A Real-Time Auto-Adjusting Vision System for Robotic Soccer*

Matthias Jüngel, Jan Hoffmann, and Martin Lötzsch

Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin
Rudower Chaussee 25, 12489 Berlin, Germany
{juengel,jan.hoffmann,loetzsch}@informatik.hu-berlin.de

Abstract. This paper presents a real-time approach for object recognition in robotic soccer. The vision system does not need any calibration and adapts to changing lighting conditions during run time. The adaptation is based on statistics which are computed when recognizing objects and leads to a segmentation of the color space to different color classes. Based on attention, scan lines are distributed over the image ensuring that all objects of interest intersect with the number of lines necessary for recognition. The object recognition checks the scan lines for characteristic edges and for typical groupings of color classes to find and classify points on the outlines of objects. These points are used to calculate size and position of the objects in the image. Experiments on Sony's four-legged robot Aibo show that the method is able to recognize and distinguish objects under a wide range of different lighting conditions.

1 Introduction

The RoboCup real robot soccer leagues (middle-size, small-size, Sony four-legged league) all take place in a color coded environment. The method described in this paper was implemented for the Sony four-legged league. The objects the robot needs to see are two-colored flags for localization (pink and either yellow, green, or sky-blue), two goals (yellow and sky-blue), the ball (orange), the robots of the two teams (wearing red and blue tricots), and the white field lines (center circle, middle line, penalty lines).

A very common preprocessing step for vision-based object recognition in color coded scenarios is color segmentation using color tables, e. g. [2, 7]. Such methods directly map colors to color classes on a pixel by pixel basis, which has some crucial drawbacks. On one hand, the color mapping has to be adapted when the lighting conditions change, on the other hand, the mapping results in a loss of information, because the membership of a pixel in a certain class is a yes/no decision, ignoring the influences of the surrounding pixels. Some researchers try to overcome these limitations [4], but the solutions are too slow to work under real-time conditions on a robot with limited computational power.

* The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 214–225, 2004.

A method for off-line self-calibration on sample images that first filters the images to improve color invariance and then applies k-means clustering for the adaptation of color classes is described in [6]. Lighting conditions can differ depending on the position on the field or the situation (the ball can be in the shadow of a player, the carpet can appear in different shades when seen from different viewing angles). In this paper a method is presented that adapts color classes during run time, so it is not necessary to perform any kind of pre-run calibration.

Although it is not necessary to use the full resolution of images to detect objects, it helps in determining their sizes and positions precisely. A very fast method for object finding which uses the concept of perspective view is suggested in [5]. Several additional methods restricting the number of pixels being processed during object finding are presented in this paper.

2 Image Processing

The key ideas of the image processing method presented in this paper are that speed can be achieved by avoiding to process all pixels of an image, and independence of the lighting conditions can be reached by focusing on contrast patterns in three different color channels and auto-adapting color classification.

2.1 Guiding Attention

Usually objects in the image are larger than a single pixel. Thus for feature extraction, a high resolution is only needed for small objects. The performance and robustness of image processing can be improved by guiding more attention to areas within the image where small objects are expected. Instead of processing all pixels in the image, a grid of scan lines is used to reduce the number of pixels processed. Different types of additional information about the image and the environment can be incorporated:

1. Image sequences: a robot usually processes a continuous sequence of images. Low level information (e.g. color clusters) gained in the previous image can be used to speed up object detection (e.g. a ball will not have moved too far from one image frame to the next, therefore it is in most cases beneficial to start searching for the ball in the area of the image where it was previously detected).
2. Iterative processing: a part of the image is processed and the information gained is used in the further processing of the image. For example, the image can first be scanned for more prominent features; once these are found, their location in the image can hint to the whereabouts of other features.
3. Other Sensors: readings from other sensors (such as distance or tilt sensors) can be analyzed and used to guide visual attention. For example, far away objects may or may not be of importance and therefore only parts of the image need to be scanned.

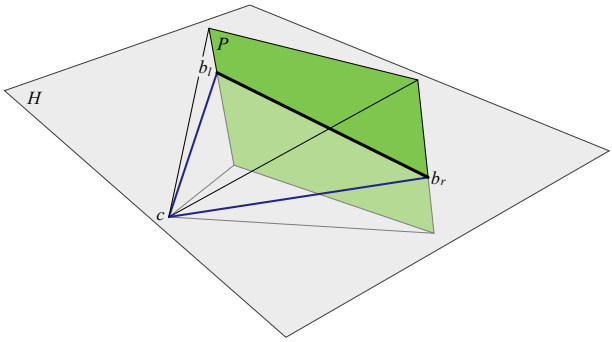

Fig. 1. *Calculating the horizon.* The horizon is the line of intersection of the projection plane P and the plane parallel to the ground H. $b_{l/r}$ is the intersection of the horizon with the left and the right border of the image, c is the focal point of the camera.

4. Knowledge about the environment: domain specific knowledge can be used to simplify image processing. Heuristics can be derived from this, e. g. in the RoboCup domain, robots and the ball are in contact with the field at all times. Therefore the larger parts of these objects will be below the horizon in the image.
5. Explicit feedback of (high level) world model information: The robot's knowledge (or hypothesis) of where it is in the environment can be used to simplify the search for relevant features. Only areas in the image that, according to the hypothesis, contain information are being scanned.

2.2 Arranging Scan Lines

First the position of the horizon in the image is calculated from the rotation of the head and the rotation of the body. The roll and tilt of the body are estimated from the readings of the robot's acceleration sensors (indicating the direction of the gravity), while the rotation of the head is determined from the angles of the three head joints (tilt, pan and roll). Then the areas below and above the horizon are scanned. For each of the areas an optimized grid layout is used that assures that relevant features cannot escape detection.

Calculating the Horizon. The horizon is the line of intersection between the two planes P and H where

P is the projection plane of the camera, and H is the plane parallel to the ground through the origin c of the camera coordinate system. To calculate the horizon line pixels $\overrightarrow{b}_{l/r}$ on the left/right border of the image and points $\overrightarrow{h}$ on the horizon H are examined, see Fig. 1:

$$\overrightarrow{b}_{l/r} = \begin{pmatrix} \frac{s}{\tan\alpha} \\ \pm s \\ z_{l/r} \end{pmatrix} \qquad \overrightarrow{h} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \tag{1}$$

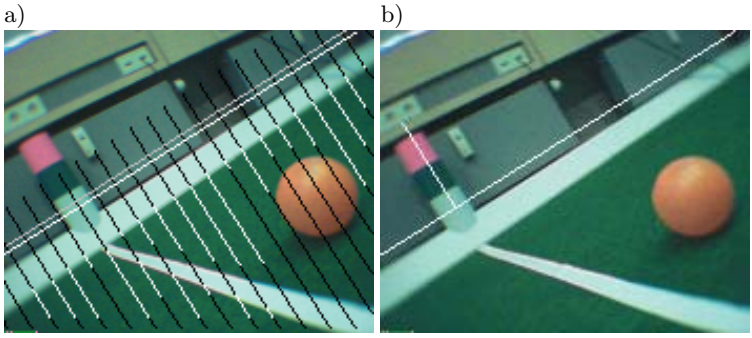

Fig. 2. *Scan Lines.* a) Gray horizontal line: horizon calculated from sensor readings, white horizontal line: horizon calculated from the white border. Vertical scan lines extend from the horizon to the lower half of the image. The white parts are used for auto-adaptation. b) A scan along the horizon determines the foot of the flag and its width. A second scan line vertical to the horizon through the center of the foot is used to determine the height of the flag.

with opening angle 2α and the screen resolution $2s$. Pixels that are both on the horizon and on the border of the image satisfy

$$R \cdot \overrightarrow{h} = \overrightarrow{b}_{l/r} \tag{2}$$

with R being the rotation matrix of the camera. This can easily be solved for the pixel coordinates of the horizon in the camera image:

$$z_{l/r} = \frac{\mp r_{32} s - r_{31} s \cdot \cot \alpha}{r_{33}} \tag{3}$$

with the entries r_{nm} of the rotation matrix.

Grid Lines below the Horizon. From the horizon, vertical scan lines (i.e. perpendicular to the horizon) extend into the lower half of the image, see Fig. 2a). The lines are evenly spaced. The distance of the lines depends on the expected distance to the ball. It is chosen such that at least two lines will intersect with the ball. Since the ball is the smallest object encountered on a RoboCup field, all other objects (robots) will also be hit. In addition to the dynamic objects mentioned, some static features of the field are found below the horizon: field lines, lower regions of the goals, and field borders.

Scan Lines above the Horizon. Color coded flags are placed at the corners and at the side of the field above borders to offer additional help for localizing the robot. Scan lines to find the flags are positioned in three steps: First based on the robots position and the direction of the camera a prediction of the positions of all visible flags in the image is calculated and vertical scan lines are arranged through the predicted flags. If the predicted flags are not found in the image a second attempt to find the flags is started on a scan line parallel to the horizon. If

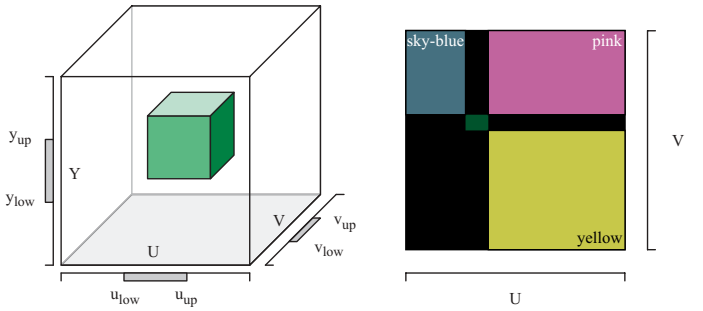

Fig. 3. *Left:* The reference color is a sub cube in the YUV color space. Other colors are defined relative to this cube. *Right:* The ranges for sky-blue, pink and yellow in the color cube.

this also fails the sensor readings are assumed to be disturbed by the influence of other robots and the position of the horizon is recalculated based on the position of the white border found in the image. If the foot of a flag is found on the scan line parallel to the horizon (based on sensors or image) a vertical scan line is positioned there, see Fig. 2b).

2.3 Color Classification and Adaptation

Each pixel at the grid lines is assigned to a color class. The color class depends on the position of the color of the pixel in the color cube. How the color cube is subdivided into color classes and how this subdivision is adapted automatically to changing lighting conditions is described in this section.

Color Classification. The camera of the Aibo provides YUV-images. In YUV, the luminosity of a pixel is stored in the Y channel, its color information in the chrominance channels U and V. To classify the color a very simple model is used. Within the YUV color cube, a reference color is defined. In the soccer application this reference color is the green of the carpet. This color is defined by upper and lower bounds in the three dimensions, thus producing a sub cube. Other colors are defined in relation to this cube. For example, *sky-blue* is defined as having a V value greater than the upper bound of the reference cube and a U value lower than the lower bound of the reference cube. Other colors are defined in a similar fashion. Only a limited number of colors is classified. Colors not necessary for object recognition are omitted, see Fig. 3.

As this method is so simple it is not possible to separate colors that are close to each other in color space, e.g. yellow and orange. Such colors are grouped into one color class. This disadvantage is accepted due to the simplicity of auto-adaptation of the reference color and the segmentation of the color cube. Ambiguities that arise from the bounded number of distinguishable colors are resolved by using additional information from the image and the environment (see sections 2.4 and 2.5).

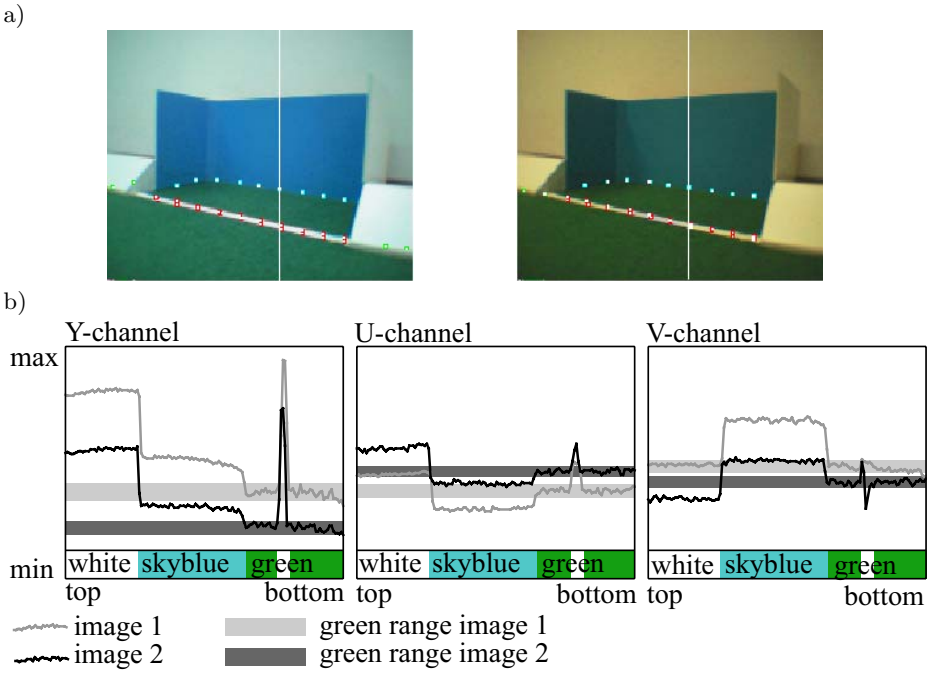

Fig. 4. a) Two images of the same scene recorded under different lighting conditions with a highlighted scan line and the recognized points on lines (border, field line, goal). Left: day light, right: artificial light. b) The intensities of the three channels along the scan line for both images. Left: image 1, right: image 2. The light and dark gray bars show the range of green after auto-adaptation for the left and the right image. For each image the values for sky-blue in the U channel are lower than the values for green. But the values for sky-blue in the U channel in image 2 are greater than the values for green in image 1 which shows the need for auto-adaptation.

Adaptation to Lighting Conditions. The effect of changing lighting conditions to the position of the colors in the color space is illustrated in Fig. 4. The positions of all colors move along the three axes and the intensities are amplified or weakened. But the relative position of the colors to each other remains. Thus it is sufficient to change position and dimensions of the reference cube with the lighting conditions.

To update the reference cube, from each image a set of green pixels is extracted. This extraction is not based on the previous reference cube but on a different heuristic: The changes in the YUV channels on a scan lines have certain characteristic properties which are used to identify pixels that are considered to be green. A scan line starting near the horizon will show a characteristic drop in the luminosity channel Y where the color changes from the white of the border to the green of the field. Starting from this point, all pixels on the rest of the scan line are considered to be green. If there is a colored object on the field,

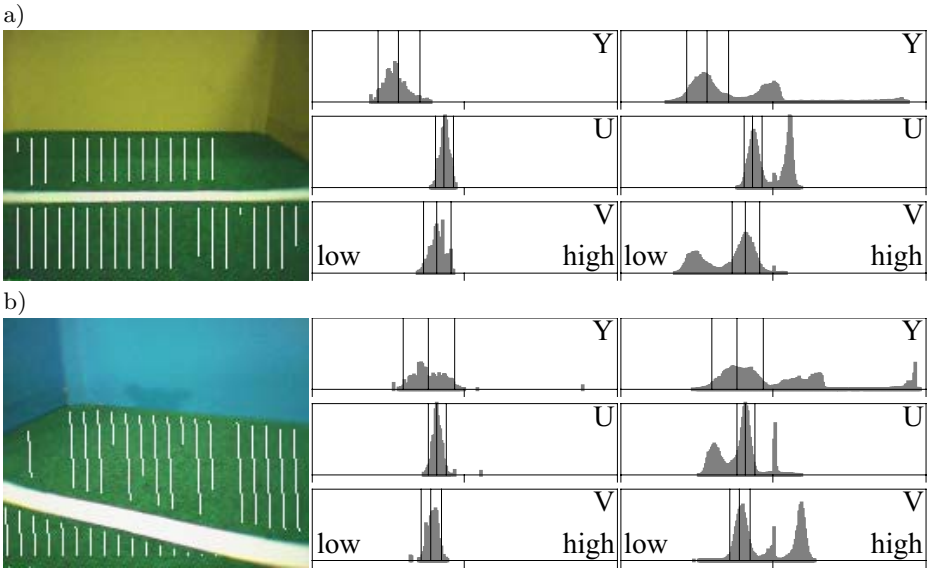

Fig. 5. Left: Image of a yellow (a) and a sky-blue (b) goal. The highlighted pixels have been selected by the heuristic for green extraction. Center: Frequency distribution of the intensities of the extracted pixels. The vertical lines show the 98% ranges. Right: Frequency distribution of the intensities of each pixel in the image. The vertical lines show the 98% ranges obtained from the extracted pixels.

there is a change in one of the color channels and the rest of that scan line is omitted (see the white parts of the vertical scan lines in Fig. 2a). This condition is rather stringent to assure that no pixels are wrongly added to the set of pixels used for color-adaptation.

This method is very robust and simple. No classification of clusters in color space or the frequency distribution of intensities for all pixels is needed.

The reference cube is updated with every image with enough green in it. Most of the images taken during a RoboCup game satisfy this condition. The YUV values of the extracted green pixels are used to recalculate the reference cube such that the new cube contains almost all green pixels. A typical distribution of the intensities for each channel of the selected pixels is shown in Fig 5. Note that the distribution of green pixels cannot be assumed as Gaussian but is dependent on scene illumination, carpet texture, camera noise, etc. The lower and the upper bound of the cube in each channel are chosen such that they enclose the most frequent 98% of the intensities.

2.4 Edge Detection and Classification

The edge detection finds characteristic changes in the three color channels. The points in the image where edges are found are classified using two criteria: the three dimensional contrast patterns and the surrounding color classes.

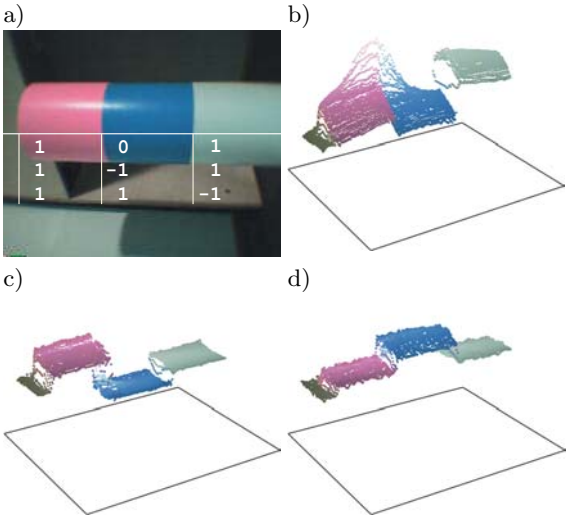

Fig. 6. *Scan Lines:* a) An image of a pink/sky-blue/white (from left) flag with a scan line and Y-U-V contrast patterns for each edge. b) the Y-channel of the image of the flag as height map. There is an up-edge (1) at the begin of the flag, the intensity increases near the highlight but there is no edge at the change from pink to sky-blue (0), there is an up-edge (1) from sky-blue to white. The first row of values in the contrast patterns is therefore (1,0,1). c, d) The height maps of the U- and V-channel visualize the second and the third row of values in the contrast patterns. Up-Down-Up (1,-1,1) and Up-Up-Down (1,1,-1).

Contrast Patterns. The contrast pattern for a pixel p_i on a scan line is defined as

$$C(p_i) = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

with

$$c_j = \begin{cases} 1 & , value_j(p_i) - value_j(p_{i-1}) > t_j \\ -1 & , value_j(p_i) - value_j(p_{i-1}) < -t_j \\ 0 & , else \end{cases}$$

for $1 \leq j \leq 3$, where $value_j(p_i)$, denotes the values of the color channels of the i^{th} pixel of the scan line and t_j denotes the thresholds for large changes in the three color channels. These thresholds are chosen once so that only edges and no noise inside objects are detected. The thresholds do not need to be recalibrated when lighting conditions change, see Fig. 6.

Each pixel p_i with $C(p_i) \neq (0, 0, 0)$ is assigned to one or more edge classes. For example $(0, -1, -1)$ is assigned to *yellow-black*, *yellow-green*, *orange-black*, and *orange-green*, because a significant drop in the U and V channel is an indication for one of these edges.

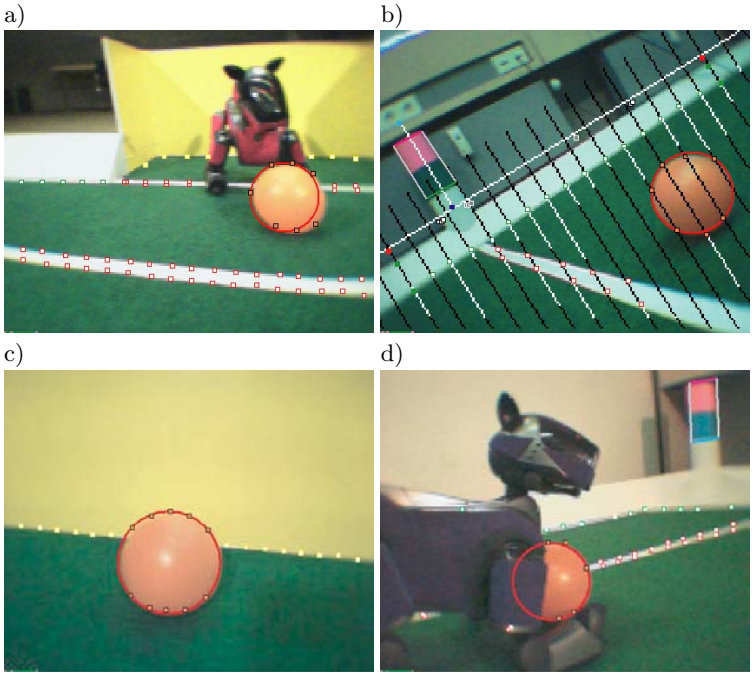

Fig. 7. *Scan Lines:* a) Three types of lines: field/border, field/goal, and field/line. Points on the outline of the ball. Circle calculated from the points. b) The ball on the field and a flag. c) The ball inside the yellow goal. d) A partly occluded ball and a flag.

Surrounding Color Classes. The color classes of the pixels surrounding the detected edges at the scan lines are taken into account to resolve ambiguities appearing during the classification of contrast patterns and to filter edges caused by noise.

For example the set of edge classes { *yellow-black*, *yellow-green*, *orange-black*, *orange-green* } can be reduced to { *yellow-green*, *orange-green* } if the preceding pixels belong to the color class orange/yellow and the following pixels are green.

The contrast pattern $(0, 1, 1)$ occurs at the top edge of the orange ball in a yellow goal but sometimes also if there is a reflection of a spotlight on a robot. Such contrast patterns are assigned to *yellow-orange* only if the preceding and the following color class is orange/yellow, see Fig. 7c).

2.5 Object Recognition

The object recognition combines several classified edges to objects and determines the size and position of the objects in the image. Different combining methods are applied to edges from different types of the scan lines.

Flags. To determine size and position of the flags in the image, the sequence of classified edges on each vertical grid line above the horizon is considered.

Each flag is characterized by a triple of classified edges containing the expected color changes at the upper edge of the flag, at the color change inside the flag, and at the lower edge of the flag. Thus a simple pattern matching algorithm is sufficient to detect all flags. Flags found in an image are represented by the four angles describing their bounding rectangle (top, bottom, left, and right edge) in a system of coordinates that is parallel to the field. The angles to the top and the bottom of the flag are determined from the positions of the first and the last edge of the matching edge pattern. To determine the angles to the left and the right edge of the flag, a new scan parallel to the horizon is started from the center of the section with higher contrast to the background, i. e. the pink one, in both directions. The first contrast pattern with a large decrease for the U-channel on this scan line marks the "end" of the flag and provides a point on a vertical edge.

Goals and Field Lines. Four different types of lines can be detected on the RoboCup field: edges between the sky-blue goal and the field, edges between the yellow goal and the field, edges between the border and the field, and edges between the field lines and the field, see Fig. 7a). The object detection has not to extract *lines* from the image, but *pixels on lines* instead. This approach is faster and more robust against misinterpretations, because lines are often partially hidden either by other robots or due to the limited opening angle of the camera. The *pixels on lines* are the input for a Monte-Carlo localization that is described in [8].

Ball. The edges at the bottom of the ball and at the bottom of the yellow goal have the same contrast patterns and the same color classes above and below the edges and thus are in the same edge class. But the top edge of the ball is unique because the top of the yellow goal is always about the horizon and does never intersect with the scan lines. Thus bottom ball edges are only accepted if there is a top ball edge above. As described in section 2.4 even a ball in a yellow goal can be detected, although yellow and orange are in the same color class.

The points found on the border of the ball are measured values and thus will not always lie on one and the same circle. First for each triple of points the resulting circle is calculated. In a second step each of the circles obtains a vote from each pixel having a distance of less than 3 pixels to the circle. The circle with most votes is assumed to be the border of the ball. This method assures that even partly occluded balls are detected, see Fig. 7.

3 Experimental Results

3.1 Lighting Independence

The approach described in this paper was tested in robot soccer games. The robots were able to localize themselves on the field and to handle the ball in an accurate manner. Approximately 80% of all taken images could be used to re-calibrate the color classification. Independence of lighting conditions was tested

by manually changing the white balance modes of the camera (indoor, outdoor, fluorescent light) and changing the lighting conditions itself. The objects on the first image taken under new conditions are not detected as auto-adaptation is always done for the next image. This does not matter, because each of the 25 images per second provided by the robot is processed. If enough green is in the first image with new conditions the objects in the second image are recognized properly. The need for auto-adaptation was shown by changing lighting conditions while keeping old color calibration which lead to misclassifications. The system fails when there is very little contrast in the image. This is the case in bright daylight (overexposed camera images) and under very little light (underexposure). In both cases colors become indistinguishable even for the human observer. If the scenario is lighted up with a very yellow light and the white balance of the camera is set to fluorescent light mode, the orange of the white of the borders and the yellow of the goal become indistinguishable, too. Under conditions similar to those at competition sites, the selected white balance mode had no influence on the quality of object recognition.

3.2 Performance

On an Aibo which is equipped with a 400 MHz processor the whole image processing (calculating and scanning the lines, finding edges, classifying colors and recognizing object) takes 7 ms if the grid lines are dense and 4 ms if the grid lines are wide spaced (ball is expected near the robot). There is no difference in running time if the auto-adaptation is switched off. Thus the approach is faster than classifying the color of each pixel of the image and then finding regions as described in [2]. It is as fast as the method that was used by the GermanTeam for the RoboCup 2002 competitions in Fukuoka that is described in [1, 3] and which requires manual color calibration.

4 Conclusion

This paper presents a real-time auto-adjusting vision system for robotic soccer. The vision system extracts features without processing whole images by guiding attention to the areas of high interest. Independence of lighting conditions is reached by an auto-adaptation of color classes and a matching of contrast patterns. The object recognition employs environmental constraints to determine size and position of the objects. This results in a fast, auto-calibrating, and precise feature extraction.

Acknowledgments

The authors would like to thank Hans-Dieter Burkhard, Joscha Bach, Uwe Düffert and Thomas Röfer for useful discussions.

References

1. J. Bach and M. Jüngel. Using pattern matching on a flexible, horizon-aligned grid for robotic vision. *Concurrency, Specification and Programming - CSP'2002*, 1:11–19, 2002.
2. J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061–2066, 2000.
3. U. Düffert, M. Jüngel, T. Laue, M. Lötzsch, M. Risler, and T. Röfer. GermanTeam 2002. In *RoboCup 2002 Robot Soccer World Cup VI, G. Kaminka, P. Lima, R. Rojas (Eds.)*, Lecture Notes in Computer Science, 2003. to appear. more detailed in http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf.
4. R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Towards RoboCup without color labeling. *RoboCup International Symposium 2002*, 2002.
5. M. Jamzad, B. Sadjad, V. Mirrokni, M. Kazemi, H. Chitsaz, A. Heydarnoori, M. Hajiaghai, and E. Chiniforooshan. A fast vision system for middle size robots in RoboCup. In *RoboCup 2001 Robot Soccer World Cup V, A. Birk, S. Coradeschi, S. Tadokoro (Eds.)*, number 2377 in Lecture Notes in Artificial Intelligence, pages 71–80, 2002.
6. G. Mayer, H. Utz, and G. Kraetzschmar. Towards autonomous vision self-calibration for soccer robots. In *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2002.
7. F. K. H. Quek. An algorithm for the rapid computation of boundaries of run length encoded regions. *Pattern Recognition Journal*, 33:1637–1649, 2000.
8. T. Röfer and M. Jüngel. Vision-based fast and reactive Monte-Carlo localization. *to appear: IEEE International Conference on Robotics and Automation*, 2003.