest green colors lying under the floodlight after 2.0 seconds, but after letting the method run for about 12 seconds, the classification improves (not illustrated), without managing to classify some of the extremely bright green colors. The fourth and last row of images was captured under neon lighting and with three floodlights directed down on the field. A successful classification of this scene was obtained after 2.9 seconds. Note however, that the colors of the inner penalty area are not classified. This is due to the fact that the goalie is placed in the middle of it, and thereby cuts the homogeneous color region in half. It can therefore not be matched properly to the expected area of the inner penalty area.

A short video (2-3 sec.) from a second scene was used to compare the automatic calibration to the previously used manual calibration. The number of color values assigned to color classes, were 46422 with the manual calibration and 45804 with the automatic calibration. **Ignoring wrongly classified entries, this equals a** 98.7% **success rate in the automatic calibration compared to the manual one.** As can be seen in figure 4.3 the difference in the segmented image obtained with the manual and auto-

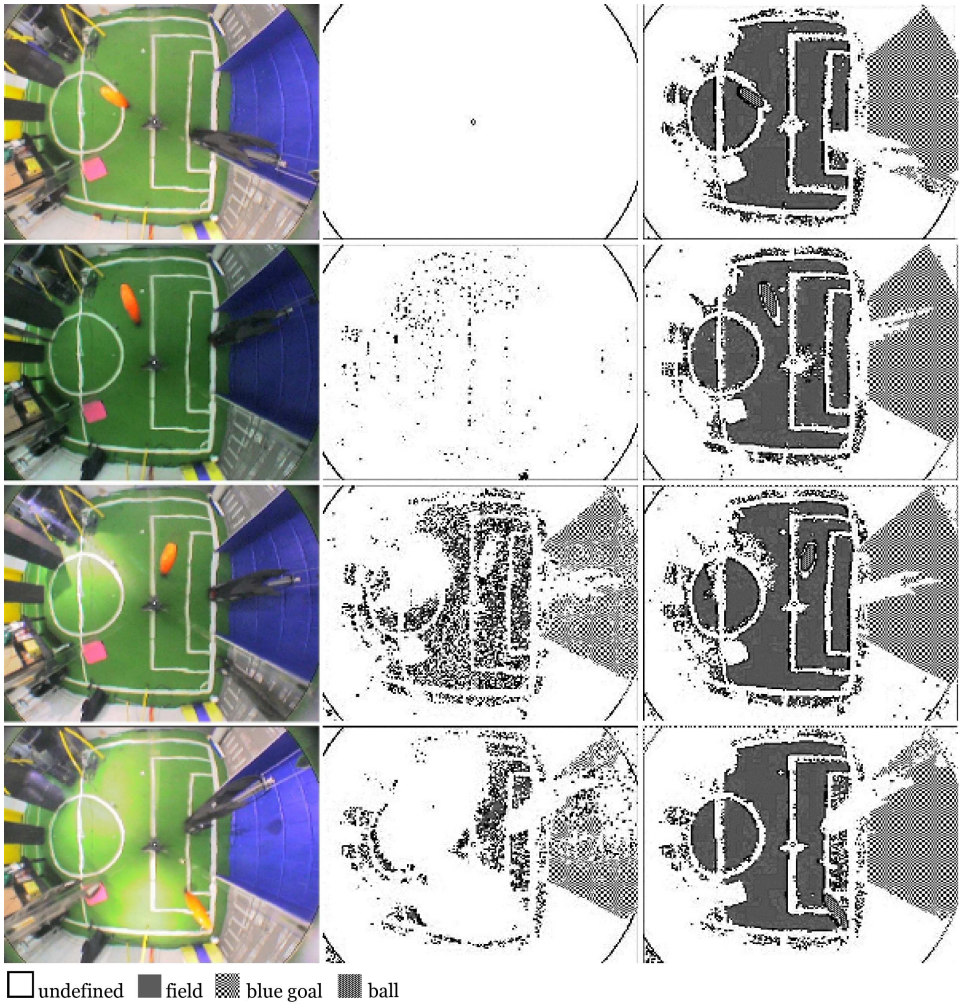

Figure 4.2: Row 1 to 4 (counting from top to bottom): Original images on the left, static classifications from previous illumination setting in the middle, and the result of the automatic classifications on the right. The CPU-time it took the method to produce the automatic classifications (right), starting with the classifications achieved from the prior illumination (middle) for row 1-4 was: 0.5, 0.8, 2.0, and 2.9 seconds, respectively. The color code is: white = unclassified, gray = field, check-board-pattern = blue goal, diagonal-pattern = ball.

matic calibration is insignificant. If examined closely one can see that the segmented image of the manual calibration contains less unclassified pixels in the yellow goal and the ball as well as one the field border (the actual implementation does not try to classify color regions grown on the field border). Another way of comparing the two calibration is by drawing the color values in a YUV color cube. Figure 4.4 shows a few snapshots from the color clouds resulting after each calibration method. In this image it can also be observed that the manual calibration classifies slightly more color values, therefore producing larger clouds in the YUV color cube.

Figure 4.3: A comparison of the number of color values classified with the automatic calibration against the number classified with the manual classification. Top: An image from the test video used for the comparation. Center: a segmented image obtained with the automatic calibration. Bottom: a segmentated image obtained with the manual calibration.

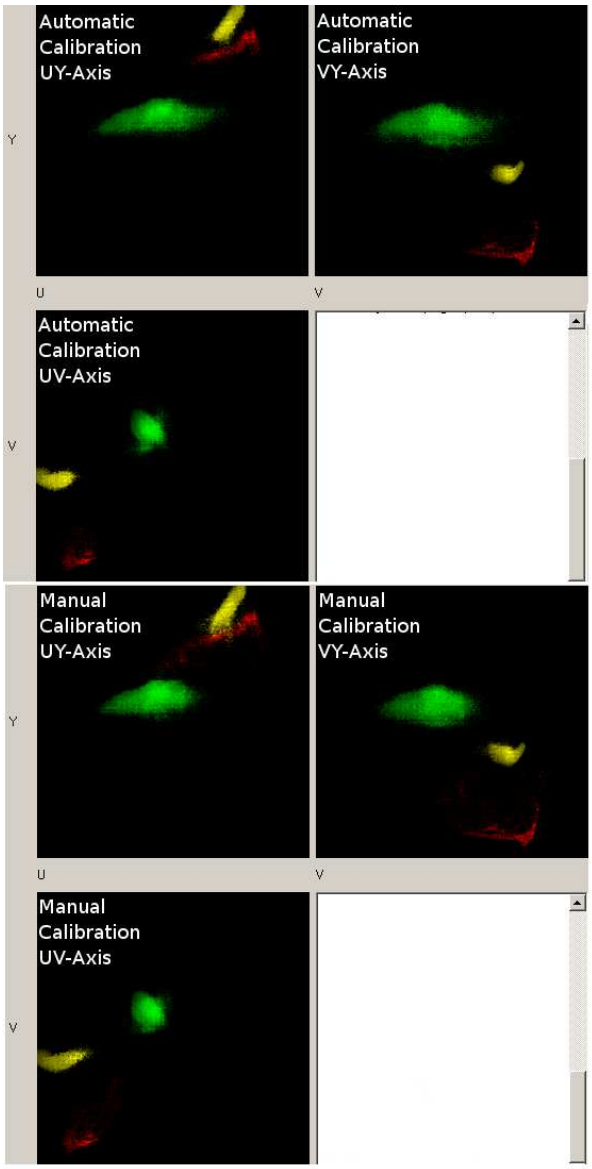

Figure 4.4: Continuation of figure 4.3. The top three figures are snapshots of the color values classified with the automatic calibration in the YUV color space. The bottom three figures are the respective snapshots for the manual calibration. One can see that the manual calibration produces larger color clouds, especially for the red and yellow color class.

Chapter 5

Future Work and Summary

Another potential weakness of the method is that a color does not become "outdated", e.g. a color cannot loose a previous classification. This can present a problem when the lighting is changed, for example from white neon lighting to a more warm, yellow lighting. Now, colors that were previously classified as the yellow goal can appear on the white field-lines. An approach used in [9], is to incorporate a color decay factor.

A method for tuning the camera-parameters is presented in [8], and could be combined with the method presented here to enable the robot to operate in a wider range of lighting-conditions. The "ground truth" (manually defined color classes) needed in that method could be provided by the automatic calibration.

Since the two team colors magenta and cyan are not used by the FU-Fighters vision-system they are currently not included in the automatic calibration process. Calibration of cyan and magenta could be done in the following manner. First, calibrate colors with the current automatic calibration method. Then, find the robots using the current object detection system. Finally, search for a patch of magenta or cyan color in the expected position. It is uncertain how accurately cyan and magenta could be calibrated with this approach. Also, as the number of color classes increases, the chances of multiple color class membership increases as well.

As the method is based on comparing observed geometrical objects to known objects, it is crucial that the distance function used to convert from image to world coordinates is accurate. In the FU-Fighters system this function is tuned manually which often results in erroneous transformations. Reducing this error by automatically calibrating the distance function would improve the performance of the automatic color calibration method.

In this work a method that can be used for automatic color calibration of autonomous soccer playing robots was presented. It is based on a color-less localization of the robot, a geometric line-model of the field and a geometric model of the ball. The method needs no manual calibration and can deal with various difficult lighting conditions that change abruptly over time. It can be integrated into existing systems and was used by the FU-Fighters Mid-Size robots at RoboCup 2005 in Osaka. There the FU-Fighters Mid-Size team got the second place, whereby the automatic calibration considerably reduced setup-time and contributed to a more exact calibration.

Bibliography

[1] Felix von Hundelshausen, Michael Schreiber, Fabian Wiesel, Achim Liers and Raúl Rojas: *MATRIX: A force field pattern matching method for mobile robots*, Technical Report B-08-03, Freie Universität Berlin, Institute of Computer Science, Takustr. 9, 14195 Berlin, Germany, available at: http://robocup.mi.fu-berlin.de/docs/matrix.pdf, link checked: 2.feb 2005.

[2] Felix von Hundelshausen, and Raúl Rojas: *Tracking Regions*, in Daniel Polani et al.(editors):RoboCup-2003: Robot Soccer World Cup VII (Lecture Notes in Computer Science), Springer, 2004. Available at: http://robocup.mi.fu-berlin.de/docs/RegionTrackingRoboCup03.pdf, link checked: 2.feb 2005.

[3] Felix von Hundelshausen, *A constructive feature-detection approach for mobile robotics.* Proceedings of the RoboCup 2004 International Symposium, Lisboa, Italy July 5-7, 2004.

[4] Felix von Hundelshausen, Raúl Rojas, Fabian Wiesel, Erik Cuevas, Daniel Zaldivar, and Ketill Gunnarsson: *FU-Fighters Team Description 2003*, in D. Polani, B. Browning, A. Bonarini, K. Yoshida (Co-chairs): RoboCup-2003 - Proceedings of the International Symposium.

[5] Mayer, G., Utz, H., Kraetzschmar, G.K.: *Towards autonomous vision self-calibration for soccer robots.* IEEE/RSJ International Conference on Intelligent Robots and Systems (2002) 214-219.

[6] Jüngel, M., Hoffmann, J., Lötzsch, M. (2004). *A real-time auto-adjusting vision system for robotic soccer.* In: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence, Padova, Italy, 2004. Springer (to appear).

[7] Matthias Jüngel. *Using Layered Color Precision for a Self-Calibrating Vision System.* Daniele Nardi, Martin Riedmiller, Claude Sammut, José Santos-Victor (Eds.): RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Computer Science 3276 Springer 2005, ISBN 3-540-25046-8.

[8] Erio Grillo, Matteo Matteucci, Domenico G. Sorrenti: *Getting the Most from Your Color Camera in a Color-Coded World.* Daniele Nardi, Martin Riedmiller, Claude Sammut, José Santos-Victor (Eds.): RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Computer Science 3276 Springer 2005, ISBN 3-540-25046-8.

[9] D. Cameron, N. Barnes, *Knowledge-Based Autonomous Dynamic Colour Calibration*, in Proc. Robocup Symposium, 2003, Padua, Italy, July, 2003. RoboCup 2003 award-winning paper: *Engineering Challenge Award*.

[10] Austermeier, H., Hartmann, G., Hilker, R.: *Color-calibration of a robot vision system using self-organizing feature maps.* Artificial Neural Networks - ICANN 96. 1996 International Conference Proceedings (1996) 257-62.

[11] http://www.fu-fighters.de

[12] http://ais.gmd.de/GO/2003/pictures_main.html

[13] http://www.zdnet.co.jp/news/0206/19/simulation.jpg

[14] G. Buchsbaum. *A spartial processor model for object color perception.* In Journal of the Franklin Institute, Volume 310, Seiten 1-26. 1980.

[15] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. *Towards RoboCup without Color Labeling.* In RoboCup International Symposium 2002, Fukuoka, Japan, 2002, award-winning paper.

[16] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. *Fast Image-based Object Localization in Natural Scenes.* In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002, Lausanne, Switzerland, 2002

[17] Slav Petrov. *Computer Vision, Sensorfusion und Verhaltenssteuerung für Fussball-Roboter*, master thesis (Diplomarbeit), Institut für Informatik, Freie Universität Berlin, 2004.

[18] G. D. Hines, Z. Rahman, D. J. Jobson, G. A. Woodell, S. D. Harrah. *Real-time Enhanced Vision System*, Enhanced and Synthetic Vision 2005, Proc. SPIE 5802, (2005).

[19] Land, E. H. and McCann, J. J. *Lightness and retinex theory.* (1971) Journal of the Optical Society of America, 61 (1), 1-11.

Hiermit versichere ich, dass ich diese Diplomarbeit selbständig angefertigt habe, sämtliche Quellen, die verwendet wurden, angegeben habe und dass die Arbeit nicht schon an anderer Stelle zur Prüfung vorgelegt wurde.

Ketill Gunnarsson

Berlin, den 1. November 2005.