Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

An Automatic Approach to Online Color Training in RoboCup Environments

Patrick Heinemann*, Frank Sehnke*, Felix Streichert* and Andreas Zell*
* Wilhelm-Schickard-Institute
Department of Computer Architecture
University of Tübingen
Email: {heinemann, sehnke, streichert, zell}@informatik.uni-tuebingen.de

***Abstract*— Many approaches for extracting landmarks and objects from a camera image based on their color coding were published in the RoboCup domain. They are quite sophisticated and tuned to the typical RoboCup scenario of constant bright lighting using a static subdivision of the color space into different color classes. However, such algorithms will soon be of limited use, as the future requirements of RoboCup include the possibility to play under changing and finally natural lighting. This paper presents an algorithm for automatic online color training, which is able to robustly adapt the mapping of colors to color classes onto different lighting situations online. Using the ACT algorithm a robot will be able to play a RoboCup match while the illumination of the field varies.**

I. Introduction

The extraction of landmarks and objects from a camera image is a crucial task for robots using vision as their main sensor. In many applications the color of such features is sufficient for their extraction. To further improve the robustness of the feature extraction, all objects and landmarks in RoboCup are of distinct colors that are easy to distinguish in terms of their representation in the color space. Although there were attempts to detect objects - mainly the ball - using form or texture [4],[12], the majority of the RoboCup teams uses algorithms exploiting these special colors to reduce the computational load. Most of these algorithms utilize a predefined subdivision of the three-dimensional color space (RGB, HSI, or YUV) into several color classes. In the Middle-Size-League (MSL) there are 6 color classes corresponding to the different objects: green for the floor, white for the field lines, black for the robots, orange for the ball, and blue and yellow for the goals. If all or part of the image pixels are transformed from the three-dimensional color space into these color classes the extraction of the different objects can be very efficient [6]). The approaches mainly differ concerning the subdivision of the color space. Bruce *et al.* use a rectangular subdivision with a minimum and maximum threshold for each color class [2]. Others, such as Bandlow *et al.* use a prism composed of an arbitrary base in the two color dimensions of the YUV color space and two thresholds in the intensity channel in order to achieve some independency of the lighting conditions [1]. However, with changing lighting, the colors do not only change their representation in two, but in all three dimensions of the color space (cf. [10]). Therefore, our team uses a color look-up table that maps each color to its corresponding class. With such a table it is possible to model an arbitrary subdivision of the color space that allows to include the changes of a color at different lighting conditions. For recent competitions this look-up table was trained by manually chosing pixels in one or more camera images. A sphere of colors centered at the color of a chosen pixel was then marked as one of the color classes. In order to model arbitrary subdivisions of the color space in acceptable time there was also the possiblity to subtract colors from a color class with the same mechanism. The robustness of this color segmentation shows that this was a very good approach even on fields where the lighting was moderately varying over the field. Nevertheless, even this approach will fail if the amount and the speed of changes in lighting conditions rises when playing at natural light. In addition, the time needed to manually train a color look-up table was still up to 5 minutes per robot. Thus, there is clearly a need for a fast and automatic training of a look-up table that dynamically maps the colors to different color classes with changing lighting.

Related work on this topic includes the semi-automatic self-calibration system for MSL robots presented in [9], that subdivides the color space automatically into circular clusters. However, the mapping of the clusters to their corresponding color class has to be decided by a supervisor and the clusters cannot be of arbitrary shape. Other methods used on robots of the four-legged league require special movements of the robot and the robot's head to train the color look-up table [3] or simplify the problem by training a color mapping for only three different illumination scenarios and by classifying the camera images into one of these scenarios [11]. A very promising method coming from this league is presented in [8]. This combined object detection and color training method retrains a color look-up table without an a-priori known subdivision of the color space. However, the three-layered training method would require too much computation time when applied to an image of our MSL robots (cf. section III).

In this paper we present a new algorithm to automatically train such a table for a RoboCup robot, using only knowledge of the field geometry. By incorporating the pose of the robot computed by our self-localization algorithm ([5]) this algorithm is able to constantly re-train the mapping according to the changing lighting conditions. By keeping the amount of training per cycle as low as possible, the algorithm can be

1-4244-0259-X/06/$20.00 ©2006 IEEE

processed 50 times a second on our RoboCup MSL robots. With all the other processes like self-localization and path planning running at the same cycle rate, our robot is able to cope with the highly dynamic RoboCup environment. Yet, the color training algorithm is still capable of adapting to sudden changes in illumination in only a few cycles. The remainder of the paper is organized as follows: the proposed algorithm is presented in detail in the following section. Section III contains a description of the hardware used for the experimental results presented in section IV, which emphasize the robustness of the algorithm concerning changes in lighting. The last section concludes this paper.

II. The Automatic Color Training Algorithm

The main idea of the automatic color training (ACT) algorithm is to exploit the knowledge about the well-structured environment to train a color look-up table automatically. Instead of a user specifying which pixels should be classified as green or white, this algorithm uses the pose of the robot from the self-localization and a model of its environment to compute the expected color class of the image pixels. For this, the field parameters (length, width, radius of center circle, etc.) and a mapping from pixel to two-dimensional world coordinates of the field are needed. While the field parameters for RoboCup tournaments are known, the mapping from pixel to world coordinates is trained automatically, too, using the field markings and a predefined pose of the robot ([7]).

As this paper does not address self-localization, these prerequisites are assumed to be known for the following. However, the use of self-localization for the automatic color training results in a mutual dependency, as there can be no self-localization without the extraction of color coded landmarks and there will be no color-based landmark detection without a pose estimation of the self-localization. Two features are used to overcome this mutual dependency. First, the image that is used for automatic training of the mapping from pixel to world coordinates is used for the training of an initial color look-up table, too. Second, the extraction of the green and the white color class is robust enough to cope with a sudden change in illumination as shown in section II-A. This enables the robust self-localization [5] to keep track of the correct pose of the robot until the other color classes are adapted.

With the color values of the pixels and the expected color class a look-up table can be trained automatically. As there will obviously be errors in the computation of the expected color class, e.g. moving black robots on the field where the static model expects green floor, the algorithm does not make a direct mapping from color values to the computed color classes. Instead, ACT tracks clusters of the color classes in the color space with a mean value and standard deviation, to filter out such errors. Only colors of pixels that fit into a sphere in the color space centered in the mean value of a color class with radius equal to a multiple of the standard deviation, are added to the look-up table. In addition, colors of pixels that correspond to coordinates outside of the playable field are removed from the look-up table, as these pixels would otherwise decrease the robustness of the following algorithms. This removal is only done for color values outside a sphere around the mean value with a radius equal to a multiple of the standard deviation, to keep a minimum configuration for each color class. The multiples of the standard deviation for addition and removal of colors are chosen such that there is a hysteresis for a stable optimization of the color look-up table.

A cycle of the ACT algorithm contains the following steps:

1) Obtain a new image and a new robot pose estimation
2) Select a subset of image pixels to adapt the cluster for each color class
3) Select a different subset of pixels to add colors to the look-up table
4) Select a different subset of pixels to remove colors from the look-up table

A. Computation of the expected color class

Given the pose of the robot and the mapping from pixel to world coordinates, the algorithm can compute, which part of the field should correspond to which pixel. This works fine for the floor and the lines, as the mapping is usually trained for the field plane. Every pixel that corresponds to coordinates inside of the field is either classified as white field line or green floor, according to the field model. As green is the predominant color in the image in RoboCup, situations where the static model computes a different color class for a green pixel are very uncommon, even if the pose estimation from the self-localization is not accurate. Therefore, the mapping of colors that are already marked as green cannot be changed. White, however, is very rare in the image but has the main influence on the landmark-based self-localization. Small deviations in the position and the orientation of the pose estimation may already result in a completely wrong mapping from colors to this class. Therefore, a special treatment is used for pixels that are mapped to the white class according to the model. Only those pixels that have a higher intensity than their surrounding are ultimately used to train white. Given the intensity of a pixel $I(p_{x,y})$ at position (x, y) this filter is defined as

$$I(p_{x,y}) > \sum_{i=x-2}^{x+2} \sum_{j=y-2}^{y+2} I(p_{i,j}). \quad (1)$$

Objects and landmarks like the two goals, extend into the third dimension and thus only the contact points of these objects to the floor can be mapped correctly. Starting with these pixels, however, there is usually a clearly defined region of the image that displays the rest of the object, depending on the camera system used. In case of the omnidirectional camera system used on our RoboCup robots this region would be a trapezoid for the goals. Only pixels inside this area are mapped to one of the goal color classes, yellow and blue. As the other robots on the field are not static, the black color class is trained using only pixels that correspond to the black chassis of the robot itself. The ball color, though, is a problem for a calibration-free algorithm, as the ball is not static, too, and there is no way of training the ball color without some

previous knowledge about the color or position of the ball. One possibility to overcome this problem would be to have a special color marker of the ball color on the robot itself. But as RoboCup robots are not allowed to use the ball color this marker must on the one hand be hidden such that other robots will not get distracted and on the other hand be still illuminated enough to reflect the real ball color. The other possibility would be to define a base color which represents the real balls color good enough to initially locate the ball on the field. From this point onwards, the balls position could be used to retrain the ball color just as the color of the goals or the robots. All pixels that are mapped to a position outside of the field are assigned to the special color class *unknown*.

B. Adaptation of the cluster for each color class

For each color class $k = 1 \dots 6$, ACT tracks a cluster in the color space with a mean value μ_k and a standard deviation σ_k resulting from the previous cycles. For color values

$$c = (u, v, w) \in \{0, C_{max}\}^3, \tag{2}$$

the parameters of the different clusters are initialized as

$$\mu_{k,0} = \frac{1}{2}(C_{max}, C_{max}, C_{max}) \tag{3}$$

$$\sigma_{k,0} = \frac{\sqrt{3}}{2} C_{max} \tag{4}$$

for $t = 0$ and are updated in each cycle as follows.

A random set of pixels of the image is taken to update the clusters. To save computation time, the algorithm uses a fixed pattern of every 400th pixel, starting at a random pixel. According to the estimated pose of the robot and the static field model the expected color class of all these pixels is computed considering the special treatments of green and white explained in section II-A. Given the set of colors $X_{k,t} = c_1, \dots, c_m$ at cycle t of all pixels that are expected to belong to color class k the new values are computed as

$$\mu_{k,t} = \frac{1}{\eta + 1}\left(\eta\, \mu_{k,t-1} + \frac{1}{m}\sum_{i=1}^{m} c_i\right) \tag{5}$$

$$\sigma_{k,t} = \frac{1}{\eta + 1}\left(\eta\, \sigma_{k,t-1} + \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(c_i - \mu_{k,t})^2}\right) \tag{6}$$

$$\sigma_{k,t} = \sigma_{min}\,, \text{ if } \sigma_{k,t} < \sigma_{min}. \tag{7}$$

The choice of η determines the responsiveness of the color look-up table update. A value of $\eta = 4$ was empirically determined as optimal, enabling the algorithm to extremely reduce the number of examined pixels. Values of $\eta < 4$ for the filtering result in quicker adaptation of the table but also in a very noisy estimation of the cluster parameters. In order to avoid the cluster from collapsing, a lower bound of the standard deviation σ_{min} is introduced, ensuring a minimum cluster size. If sudden changes in illumination occur, it is possible that the cluster is too small to include the new color values. In such cases the set of color values for a color class is empty, and the standard deviation is doubled

$$\mu_{k,t} = \mu_{k,t-1} \tag{8}$$

$$\sigma_{k,t} = 2\,\sigma_{k,t-1}, \tag{9}$$

to increase the size of the cluster until it includes the new color values. The resulting clusters do not specify the colors that are finally stored in the color look-up table for the associated color class. They are rather a hint, where the color values of the color class might be located in the color space, however, they can be arbitrarily distributed around the cluster center and part of the cluster may belong to another color class.

C. Add colors to the color look-up table

To find out which color values are finally mapped to the different color classes, again a subset of every 400th image pixel is selected, starting at a different random pixel. After the calculation of the expected color classes, each color value is compared to the mean value of the corresponding color class. Given a color value c that is computed to belong to color class k, the Euclidian distance to the mean of color class k is compared to a multiple of the standard deviation σ_k, and the mapping from c to k is only added into the look-up table if

$$\|\mu_k - c\| < \zeta\, \sigma_k, \quad \text{with } \zeta > 1, \tag{10}$$

with $\|\cdot\|$ being the Euclidian norm and ζ being a threshold controlling the ratio between higher adaptability of the color look-up table and a higher false positive rate. The influence of this parameter is investigated through experiments in section IV-A. Similar to the manual training of the look-up table not only the color value itself is added into the table but also a small set of colors around c.

D. Remove colors from the color look-up table

Sometimes the manual or automatic addition of such a set of color values is too much, resulting for example in many occurences of the white color class outside of the field. Therefore, after the addition of the color classes to the look-up table, colors that are mapped to the special *unknown* class are removed from the table. Once again a different subset of pixels is selected to do this. To process a large number of pixels outside of the field, this time every 20th pixel is used, starting from a random pixel. The higher number of processed pixels is also neccessary to completely remove unwanted color mappings, as this time only the color value itself is removed from the color look-up table. A color value c that is expected to belong to the *unknown* class in this step but that is already mapped to color class k is removed from the table. As this may remove colors that are needed for a good classification of pixels inside the field, only colors that comply with

$$\|\mu_k - c\| > \xi\, \sigma_k, \quad \text{with } \xi > 1, \tag{11}$$

are removed, with ξ being a threshold controlling the ratio between a lower false positive rate and lower true positive rate. The influence of ξ is also investigated in section IV-A.

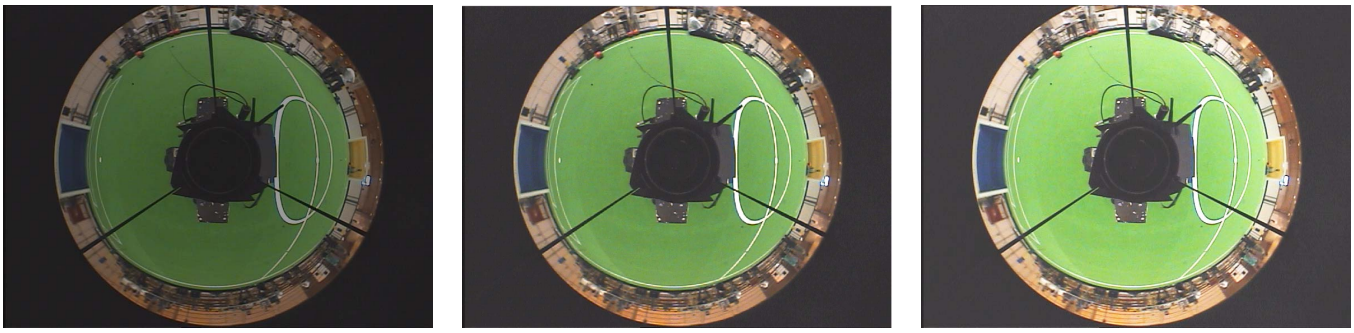

Fig. 1. Test images: The one in the middle was taken at the RoboCup World Cup 2004 at Lisbon with robot I, the other two were generated from this image by changing the brightness and the contrast of the image by 50% with an image manipulation program to simulate different lighting situations.

III. Hardware and Settings Used for the Experiments

Three types of robots were used for the following experiments, two generations of robots of our own Attempto Tuebingen team and one robot of the Mostly Harmless team. All robots are equipped with an omnidirectional camera system consisting of a perspective camera pointing upwards to a hyperbolic mirror. Robot I is from our previous RoboCup MSL team and has a standard 25fps camera with a resolution of 768×576 pixels and only 5bit resolution per color channel. The color look-up table trained for this camera system uses only 32KB of data which easily fits into the 2nd level cache of modern computers. However, the resolution of 5bit per color channel is critical if the illumination on the field is very low. Therefore, robot II of our new team is equipped with a 50fps, 580×580 pixel camera with 8bit resolution per color channel. Although the size of the look-up table is increased to 16.7MB, this larger table improves the performance of the algorithm significantly, when playing in changing, low illumination scenarios. Robot III is a robot of the Mostly Harmless MSL team and is equipped with a camera comparable to that of robot II but has a different mirror. Although all camera systems used for the experiments are omnidirectional, the ACT algorithm is independent of the type of camera system used, if a mapping from camera to world coordinates exists. In addition to different robots and camera systems, we also tested the algorithm on different RoboCup fields. Two of the fields are training fields, which are not large enough to fulfill the requirements of current MSL fields. They are of $6 \times 5m^2$ (field I) and $10 \times 5m^2$ size (field II). For the third setting (field III), images from the RoboCup World Cup 2004 at Lisbon from a field of size $12 \times 8m^2$ are used.

IV. Results

This section presents experimental results of the ACT algorithm. First, the influence of the two thresholds for adding and removing color mappings from the color look-up table is analysed. To prove the applicability of the ACT algorithm to automatic color training in RoboCup, several experiments with different static robots are presented followed by an experiment with a moving robot.

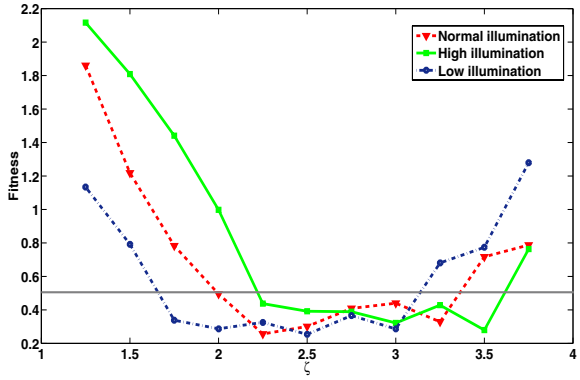

Fig. 2. Fitness of the color mapping of the resulting color look-up tables when ACT is applied to the test images.

A. Influence of the thresholds

To analyse the influence of the thresholds $\zeta\,\sigma_k$ and $\xi\,\sigma_k$ for adding and removing color mappings from the color look-up tables the algorithm was tested on three images with different brightness shown in figure 1. The image in the middle is an original image taken by robot I on field III, while the other two were generated from this image by changing the brightness and the contrast of the image with an image manipulation program to simulate different lighting situations. First, for each image, several runs of the algorithm were started with different values for ζ and using the appropriate pose estimation. As these experiments were aimed to investigate the influence of ζ when adding colors to the look-up table, the removal of colors was not included. After a few cycles the color look-up table converged to a stable state in each run and the quality or fitness of the resulting color look-up table was computed as the sum of the fitness of the k color classes

$$f = \sum_k \left(1 - \frac{TP_k}{TP_k + FN_k}\right), \tag{12}$$

where TP_k is the number of true positives for class k and FN_k is the number of false negatives for class k. The fitness values for all runs are shown in figure 2. For the darkened image, the algorithm converges to the best fitness for values $\zeta = [1.75, 3.0]$, while the interval of best fitness is $\zeta = [2.0, 3.25]$

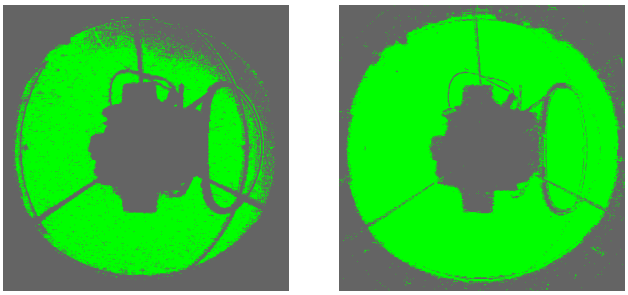

Fig. 3. Influence of ζ: The left image was classified with a very low value for ζ, while for the right image ζ was set too high.

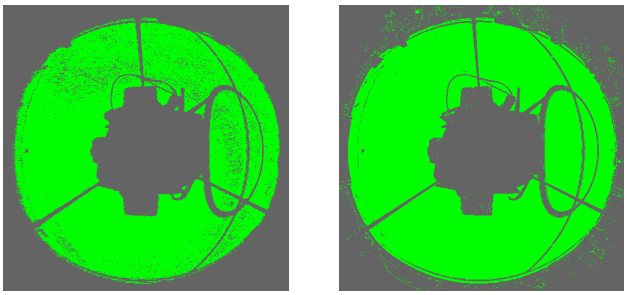

Fig. 4. Influence of ξ: The left image was classified with a very low value for ξ, while for the right image ξ was set too high.

and $\zeta = [2.25, 3.0]$ for the normal and the brightened image, respectively. On the one hand, lower values of ζ force the color classes to converge to a very small part of the color space that might not include all color values needed to correctly classify all pixels (cf. figure 3, left). If the image is very dark, however, the differences between the color classes and the deviation of the colors from the common mean per class are very low. Therefore good results can be achieved with lower values of ζ in the darker image. On the other hand, higher values of ζ enable the color class to spread out in the color space, resulting in a high standard deviation. This includes many colors that should not be mapped to this color class (cf. figure 3, right). Fortunately, there is a broad range of values $\zeta = [2.25, 3.0]$ for which all three images are classified with a very high fitness. For all subsequent experiments, a value of $\zeta = 2.5$ was used.

Secondly, the influence of ξ was tested. The experiments done with the three test images show that different values of ξ result in similar classification results for all three images. In fact, the selection of ξ for a good quality of the algorithm is depending far more on the selection of ζ. With $\zeta = 2.5$ the best results were achieved with a value of $\xi = [1.2, 1.5]$. For lower values of ξ, too many colors are removed from the table, while for higher values of ξ, too many colors from outside of the field remain in the table (cf. figure 4). For all subsequent experiments, a value of $\xi = 1.35$ was used.

B. Automatic color training on a static robot

Two experiments were carried out on a static robot. The first experiment demonstrates, that the ACT algorithm works on different camera systems and field settings. An image taken by robot III on field I and the same image classified with the color table trained by ACT are shown in figure 5. The result is a clear classification of the important parts of the field, while the number of classified pixels outside of the field is low.

The second experiment demonstrates that a robot using the ACT algorithm is able to cope with sudden changes of lighting. First, the image in the middle of figure 1 is used for training a color look-up table. The classified version of this image using the table trained by ACT is shown on the left side in figure 6. Then, to simulate such a sudden change in lighting, the same look-up table was used to classify the darkened version of this image (cf. figure 1, left). The image shown on the right side in figure 6 shows how a robot with static color look-up table would classify an image after the change of lighting. Clearly, a robot without online color training would have no chance of playing using such a color classification. With the ACT algorithm, however, the color look-up table is adapted to a stable optimum in only 12 cycles, which would result in only 240ms without color classification on robot II. In addition, the look-up table is already close to the optimum after 2 cycles, at least for the important color classes green and white. Figure 7 shows the classified image after 1, 2, and 8 cycles of adaption.

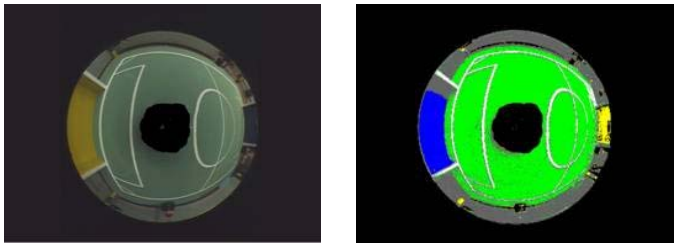

Fig. 5. An image taken from robot III on field I (left) and the same image classified with the color look-up table trained by the ACT algorithm (right)

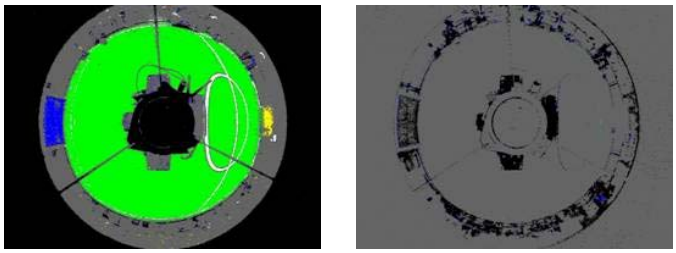

Fig. 6. The classified version of the image shown in the middle in figure 1 using the look-up table trained by ACT (left). Using the same table to classify the darkened version of this image (cf. figure 1, left) to simulate a sudden change in lighting, results in the image shown on the right. Clearly, a robot without online color training would have no chance of playing using such a color classification.

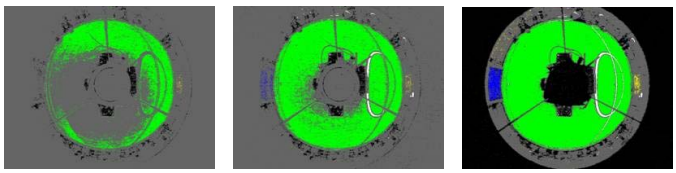

Fig. 7. The classified image after 1, 2, and 8 steps of retraining the color look-up table shown on the right in figure 6 with the ACT algorithm.

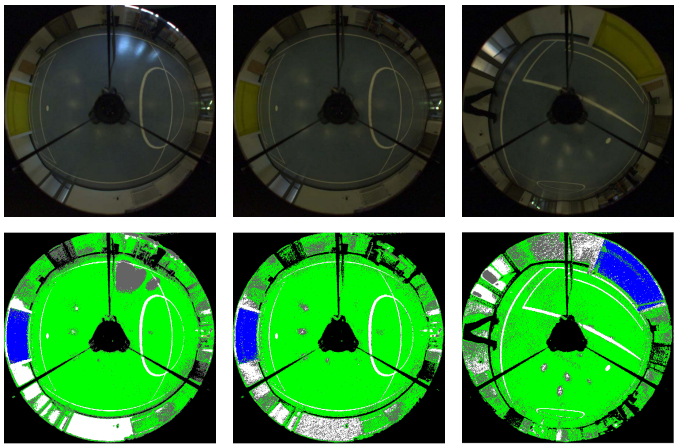

Fig. 8. After a few cycles of adaptation the ACT algorithm is able to train a color look-up table in these situations, even with a wrong pose estimation.

C. Automatic online color training on a moving robot

As the presentation of results from a moving robot is difficult, one experiment was carried out to show that the ACT algorithm embedded in the robot control system including the self-localization is able to handle a wrong pose estimation. For that, three different images were consecutively given to the algorithm, two from a known pose of the robot and one from a pose where the robot was located 2.5m away from the old pose and rotated by 165 degrees (cf. figure 8, right). These images were taken by robot II on field II, the first one with some direct illumination that is reflected on the blue PVC floor, the others with very low illumination (cf. figure 8, top row). While the self-localization uses the trained color look-up table for the pose estimation, the ACT algorithm always used the known pose as estimation. The classification results are shown in figure 8. With this setup, two features are shown: first, the ACT algorithm is able to cope with the worst-case scenario that contains very low ambient illumination (lower than 200lux) and rather bright spot lights. Secondly, ACT is able to keep a good color look-up table, even if the pose used for retraining is completely wrong. As the quality of the look-up table adapted by ACT is hardly depending on the quality of the pose estimation, the mutal dependency of the self-localization and the color training is no problem when using the ACT algorithm. The quality of the self-localization, however, might be decreased for those cycles where ACT is adapting to considerable changes in lighting. Nevertheless, the pose estimation was very good in this experiment after a few cycles of retraining the color look-up table for each image.

As the computation time per cycle was below 4ms on an Athlon XP 2400+ with 2GHz in all experiments presented in this paper, the ACT algorithm perfectly fits into the 20ms main cycle time on our new RoboCup MSL robots. First experiments show that the robot is able to maintain this main cycle during the play which is very important to compete in the highly dynamic environment of RoboCup MSL. In these experiments we could also show, that the robot is able to localize and play well using the online adapted color look-up table generated by the ACT algorithm.

V. Conclusions

This paper presents an algorithm for automatic online training of a look-up table that maps the colors of a three-dimensional color space onto different color classes used for the detection of objects and landmarks in camera images. For that the ACT algorithm incorporates knowledge about its environment to compute which colors correspond to which color class. In the RoboCup scenario this knowledge is the dimension and the layout of the field as stated in the rules and a pose estimation of the robot coming from a self-localization algorithm. ACT consecutively adapts the look-up table to changing lighting situations resulting in a robust classification of the image pixels. The presented results show that the main parameters of the algorithm can be chosen in a way to produce good results over a large variety of lighting scenarios. Additionally, it is shown that the algorithm is not restricted to specific hardware and offers a good performance even with sudden changes in lighting. Finally, the algorithm was implemented on a RoboCup MSL robot for online training of the color look-up table. Here, the mutual dependency of the color training and the self-localization is shown to have very little impact on the robustness of the algorithm, as the color training is very stable, even for a completely wrong pose estimation. With a cycle time of only 4ms the ACT algorithm was easily embedded into the control system of our RoboCup robots with a main cycle time of 20ms.

References

[1] T. Bandlow, M. Klupsch, R. Hanek, and T. Schmitt. Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario. In *3. RoboCup Workshop, IJCAI'99*, 1999.

[2] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proc. 2000 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, volume 3, pages 2061–2066, 2000.

[3] D. Cameron and N. Barnes. Knowledge-Based Autonomous Dynamic Colour Calibration. In *RoboCup-2003: Robot Soccer World Cup VII*, volume 3020 of *LNCS*, pages 226–237. Springer, 2004.

[4] R. Hanek, T. Schmitt, S. Buck, and M. Beetz. Towards RoboCup without Color Labeling. In *RoboCup 2002: Robot Soccer World Cup VI*, volume 2752 of *LNCS*, pages 426–434. Springer, 2003.

[5] P. Heinemann, J. Haase, and A. Zell. A Novel Approach to Efficient Monte-Carlo Localization in RoboCup. In *RoboCup 2006: Robot Soccer World Cup X*. Springer, 2006.

[6] P. Heinemann, T. Rückstieß, and A. Zell. Fast and Accurate Environment Modelling using Omnidirectional Vision. In *Dynamic Perception 2004*. Infix, 2004.

[7] P. Heinemann, F. Sehnke, F. Streichert, and A. Zell. Automatic Calibration of Camera to World Mapping in RoboCup uing Evolutionary Algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2006)*, 2006.

[8] M. Jüngel. Using Layered Color Precision for a Self-Calibrating Vision System. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *LNCS*, pages 209–220. Springer, 2005.

[9] G. Mayer, H. Utz, and G. Kraetzschmar. Towards autonomous vision self-calibration for soccer robots. In *Proc. 2002 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2002.

[10] G. Mayer, H. Utz, and G. Kraetzschmar. Playing Robot Soccer under Natural Light: A Case Study. In *RoboCup-2003: Robot Soccer World Cup VII*, volume 3020 of *LNCS*, pages 238–249. Springer, 2004.

[11] M. Sridharan and P. Stone. Towards Illumination Invariance in the Legged League. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *LNCS*, pages 196–208. Springer, 2005.

[12] A. Treptow, A. Masselli, and A. Zell. Real-Time Object Tracking for Soccer-Robots without Color Information. In *European Conference on Mobile Robotics (ECMR 2003)*, pages 33–38, 2003.