# Using Layered Color Precision for a Self-Calibrating Vision System*

Matthias Jüngel

Institut für Informatik, LFG Künstliche Intelligenz,
Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany
`http://www.aiboteamhumboldt.com`

**Abstract.** This paper presents a vision system for robotic soccer which was tested on Sony's four legged robot Aibo. The input for the vision system are images of the camera and the sensor readings of the robot's head joints, the output are the positions of all recognized objects in relation to the robot. The object recognition is based on the colors of the objects and uses a color look-up table. The vision system creates the color look-up table on its own during a soccer game. Thus no pre-run calibration is needed and the robot can cope with inhomogeneous or changing light on the soccer field. It is shown, how different layers of color representation can be used to refine the results of color classification. However, the self-calibrated color look-up table is not as accurate as a hand-made. Together with the introduced object recognition which is very robust relating to the quality of the color table, the self-calibrating vision works very well. This robustness is achieved using the detection of edges on scan lines.

## 1 Introduction

The vision system that is described in this paper was implemented for the Sony four-legged league. This and the other RoboCup real robot leagues (middle-size, small-size) take place in a color coded environment. The robots play with an orange ball on a green ground and have to kick to yellow and sky-blue goals. At the corners of the field there are two-colored poles for localization. The environment for the games, as the size of the field and the colors of the objects, is well defined. However, the lighting conditions are not known in advance and might change during the competition and even during games.

This paper presents a vision system that uses a method for object recognition that is very robust relating to the quality of the color calibration. This is needed as the color calibration is not done by a human expert before the game, but during the game by the vision system itself.

The task of vision-based object recognition in color coded scenarios often is divided in several subtasks. The first step is the *segmentation* which assigns a

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

color class (orange, green, etc.) to each color (defined by three intensities) of the color space. How this can be done using thresholds or color look-up tables is described in [1].The thresholds and look-up tables usually are created by hand with the assistance of semi-automated tools. More sophisticated tools do an off-line self-calibration on sample images [5]. The general problem of all segmentation methods is, that the color calibration is only valid as long as the lighting conditions do not change. The effect of such a change is described in [6]. A method for autonomous dynamic color calibration is described in [2]. The drawback of this method is that it requires for special actions (walk and head motions) to be done by the robot in order to recalibrate. Sometimes there is a *color space transformation*, before the segmentation is done [4, 8]. In the *clustering* step connected regions of the same color class are identified [7, 1]. The final step is the *classification* which extracts objects and belonging properties like size and position from the set of clusters. This step is domain specific and not standardized. Besides the color based approach there are methods that concentrate on the boundaries of objects. A very robust method that finds image boundaries is described in [3] but this solution is too slow to work under real-time conditions on a robot with limited computational power.

This paper describes a vision system that is based on an analysis of the color of scan lines segments. How these segments are created and examined is described in section 2. The method of calibration-free color classification that is applied to the average colors of the segments is shown in section 3. Section 4 shows some of the experiments that were done to evaluate the vision system.

## 2 Segment Based Image Processing

The vision system subdivides the scan lines into several segments that are separated by edges. The segments are the starting point for all further image processing. Section 2.1 shows, how the scan lines are placed and how they are split to segments. The object recognition that is described in section 2.2 is based on the classification of the colors of the segments. For this classification an auto-calibrated color table is used (cf. section 3).

### 2.1 Distribution and Segmentation of Scan Lines

To detect objects in an image and measure their sizes, a set of scan lines is used. The scan lines are horizon-aligned and distributed such that the density of scan lines is higher in areas where more attention is needed to detect all objects. The horizon is used to align vertical scan lines which are perpendicular to the horizon. The lines run from 10 degrees above the horizon to the bottom end of the image and have a spacing of 3 degrees.

The segments of a scan line are found using a simple edge detection algorithm. The algorithm finds the first and the last pixel of an edge. An edge is a sequence of so called *edgels*. Edgels are the pixels of scan lines where there is a sharp

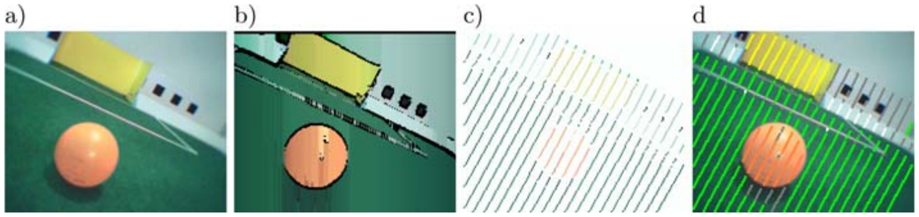

**Fig. 1.** Segments. a) The original image. b) For each vertical line of the image all edges on the line were detected. The color of the segments between the edges is the average color of the pixels of the respective segment. c) Same as b, but with horizon aligned scan lines that are subdivided into segments, each segment has the average color of all belonging pixels. d) The segments shown in c displayed with the color class that is assigned to the average color of each segment

variation of the intensity of at least one color channel. The segments are the parts between the edges of each scan line.

While the scan lines are scanned for edges, for each segment the first and the last point and the average intensity over all pixels for each color channel are determined. Fig. 1 shows such segments.

## 2.2 Finding Objects by Analyzing Segments of Scan Lines

**Finding Points on Lines.** On a RoboCup field there are different types of lines: edges between a goal and the field, edges between the border and the field, and edges between the field lines and the field.

*The Border of the Field and Field Lines.* To detect the border of the field and the field lines, all segments whose average intensities are classified as white are analyzed. The begin and the end point must be below the horizon, and the segment below must be green. To distinguish between the field lines and the field border, the expected size of a field line is examined. This size depends on the distance from the robot to the field line. Thus the distance to the end point of the white segment is calculated based on the rotation of the head and the position of the point in the image. If the length of the segment is more than fivefold the expected size of a field line, it is classified as a field border. To be classified as a field line, the segment's size must not be larger than twice the expected size of a field line. The figures 2b and 2c show how the vision system can distinguish between a field line and the field border.

*Goals.* To recognize the points at the bottom of the goals all sky-blue and yellow segments are analyzed. The end points must be below the horizon and the begin points must be above the horizon or lie on the image border. The length of the segment must be at least 5 pixels and the segment below must be green. Figure 2a shows such points.

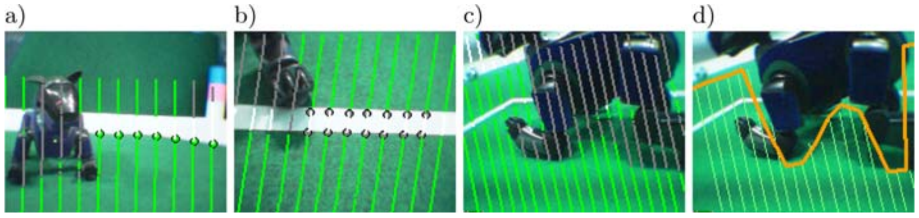

**Fig. 2.** Recognition of the border, the field lines, and obstacles. a + b) Points at the field border and at a field line. The border and the line have a similar size in the image and are around the same position in the image. They are distinguished based on the different direction of view of the head. c) The segments that are classified as green. d) Green Lines: The obstacles percept is the combination of adjacent green segments. Orange Line: The resulting Obstacles Model

**Finding Obstacles.** The obstacles percept is a set of lines on the ground that represents the free space around the robot. Each line is described by a *near point* and a *far point* on the ground, relative to the robot. The lines describe segments of green lines in the image projected to the ground. In addition, for each far point a marking describes whether the corresponding point in the image lies on the border of the image or not.

To generate this percept for each scan line the bottom most green segment is determined. If this green segment meets the bottom of the image, the begin and the end point of the segment are transformed to coordinates relative to the robot and written to the obstacles percept; else or if there is no green on that scan line, the point at the bottom of the line is transformed to coordinates relative to the robot and the near and the far point of the percept are identical.

Small gaps between two green segments of a scan line are ignored to assure that field lines are not misinterpreted as obstacles. In such a case two neighboring green segments are concatenated. The size limit for such gaps is $4 \cdot width_{fieldline}$ where $width_{fieldline}$ is the expected width of a field line in the image depending on the camera rotation and the position in the image.

## Finding the Ball

*Finding Points on the Border of the Ball.* Usually there should be at most one orange segment per scan line. However, if there is a highlight or a shadow on the ball, this might result in a higher number of orange segments per scan line that represent the ball. In such a case all orange segments that lie on the same scan line and are part of a sequence of subsequent segments are grouped to one large segment. Then the begin and the end of the bottom most orange (grouped) segment on each scan line is added to the list of ball points if there is a green segment below.

*Additional Scan Lines.* If the set of ball points contains at least two points, three additional scan lines are calculated based on the smallest rectangle that

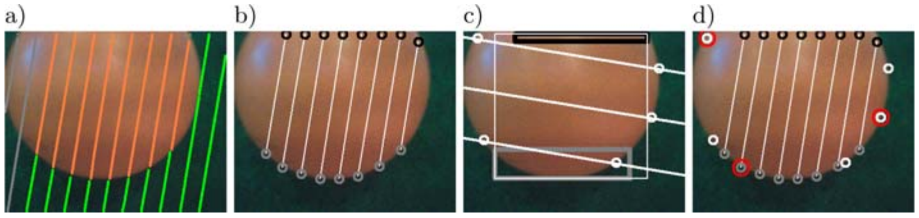

**Fig. 3.** Ball Recognition: a) The image of an ball and the color classified segments. b) White lines: The orange segments. Black circles: All end points of the orange segments, that are close to the border of the image. Gray circles: All meeting points of an orange and a green segment. c) White box: The bounding box of all black and gray points in b. White lines: Lines parallel to the horizon; the center line goes through the center of the white box. The spacing between the lines depends on the extension of the white box. White circles: Transitions from orange to green on the white scan lines. d) Red circles: The largest triangle amongst the gray and the white circles

contains all points of this set. The new scan lines are parallel to the horizon and evenly overlap the bounding rectangle. This distribution of the additional scan lines assures that more points on the border of the ball can be found on the left and the right side. These new scan lines are divided into segments in the same way as the long vertical scan lines. For each edge that separates an orange and a green segment a point is added to the set of ball points.

*Calculation of the Circle.* To calculate a circle that describes the ball first all points from the set of ball points that lie on the image border are removed. From the remaining points two points with the highest distance are chosen. Then the point with the highest distance to these points is selected. If these three points do not lie on one and the same straight line they are used to calculate the circle that describes the ball.

**Finding Goals.** Besides the points at the bottom of the goal, the bearings to the four sides of the goal are determined. Additionally the angle to the left and the right side of the larger part of the goal are determined.

The segments that were used to generate the points at the bottom of the goals are combined to clusters. Two segments of two different scan lines belong to the same cluster if there is no scan line without a goal colored segment between them.

Additional horizontal scan lines determine the horizontal extensions of the clusters. All clusters that have white below are rejected because goals are not outside the border. All clusters that have pink below are rejected because they probably are a part of a landmark. All other clusters are combined leading to a bounding box that includes all clusters. Such a bounding box, generated by combining all valid clusters, is accepted as a goal if it fulfills the conditions: 1: The bottom is below the horizon. 2: The top is above the horizon or intersects with the image border. 3: The width is at least 1.5 times the height of the bounding box or the left or the right intersects with the image border.

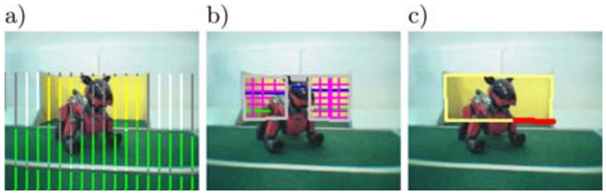

**Fig. 4.** Finding the goal. a) An image of the goal with color classified segments. b) The gray boxes show the vertical and horizontal extension of the clusters. The horizontal extension was measured using additional horizontal scan lines (the pink lines). c) Combination of the clusters. The red line marks the larger free part of the goal

For a robot that plays soccer the position of the angle and the distance of the goal are important for localization. To avoid to kick into the goalie the largest free part of the goal has to be determined. This is done by comparing the clusters that lead to the goal bounding box. If no cluster or the larger one intersects with the image border, the angles to the left and the right side of the larger cluster determine the best angle for a goal kick. If the smaller part intersects with the border, it can not be decided, whether this part extends outside the image and is the larger free part of the goal, or not. The vision system does not suggest a goal kick angle. Figure 4 shows how the goal is recognized.

## 3 Layered Color Precision

This section shows, how a color look-up table can be created and updated automatically during the robot plays soccer. The calibration system works on the segments of scan lines that are created for object recognition (cf. section 2). The output of the color calibration is the color look-up table that is used to classify the colors of the segments for object recognition.

### 3.1 Three Ways to Represent Color Classes

The vision system uses three layers of color precision. Each layer has its own method to assign colors to color classes.

The layer with the lowest precision, *layer 1*, represents the color classes $green_1$, $white_1$, and $black_1$. The colors $green_1$ and $white_1$ are represented by cuboids in the color space. The rest of the color space is $black_1$. Each cuboid is defined by 6 threshold (min and max for each color channel).

The next layer of color precision, *layer 2*, distinguishes between more colors. It uses a cuboid in the color space to define $green_2$ as a reference color. Relative to the reference color cuboids for the color classes $yellow_2$, $skyblue_2$, and $orange_2$ are defined. The colors $yellow_2$ and $orange_2$ can be overlapping in color space, which means that some colors can be classified as $yellow_2$ and $orange_2$.

The layer with the highest precision is *layer 3*, which uses a color look-up table to represent $white_3$, $green_3$, $yellow_3$, $skyblue_3$, and $orange_3$. This look-

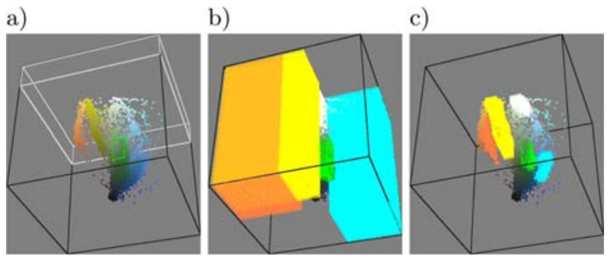

**Fig. 5.** Different layers of color precision. a) Layer1: Rough classification for green and white. The white and the green cuboid define the color classes. b) Layer2: Refined classification for green, other colors are classified based in the relation to green. The green cuboid is the reference color all other cuboids are defined in relation to the green cuboid. c) Layer3: Classification based on a color look-up table

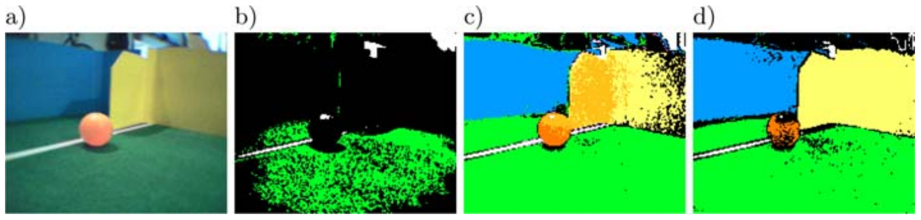

**Fig. 6.** Color classified images based on the three different layers of color representation. a) The original image. b) Layer1: Rough classification for green and white. c) Layer2: Refined classification for green, other colors are classified based on the relation to green. d) Layer3: Classification based on a color look-up table

up table is used to classify the colors of the segments for object recognition as described in section 2.2.

Figure 5 shows all three ways to represent an assignment from colors to color classes. The color classification of an image is shown in figure 6 for each of the three forms of representation.

The output of the color calibration system is the *layer 3* color representation, the color look-up table. This is created based on segments that can be identified to belong to an object with a known color for sure. For example the average colors of all segments that belong to the ball are used to calibrate orange. To extract such segments the *layer 2*-based colors of the segments and knowledge about the environment is used. The reference color of the *layer 2* color representation is determined using the *layer 1*-based colors of the segments together with knowledge about the environment. The *layer 1* color representation is constructed based on statistics over the last images. The following subsections describe, how this process of *color-refinement* works in detail.

### 3.2 Calibrating Layer 3 Colors

The vision system identifies segments that belong to a certain object for sure and adds the average colors of the segments to the color class of the object. The segments are identified based on the layer 1 and layer 2 color classes and on spatial constraints for the position of the segments in the image.

*White.* White is the color of the border and the field lines. Segments that are $white_1$ are possibly a part of the field border. But in some cases parts of the ball or the ground or field lines can be classified as $white_1$. Thus only the $white_1$ segments that fulfill the following conditions are used to calibrate $white_3$: 1: The segment is not $green_2$. 2: The begin of the segment is below the horizon. 3: The length of the segment is at least 4 pixels. 4: The length of the segment is at least 5 times the expected size of a field line at the position of the center of the segment. (This filters noise) For each such segment the color look-up table is expanded such that its average color is classified as $white_3$.

*Colors of the Goals.* The color classification based on a reference color from *layer 2* can distinguish $orange_2$, $yellow_2$, and $skyblue_2$. But the color precision is not very high. The colors yellow and orange overlap and the rim of the field lines often is classified as yellow or sky-blue. If the white of the outer border is a little bit bluish it also might be classified as sky-blue. Thus only the segments that are $yellow_2$ or $skyblue_2$ and fulfill these conditions are identified as a part of goal: 1: The begin of the segment is above the horizon. 2: The end of the segment is below the horizon. 3: The length of the segment is at least 10. This filters all segments that are part of a ball or an other robot or caused by the rim of a scan line. Because of the last condition far distant and thus small goals can not be used for calibrating the goal colors. The average colors of all these goal segments are added to the corresponding color class in the *layer 3* color class representation.

*Orange.* To calibrate $orange_3$, segments of scan lines that lie on a ball are used. Not all segments that are $orange_2$ are a part of a ball. In some cases a goal or a small part of a red robot is classified as $orange_2$. Thus only the $orange_2$ segments that fulfill the following conditions are used to calibrate $orange_3$: 1: The begin of the segment is below the horizon. 2: There is a green segment below. 3: The length of the segment is at most 1.5 times the expected size of a ball at the position of the center of the segment. 4: The length of the segment is at least 0.2 times the expected size of a ball at the position of the center of the segment if the segment does not intersect with the image border. This method filters all segments that do not belong to a ball, and unfortunately even a lot of ball segments. However, in this way enough segments are determined that belong to a ball for sure. The average color of each such segment is added to the color look-up table for $orange_3$.

*Green.* To calibrate $green_3$ all segments that are classified as $green_1$ are examined. To be used for calibration of $green_3$ the segments have to fulfill the

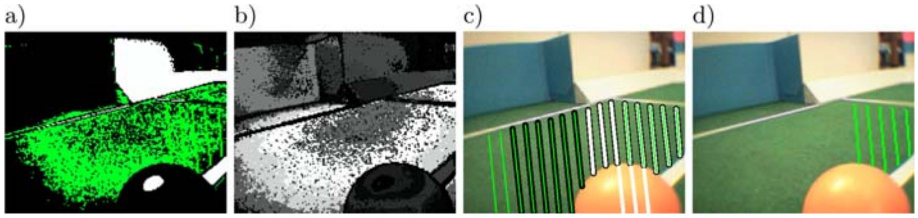

**Fig. 7.** Calibration of green. a) The layer 1 color classes of the image. b) The frequent colors displayed bright. c) Black: All long scan lines with an average color classified as $green_1$ (cf. the green pixels in a) White: All long scan lines below the horizon and below the field border. Green: All long scan lines with a frequent color as average color (cf. the bright pixels in b). d) The scan lines that are used to calibrate green

following conditions: 1: The average color must be one of the most frequent colors of the current image. (As the green of the ground is the most frequent color in most images, this helps to filter all other objects) 2: The length must be at least 30 pixels. (This filters small segments that can be caused by other robots, the referee or unexpected objects on the field) 3: The field must be found above. (This ensures that the area outside the field is not used for calibration, in the case the robot looks over the image border) 6: The begin point must be below the horizon. (No ground segments are above the horizon) 5: The difference between the maximal and the minimal intensity of the v-channel must be less than 20. (If the border of the ball intersects with a scan line in a very small angle, it is possible that no edge is detected. In this case a segment is created that has a high difference between the minimal and the maximal intensity of the v-channel as green and orange differ in that channel very much.)

For each segment that fulfills these conditions the color look-up table is expanded such that $green_3$ includes the maximal and the minimal intensities for all 3 channels of the segment. Figure 7 shows which of these conditions are fulfilled by which scan lines in an example image.

## 3.3 Calibrating Layer 2 Colors

In layer 2 the colors are defined by cuboids in relation to the reference color $green_2$. Green is a refined version of the more rough $green_1$ from layer 1. The other colors in layer 2 do not directly depend on the classification from layer 1 but are approximated using $green_2$ as a reference color.

*Green.* To define the thresholds for the cuboid of the reference color $green_2$ all segments whose average color is classified as $green_1$ are used that fulfill the conditions for green segments that are used to calibrate $green_3$ (cf. 3.2). For each segment that fulfills these conditions the cuboid for the reference color $green_2$ is expanded such that it includes the maximal and the minimal intensities for all 3 channels of the segment. The other colors are defined in relation to this reference color.

*Yellow.* The yellow of the goal has more *redness* than the green of the field and less *blueness*. The brightness for yellow is not restricted. This leads to the thresholds $y_{min} = 0, y_{max} = 255 \mid u_{min} = 0, u_{max} = u^{green}_{min} \mid v_{min} = v^{green}_{max}, v_{max} = 255$ for $yellow_2$ relative to the thresholds for $green_2$, where $u^{green}_{min}$ is the lower threshold for the *blueness* of green and $v^{green}_{max}$ is the upper threshold for the *redness* of green.

*Orange.* Orange has more red than yellow and in the YUV color space more blue than yellow. The brightness for yellow is not restricted. This leads to the thresholds $y_{min} = 0, y_{max} = 255 \mid u_{min} = 0, u_{max} = u^{green}_{average} \mid v_{min} = v^{green}_{max}, v_{max} = 255$ for $orange_2$ relative to the thresholds for $green_2$. This means, that the yellow of the goal has more *redness* than the green of the field and less *blueness*. The threshold $y_{min}$ means that sky-blue is not much darker than green.

*Sky-blue.* These are the thresholds for $skyblue_2$ relative to the thresholds for $green_2$: $y_{min} = 0, y_{max} = 255 \mid u_{min} = u^{green}_{max}, u_{max} = 255 \mid v_{min} = 0, v_{max} = v^{green}_{average}$ This means that the sky-blue of the goal has more *blueness* than the green of the field and the same or less *redness*. The brightness for sky-blue is not restricted.

## 3.4 Calibrating Layer 1 Colors

To calibrate $green_1$ and $white_1$ for each channel the average intensity $\bar{i_c}$ of all scanned pixels over the last 10 images is calculated.

*White.* The minimal brightness for $white_1$ is defined in relation to the average brightness of the last images. It turned out that for dark images all pixels that have more than twice the average brightness can be defined as white. For brighter images a minimal value for white that lies between the average brightness and the maximal possible brightness is useful. Thus the thresholds for the $white_1$ cuboid are set to: $y_{min} = min\left(\bar{i_y} \cdot 2, \frac{(\bar{i_y}+255\cdot 2)}{3}\right)$ , $y_{max} = 255 \mid u_{min} = 0, u_{max} = 255 \mid v_{min} = 0, v_{max} = 255$ Therefore all colors that are brighter than the threshold $y_{min}$ are classified as $white_1$.

*Green.* The thresholds for $green_1$ are set to: $y_{min} = \bar{i_y} \cdot 0.8, y_{max} = \bar{i_y} \cdot 1.2 \mid u_{min} = \bar{i_u} \cdot 0.9, u_{max} = \bar{i_u} \cdot 1.1 \mid v_{min} = \bar{i_v} \cdot 0.9, v_{max} = \bar{i_v} \cdot 1.1$ Thus all colors that are similar to the average color are classified as $green_1$. The tolerance for the brightness channel $y$ is slightly larger than the tolerance for the color channels $u$ and $v$.

## 3.5 Adaptation to Changing Lighting Conditions

The method for color calibration as described up to here only assigns new colors to color classes or changes the assignment of a color from one color class to another one. Therefore colors that are misclassified once will be misclassified for ever, except they are assigned to another color class. Thus for each entry of the color look-up table (layer 3 color representation) the time when it was last updated is stored. If an entry was not updated for a longer time, the assignment to the color class is deleted. This time depends on the color (green is updated more often than orange). All colors that are used to calibrate $green_3$ are also used to detect, if there was a sudden change of the lighting conditions. If $green_3$ is calibrated, the average value of the segments used for calibration in the current image is compared to the average values used for calibration in the last images. If there is a large difference in at least one color channel, all three layers of color representation are deleted. Thus the vision system reacts very quickly if there is a sudden change of the lighting conditions. In this case the robot has to see at least one object of each color for a few frames to recalibrate all colors.

## 4 Experimental Results

A crucial parameter for the auto calibration is the reference color. To test if the reference color does not contain colors different from green obstacles, an obstacle avoiding behavior was executed. The robot was able to pass the soccer field without touching any arbitrary positioned obstacle. There was no performance difference compared to a manually calibrated vision system. An easy way to test if the automatic color calibration is successful is to have a look at the resulting color classified images. These images should be similar to images that are the result of a color classification using a hand-made color table. A hand-made color

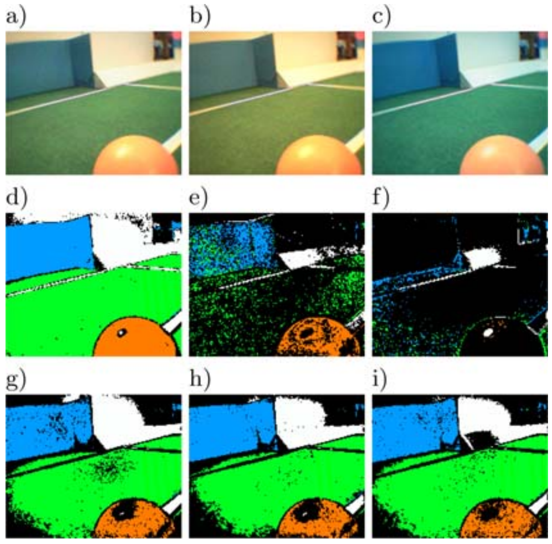

**Fig. 8.** a,b,c) Three images of the same scene, taken under different lighting conditions. d,e,f) The result of color classification based on a color table that was created by hand for image a g,h,i) The result of color classification based on a color table that was created automatically for each lighting condition

table that was created using sample images taken under certain lighting conditions usually is not usable under different lighting conditions (cf. Fig. 8a-f). The vision system described above was able to adapt the color table when the lighting conditions changed (cf. Fig. 8g-i). The system presented in this paper needs 20 ms to process an image of size 176x144 on an Aibo, which is equipped with a 400 MHz processor.

## 5 Conclusion

This paper presents a vision system for robotic soccer which needs no pre-run calibration. The independence of lighting conditions is reached by an auto-adaptation of color classes and an edge-based analysis of scan lines. The object recognition and the color calibration employ environmental constraints to determine size and position of the objects. The author would like to thank H.-D. Burkhard, Th. Röfer, and all members of the *Aibo Team Humboldt* for their support.

## References

1. 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, 2000.
2. N. B. Daniel Cameron. Knowledge-based autonomous dynamic colour calibration. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004. to appear.
3. R. Hanek, W. Schmitt, S. Buck, and M. Beetz. Fast image-based object localization in natural scenes. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002*, pages 116–122, 2002.
4. A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.
5. 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.
6. G. Mayer, H. Utz, and G. Kraetzschmar. Playing robot soccer under natural light: A case study. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004. to appear.
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. J.-C. Terrillon, H. Fukamachi, S. Akamatsu, and M. N. Shirazi. Comparative performance of different skin chrominance models and chrominance spaces for the automatic detection of human faces in color images. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, page 54ff, 2000.