The Color and the Shape: Automatic On-line Color Calibration for Autonomous Robots

Ketill Gunnarsson, Fabian Wiesel, and Raúl Rojas

Freie Universität Berlin,
Institut für Informatik,
Takustraße 9,
14195 Berlin,Germany
{ketill|wiesel}@inf.fu-berlin.de

Abstract. This paper presents a method for automatic on-line color calibration of soccer-playing robots. Our method requires a geometrical model of the field-lines in world coordinates, and one of the ball in image coordinates. No specific assumptions are made 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 classification 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 color class. This method can be run in a background thread thus enabling the robot to quickly recalibrate in response to changes in illumination.

1 Introduction

Color classification in the RoboCup Mid-Size League usually involves tedious calibration procedures. A typical approach is to manually define which parts of the color space correspond to a color class using some kind of 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 environment are strictly color-coded and the organizers try to provide lighting that is as steady, bright and uniform as possible.

Our method 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 our application, we store these lists 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.

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

- localize the robot on the field using edge detection(see section 2).
- loop:
  - Grow a homogeneous color region (see section 3).
  - Compare the grown region's size and shape to the size and shape of the corresponding expected region (see section 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 homogeneity thresholds for the color growing are computed automatically (see section 5).

Related work includes [5], which presents a method for off-line, semi-autonomous color-calibration, implemented in the Mid-Size League. RETINEX, a biologically-inspired algorithm 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 are 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, our method 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. Our method is also based on geometric knowledge of the field, but we combine this with the color-less detection of the robot's position. We then update the classification using only geometric criteria without having to incorporate any color similarity metrics to previously established classification. This enables us to deal with an abrupt increase/decrease in illumination, which is reported to be troublesome when applying KADC(by [9]). What further differentiates our method from [9] is that we also handle the ball color class.

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.

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. This solves the problem of maintaining a classification, but does not generate it. Unfortunately, the method is also very computationally expensive.

The method we present was tested on our Mid-Size robots, which are equipped with an omni-directional vision system and use conventional sub-notebooks for processing (see [4]). We now proceed with a step-by-step description of our calibration method. We then describe how the thresholds used for the color growing are found and adapted. Finally, we present experimental results.

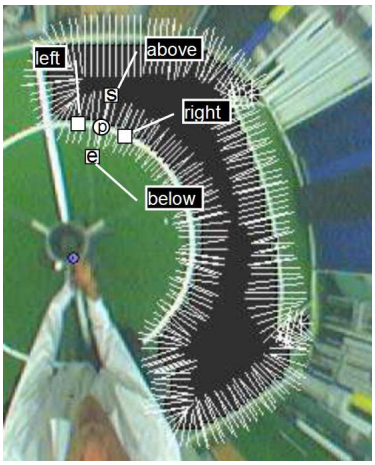

Fig. 1. A tracked region (painted black) and white scan-lines along its edges. Here a green region is being tracked. s and e are the start and end-points of the scan-line. p is the actual point to be checked for edge criteria, and the points marked above, below, left and right, are its neighbors used to determine the brightness around p.

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. We do this 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 our implementation we use 4 pixels 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. 1). We decide that a field-line is 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.

Now we need to extract the pixels that coincide with the points of the white field-lines. To accomplish this, we search 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. 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 our localization algorithm. The localization exploits the presence of certain features in the field's line-model (center circle, corners, etc. see Fig. 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 we obtain this way is uniquely determined up to the symmetry of the field because we have no information about the two goal box colors. Nonetheless, the method can proceed without it, as we explain in the next section.

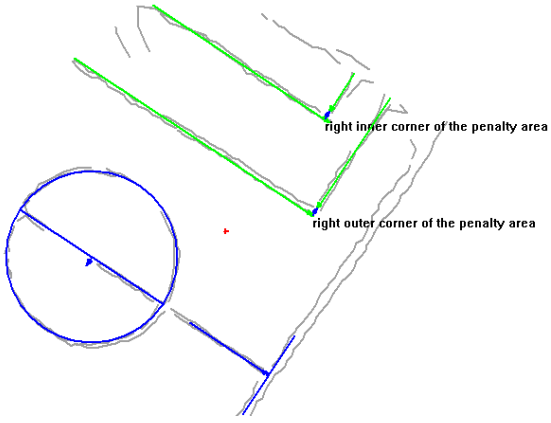

Fig. 2. Example of features found in the field-line contours. Here the center circle, a T-junction and the inner and outer right penalty area corners have been successfully detected. With these features the robot can localize itself on the field.

A drawback of this localization is that more false field-line points are found than with our conventional localization, which tracks green regions. It is also potentially slower since more pixels are processed. Even though we could 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 we cannot identify obstacles on the field.

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). The grown color regions are then passed along for further validation (see section 4). Different pixel criteria are required to obtain the various pixel-sets we use for

Fig. 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.

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, we gather pixels from our field-line detection procedure which provides us with 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.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. We decide later to which goal-box the two sets belong (see section 4).

Since the ball can be small in the image and we don't know where to look for it (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 we use only considers pixels corresponding to field-points (because the ball is on the field). It then checks if a pixel has 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, we add the pixel 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 section 5).

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), we need to make sure that they belong to the corresponding expected regions. To ensure this, the regions have to pass through validation 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, we can define 10 expected regions in world coordinates using the current field-model (see Fig. 4). In accordance with our general guidelines, a grown field-color region should lie entirely inside one of the expected regions. After checking that the region fulfills this criterion, we check if it covers enough of the expected region. In our implementation we require a field-color region to cover 70% of the corresponding expected region, and all of its points to lie inside it. If the criteria are not fulfilled, we do not add any colors from the region to the field color class in this iteration.

In the case of the goal-box color classes, we can use the field line-model and the robot's position to calculate at which angle in the image we expect a grown goal-box-color region to appear. Furthermore, we know that this region cannot

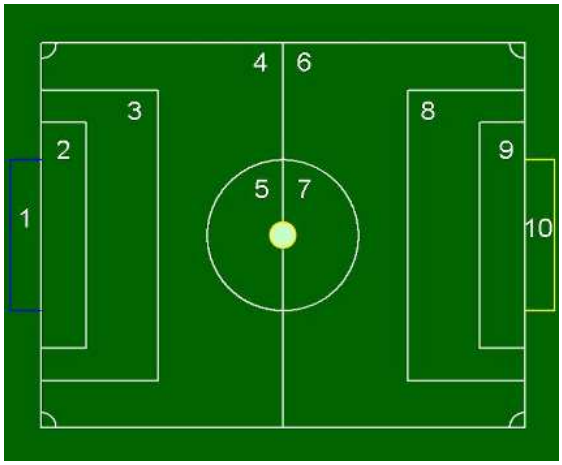

Fig. 4. The 10 expected field regions.

be inside any of the expected field regions. In our implementation we require a goal-color region to lie between the angle defined by the goal-box's posts, and to cover 70% of the angle. We also require the goal-box to be clearly visible given the current position of the robot, e.g. that the expected angle to the left and right posts is sufficiently large. Furthermore, no point of the region can lie in any of the expected field regions. If the criteria are not fulfilled, we do not add any colors from this region 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, we compare a grown goal-color region to a blue and a yellow reference color in our implementation. We then add it 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. We represent the expected ball 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 our implementation we require all pixels in a ball-color region to be inside the expected ball region, and the area to be more than 40% of the

expected area. If the criteria are not fulfilled, we do not add any colors from the region to the ball color class in this iteration.

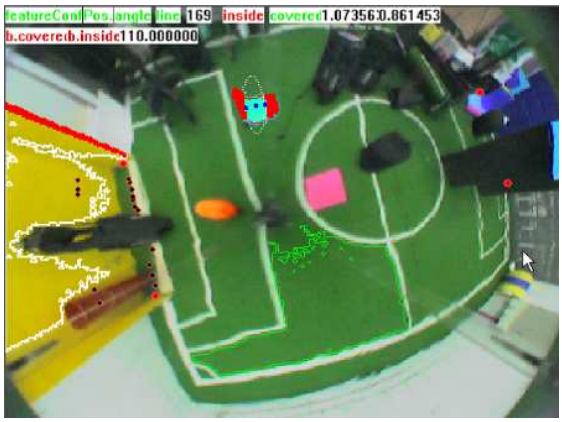

Fig. 5. Grown regions which fail to meet the validation criteria. Pixels of the goal-box and ball regions are outside the expected region, or the field region does not cover a required percentage of the expected region.

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, we deploy three separate sets of thresholds, 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 with the help of the validation criteria outlined previously in section 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, we decrease the threshold by a small random amount. If the region is too small, we increase the threshold, and try to grow a region at the same point. We continue

increasing the threshold until the region is successfully grown, or grows outside the expected region.

6 Results

The method we present in this paper is able to adapt to changes in illumination in a few seconds. We tested the method on our robots 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 we recorded how long it took to adapt to an abrupt change in illumination. The scene we processed 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.6. 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 we can see, 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 our 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. 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 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.

Fig.7 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.

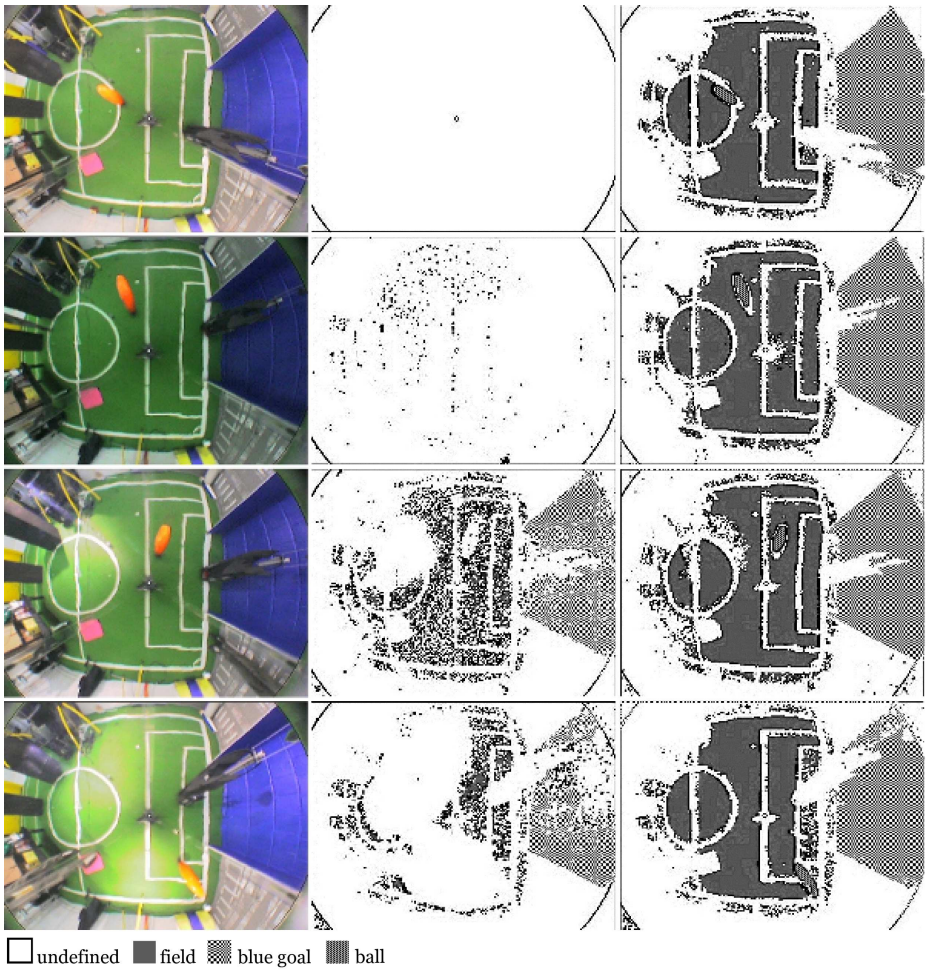

Fig. 6. Row 1 to 4 (counting from top to bottom): Original images on the left, static classifications from previous illumination setting in the middle, and the result of the automatic classifications on the right. The CPU-time it took the method to produce the automatic classifications (right), starting 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.

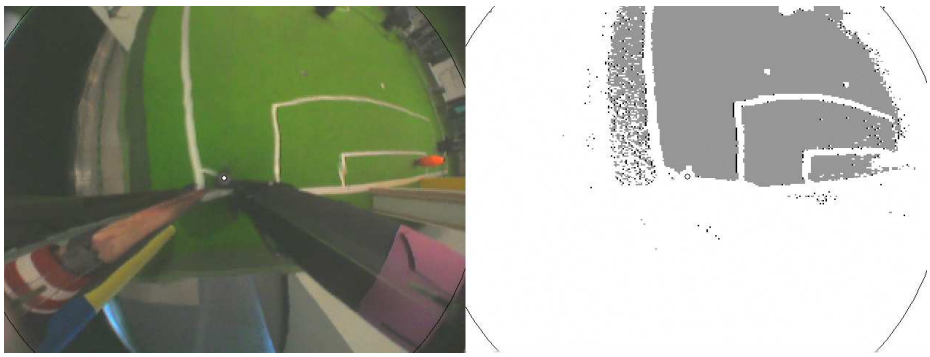

Fig. 7. The classification (on the right) was obtained after a few second with no previous classification. Here the ball and the goals are too far away to be calibrated so only the field can be calibrated. The color code is: white = unclassified, gray = field.

7 Future Work and Summary

The method encounters certain problems 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 are indistinguishable from the ones appearing 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 as well. 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. A possible solution would be to mark a color as not belonging to a class if it occurs in an unexpected region. Another approach used in [9], would be to incorporate a color decay factor.

A method for tuning the camera-parameters is presented in [8], and could be combined with our method 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 our automatic calibration.

In this paper we presented a method that can be used for automatic color calibration of autonomous soccer playing robots. 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 will be used by our robots at RoboCup 2005 in Osaka.

References

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.