In *The Twentieth National Conference on Artificial Intelligence* (AAAI 05), Pittsburgh, USA, July 2005.

Autonomous Color Learning on a Mobile Robot

Mohan Sridharan and **Peter Stone**
University of Texas at Austin
Austin, TX-78712
smohan@ece.utexas.edu, pstone@cs.utexas.edu

Abstract

Color segmentation is a challenging subtask in computer vision. Most popular approaches are computationally expensive, involve an extensive off-line training phase and/or rely on a stationary camera. This paper presents an approach for color learning on-board a legged robot with limited computational and memory resources. A key defining feature of the approach is that it works without any labeled training data. Rather, it trains autonomously from a color-coded model of its environment. The process is fully implemented, completely autonomous, and provides high degree of segmentation accuracy.

Introduction

Computer vision is a major area of research with applications in robotics and artificial intelligence. One of the principal subtasks in vision is that of *color segmentation*: mapping each pixel in a camera image to a color label. Though significant advances have been made in this field (Comaniciu & Meer 2002; Sumengen, Manjunath, & Kenney 2003), most of the algorithms are computationally expensive and/or involve a time consuming off-line preprocessing phase. In addition, segmentation is typically quite sensitive to illumination variations: a change in illumination causes a nonlinear shift in the mapping which could necessitate a repetition of the entire training phase.

This paper presents an efficient online algorithm for color segmentation in task-oriented scenarios with limited computational resources. Autonomous mobile robots operating in controlled environments represent one such scenario. A key defining feature of the algorithm is that it works without any labeled training data. Rather it trains autonomously from a color-coded model of its environment.

The problem of color segmentation as addressed here can be characterized by the following set of inputs, outputs and constraints:

1. *Inputs:*
   - A color-coded model of the world that the robot inhabits. The model contains a representation of the size, shape, position, and colors of all objects of interest.
   - A stream of limited-field-of-view images with noise and distortion as a result of motion of the camera and/or robot. The images present a view of the *structured* world with many useful objects, but also many unpredictable elements.
   - The initial position of the robot and its joint angles over time, particularly those specifying the camera motion.
2. *Output:*
   - A *Color Map* that assigns a *color label* to each point in the input color space.
3. *Constraints:*
   - Limited computational and memory resources with all processing being performed on-board the robot.
   - Rapid motion of the limited-field-of-view camera with the associated noise and image distortions.

Note in particular that there is *no* labeled training data or apriori bias regarding the labels of points in color space. This provides for robustness to different lighting conditions and even changes of entire colors (e.g. repainting all red objects as blue and vice versa) without any disruption to the robot.

Our goal is to generate a reliable mapping from the above inputs to the outputs, while operating within the constraints imposed by the test platform. In this paper, we describe an approach to autonomously generate such a color map in real-time. The algorithm is fully implemented and tested using the SONY Aibo legged robots (Sony 2004) in a pre-existing task environment.

The remainder of the paper is organized as follows. We first formally specify our problem and describe the test domain. Then, we describe the overall algorithm and the specific instantiation in our problem domain. The experimental setup and results are presented in the subsequent sections. We conclude with a brief description of related work and our conclusions.

Problem Specification

In this section we formally describe the problem of generating a color map for the robot. We also elaborate on our specific test platform and its constraints.

To be able to recognize objects and operate in a color-coded world, a robot generally needs to recognize a certain discrete number (N) of colors ($\omega \in [0, N-1]$). A complete mapping identifies a color label for each possible point in the color space (Gonzalez & Woods 2002) under consideration:

$$\forall p,q,r \in [0,255] \qquad (1)$$
$$\{C_{1,p}, C_{2,q}, C_{3,r}\} \mapsto \omega|_{\omega \in [0,N-1]}$$

where C_1, C_2, C_3 are the three color channels (e.g. RGB or YCbCr), with the corresponding values ranging from $0-255$.

For representing the colors, we use a Three-Dimensional (3D) Gaussian model with the assumption of mutually independent color channels: a 3D Gaussian was observed to reasonably approximate actual color distributions (except for some edge effects). The independence assumption implies that there is no correlation among the values along the three color channels for any given color. In practice, the independence assumption does not hold perfectly, depending on the color space under consideration. For example, in our lab, the correlation coefficients between the *Cb* and *Cr* color channels in the *YCbCr* color space are $\rho_{cbcr} = -0.71, -0.67$ for orange and yellow respectively. Nonetheless, the independence assumption closely approximates reality and greatly simplifies the calculations — computationally expensive operations such as inverting a covariance matrix need not be performed.

Each color $\omega \in [0, N-1]$ is then represented by the density distribution:

$$p(\omega|c_1, c_2, c_3) = \frac{1}{\sqrt{2\pi}\prod_{i=1}^{3}\sigma_{C_i}} \cdot \exp -\frac{1}{2}\sum_{i=1}^{3}\left(\frac{c_i - \mu_{C_i}}{\sigma_{C_i}}\right)^2 \quad (2)$$

where, $c_i \in [C_{i_{min}}, C_{i_{max}}]$ represents the value at a pixel along a color channel C_i while μ_{C_i} and σ_{C_i} represent the corresponding means and variances.

Though more complex representations (of color distributions) have been used in the literature, the Gaussian model has the advantage that the means and variances of the distributions are the only statistics that need to be collected and stored for each color that is to be learnt. This feature makes the learning process fast and feasible to execute on the robot. The next section presents the learning setup and the actual learning process that the robot goes through to learn the color map. The remainder of this section elaborates on the robot and the experimental testbed.

The SONY Aibo, *ERS-7*, is a four legged robot whose primary sensor is a CMOS camera located at the tip of its nose, with a field-of-view of 56.9^o (hor) and 45.2^o (ver), providing the robot with a limited view of its environment. The images are captured in the *YCbCr* format at $30Hz$ with an image resolution of 208×160 pixels. The robot has 20 degrees-of-freedom (dof), three in each of its four legs, three in its head, and a total of five in its tail, mouth, and ears. It also has noisy touch sensors, IR sensors, and a wireless LAN card for inter-robot communication. The camera jerks around a lot due to the legged (as opposed to wheeled) locomotion modality, and images have a relatively low resolution and possess common defects such as noise and distortion.

The RoboCup Legged League is a research initiative which currently has teams of four robots playing a competitive game of soccer on an indoor field of size $\approx 3m \times 4.5m$ (see Figure 1).

Figure 1: An Image of the Aibo and the field.

On the robot, visual processing occurs in two stages: color segmentation and object recognition (et al. 2004). In the off-board training phase, we train a color map that maps a space of $128 \times 128 \times 128$ possible pixel values[1] to one of the 9 different colors that appear in its environment (pink, yellow, blue, orange, red, dark blue, white, green, and black). The color map is then used to segment the images and construct connected constant-colored blobs out of the segmented images. The blobs are used to detect useful objects (e.g. markers, the ball, and opponents).

[1]we use half the normal resolution of 0-255 along each dimension to reduce memory requirements

The robot uses the markers to localize itself on the field and coordinates with its team members to score goals on the opponent. All processing, for vision, localization, locomotion, and action-selection, is performed on board the robots, using a 576MHz processor. Currently, games are played under constant and reasonably uniform lighting conditions but the goal is to enable the robots to play under varying illumination conditions.[2] This goal puts added emphasis on the ability to learn and adapt the color map in a very short period of time.

A variety of previous approaches have been implemented in the RoboCup domain to generate the color map (see the *related work* section at the end of the paper). But, almost all of them involve an elaborate training process wherein the color map is generated by hand-labeling several ($\approx 20-30$) images over a period of at least an hour. This process leads to a long setup time before the games can begin. This paper presents a novel approach that enables the robot to autonomously learn the entire color map, using the inherent structure of the environment and about seven images, in less than five minutes. The segmentation accuracy is comparable to that obtained by the color map generated by the hand-labeling process.

Learning Setup

In this section we describe the task that the robot executes to autonomously learn the color distributions. Our learning mechanism is summarized in Algorithm 1. Specific implementation details are explained below.

Algorithm 1 General Color Learning

Require: Starting Pose Known, Model of the robot's world.
Require: *Empty* color map.
Require: Array of poses for learning colors, $Pose[]$.
Require: Array of objects, described as shapes, from which the colors need to be learnt, $Objects[]$.
Require: Ability to move to a target pose.

```
i = 0, N = MaxColors
Time_st = CurrTime
while i < N and CurrTime − Time_st ≤ Time_max do
    Motion = RequiredMotion( Pose[i] )
    Perform Motion {Monitored using visual input}
    if LearnGaussParams( Colors[i] ) then
        Learn Mean and Variance of color from candidate image pixels
        UpdateColorMap()
        if !Valid( Colors[i] ) then
            RemoveFromMap( Colors[i] )
        end if
    end if
    i = i + 1
end while
Write out the color statistics and the color map.
```

The algorithm can be described as follows: The robot starts off at a known position with a known model of its world. It has no initial color information, i.e. the means and variances of the colors to be learnt are initialized to zero and the images are all segmented *black*. It also has three lists: the list of colors to be learnt (*Colors*), a list of corresponding positions that are appropriate to learn those colors (*Pose*), and a list of corresponding objects, defined as shapes, that can be used to learn the colors. Using a navigation function (RequiredMotion()), the robot determines the motion required, if any,

[2]The stated goal of RoboCup is to create a team of humanoid robots that can beat the human soccer champions by the year 2050 on a real, outdoor soccer field (Kitano *et al.* 1998)

to place it in a position corresponding to the first entry in *Pose*, and executes the motion command. Since the motion is noisy, the robot monitors the motion to the target position by searching for a candidate blob suitable for learning the desired color. The object shape definition – the corresponding entry in the *Objects* array – leads to a set of constraints (heuristic *candidacy tests*) that are used to select the candidate blob. This typically corresponds to an object in the environment that has the color the robot is currently interested in learning. The robot stops when either a suitable blob is found or it thinks it has reached its target position. Further details of the *candidacy tests* can be found in our team's technical report (et al. 2004).

Once in position, the robot executes the function *LearnGaussParams()* to learn the color. If a suitable candidate blob of *unknown* color (*black* in our case) exists, each pixel of the blob is examined. If the pixel value (a vector with three elements corresponding to the three color channels) is sufficiently distant from the *means* of the other known color distributions, it is considered to be a member of the color class under consideration. When the entire blob has been analyzed, these pixels are used to arrive at a *mean* and a *variance* that then represent the *3D Gaussian density function* of the color being learnt. In order to suppress the noise that is inherent in the camera image, the robot repeatedly extracts information from several screen-shots of the same image. This redundancy provides more accurate statistics corresponding to the desired colors.

The function *UpdateColorMap()* takes as input all the learned Gaussians and generates the full mapping from the pixel value to the color label. This process of assigning color labels to each cell in the $128x128x128$ cube is the most computationally intensive part of the learning process. Hence, it is performed only once very five seconds or so. Each cell is assigned a color label corresponding to the color which is most *likely* for that set of color channel values, i.e. the color whose density function (Equation 2) has the largest *probability* value. The updated color map is used to segment all subsequent images.

The segmented images are used for detecting objects, which in turn are used to validate the colors learnt. Once a color has been included in the color map, the robot checks the validity of the color parameters (*Valid()*) by looking for suitable objects in the current position. Typically, it is the object that was used to learn the color. The detected objects are also used to update the color parameters.

The entire learning procedure is repeated until all desired colors are learnt and/or the predecided learning time ($Time_{max}$) has elapsed.

By definition, the Gaussian distributions have a non-zero value throughout the color space. During the learning process, the robot could therefore classify all the image pixels into one of the colors currently included in the map, thereby leaving no candidate blobs to learn the other colors. Therefore, an image pixel is assigned a particular color label *iff* its distance from the mean of the corresponding color's distribution lies within a certain integral multiple of the corresponding standard deviation. The robot is allowed to dynamically modify these limits, within a range, while it is learning the colors.

Experimental Setup

In this paper, we use the legged league field with its color coded goals and markers, as shown in Figure 2, as the testbed. The robot starts off with *no prior knowledge* of any of the colors and its goal is to learn a color map that includes the desired colors. Here, we present the results when the robot always starts off in *Position-1* and moves through a deterministic sequence of positions, as shown in Figure 2. These positions form the elements of the array *Pose[]* in the algorithm described above (Algorithm 1).

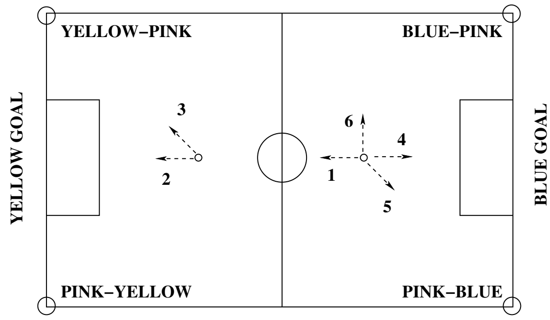
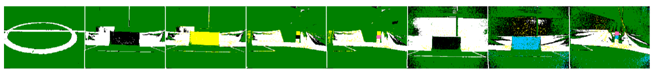


Figure 2: A Line Drawing of the Learning positions.

The steps involved in the algorithm can be presented as an ordered list of positions, colors and objects:

1. Step-1: Position1 with head tilted down, *white* and *green* learnt. Field line and center circle.
2. Step-2: Position-2, *yellow* learnt, Yellow goal.
3. Step-3: Position-3, *pink* learnt, Yellow-pink marker.
4. Step-4: Position-4, *blue* learnt, Blue goal.
5. Step-5: Position-5, *blue* learnt (Disambiguate *green* and *blue*), Pink-blue marker.
6. Step-6: Position-6 with head tilted down, ball color (*orange*) learnt, Ball.
7. Step-7: Position-6 with head horizontal, opponent's uniform color learnt, Opponent.

The robot then writes out the color map and the color statistics to a file on the robot's memory stick. Figure 3 shows a few images at various stages of the learning process - note that only images corresponding to the markers are presented. A complete video of the learning mechanism, as seen from the robot's camera, can be viewed online:
www.cs.utexas.edu/∼AustinVilla/?p=research/auto_vis.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3: Images at different stages of the learning process

A few important points are to be noted with regard to the learning process. In *Position-1*, learning is performed based on the fact that a large portion of the image (in that position) consists of *green*. The algorithm is dependent only on the inherent structure of the environment and is entirely independent of the particular color that is being learnt. A concrete test would be to start the robot (unknown to the robot) in *Position-2* facing the blue goal and run the same algorithm. We shall show that without a change in the learning process, the robot is able to learn the color *blue* as *yellow* and vice versa. For the same reason, the procedure for learning the ball color works for balls of other colors too. This feature demonstrates the robustness of our algorithm to color transformations in the environment — as long as all objects of a given color are changed to the same new color, the procedure is unaffected.

We shall show that the algorithm enables the robot to learn the entire map in any fixed illumination within a range of illuminations (range of several 100lux decided by the camera hardware).

The positions for learning the ball and opponent colors are set so as to minimize the movement.

Experimental Results

In this section we test the accuracy of the color maps that were learned autonomously on the robots, using the learning task previously described. The color maps were analyzed with respect to their segmentation accuracy on a set of sample images. We also trained a color map by hand-segmenting a set of ≈ 25 images. We refer to this color map as the *Hand Labeled* (*HLabel*) color map. This map corresponds to a fixed illumination condition and the colors are not represented as Gaussians. Instead, for each color, an intermediate (IM) map (of the same size as the overall color map) is maintained. Each cell of an IM stores a count of the number of times an image pixel that maps into that cell was labeled as the corresponding color. Each cell in the final color map is assigned the label corresponding to the color whose IM has the largest count in that cell.

Previous results (Hyams, Powell, & Murphy 2000; Minten *et al.* 2001; Sridharan & Stone 2004) suggested that the *LAB* color space could be reasonably robust to illumination variations (the *LAB* color space is an alternate 3D representation of color in spherical coordinates (Minten *et al.* 2001)). We hypothesized that the color map in *LAB* would provide better segmentation accuracy and to test that we trained a color map in *LAB* in addition to that in the *YCbCr* color space. Though the robot is able to learn the marker colors in both color spaces, the performance in the LAB color space is better. This difference is more pronounced when the color of the ball and/or the opponent is included in the color map because these colors overlap with the marker colors and create a contention during segmentation. We therefore performed the analysis in stages: first with just the fixed marker colors and then with all the colors included.

On a set of sample images of the markers (15) captured using the robot's camera (see Figure 4 for samples), we first compared the performance of all three color maps with the color labeling provided interactively by a human observer, the *Ground Truth* (*GTruth*). Under normal game conditions we are interested only in the colors of the markers and other objects on the field and/or below the horizon because other blobs are automatically rejected in the object recognition phase. Also, the *correct* classification result is unknown (even with *HLabel*) for several background pixels in the image. Therefore, in this test, the observer only labels pixels suitable for analysis and these labels are compared with the classification provided by the three color maps. On average, ≈ 6000 of the 33280 pixels in the image get labeled by the observer. Table 1 presents the corresponding results – the last row presents the averages and the standard deviations.

Images	YCbCr	LAB	HLabel
Worst	84	97	98
Best	95	99	99
avg	87.8 ± 3.18	97.9 ± 0.76	98.8 ± 0.44

Table 1: Classification Accuracies (%) on a pixel-by-pixel basis on a set of test images.

Note that the color labeling obtained by using the *HLabel* color map is almost perfect in comparison to the human color labeling. The color map generated in the *LAB* color space also provides a similar performance and though they are both (statistically) significantly better than the *YCbCr* color map (at 95% level of significance, we obtain a p-value of $6x10^{-4}$ between *YCbCr* and *LAB*), there is not much difference in the qualitative performance.

Figure 4 illustrates the results of segmentation over the set of sample *marker* images, using the color map obtained at the end of the learning process.[3]

[3]The results are clearest when the images are seen in color. These and several more images are available at: *www.cs.utexas.edu/~AustinVilla/?p=research/auto_vis*

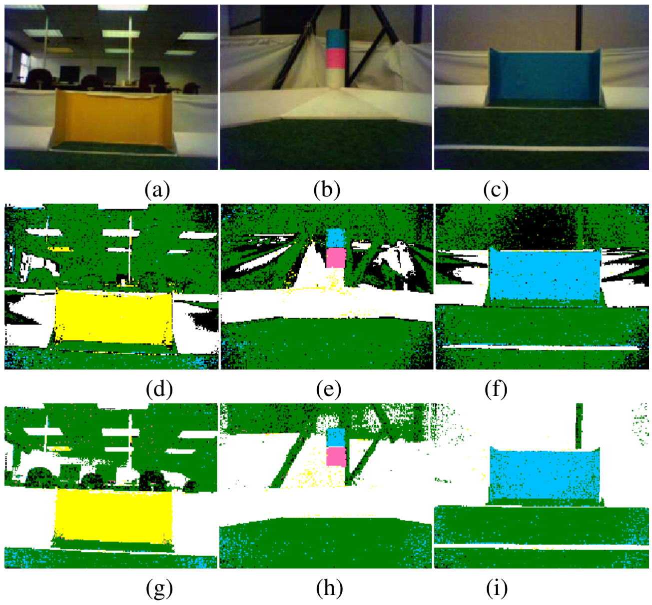


Figure 4: Sample Images without ball. (a)-(c) Original Images, (d)-(f) YCbCr color space, (g)-(i) LAB color space

As shown in the figures, the robot is able to learn a reasonable color map in both color spaces when only the fixed marker colors are considered.

To further justify the comparison over regions selected by a human observer, we compared the performance of the *YCbCr* and the *LAB* color maps with the *HLabel* color map, with the entire image taken into consideration. We compare with *HLabel* because it provides almost perfect segmentation and allows us to generate more labeled data. Over the same set of sample images used before, the average classification accuracies were 65.8 ± 7.9 and 88.3 ± 4.2 for *YCbCr* and *LAB* respectively. These values, though not a proper measure of the segmentation accuracy, can be considered to be worst-case estimates.

Next, we let the robot learn the ball color (*orange*) in addition to the marker colors. The average classification accuracies are $74.8 \pm 9.2\%$, $94 \pm 5.6\%$ and $98.5 \pm 0.8\%$ for the *YCbCr*, *LAB* and *HLabel* color maps respectively, as compared to *GTruth*. The performance in *LAB* is statistically significant as compared to *YCbCr* ($p - value = 5x10^{-4}$). Figure 5 show the segmentation results over a set of images.

We observe that in the *YCbCr* color space, the inclusion of *orange* in the color map causes the segmentation to degrade even over the colors (*pink* and *yellow*) that it could classify well before.[4] On the other hand, in the *LAB* color space, the inclusion of *orange* does not degrade the performance with regard to the other known colors. The only regions of the ball that the robot is unable to classify perfectly are the ones with highlights. This misclassification does not hurt the performance of the robot since the high-level object recognition procedure is still able to find the ball without any additional constraints (the ball is rarely found in the *YCbCr* color space). The robot is able to use this color map to play a game as well as it could with *HLabel*. We therefore learnt the color of the opponent's uniform (*red*) only in the *LAB* color space. Figure 6 shows the segmentation results with all the colors. There is still no adverse effect on the previously learnt colors.

The only disadvantage of using the *LAB* color space is that it takes a little more time to complete the color learning. While learn-

[4]Note, especially, the results on the image with the yellow goal and ball

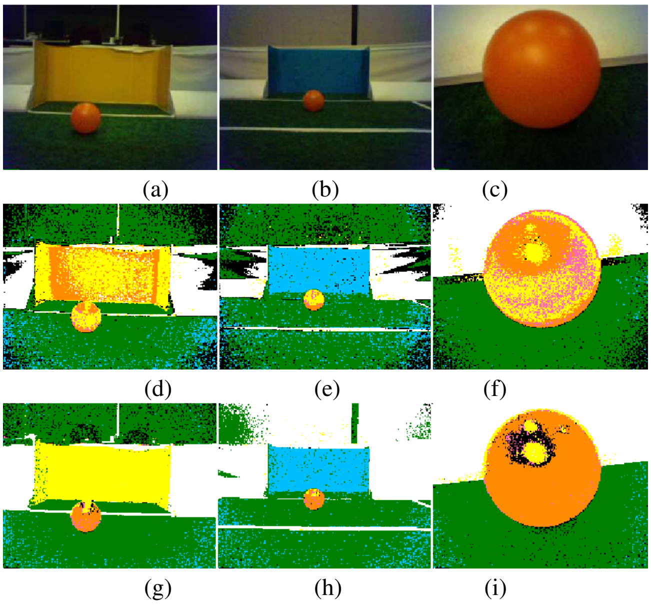


Figure 5: Sample Images with Ball. (a)-(c) Original Images, (d)-(f) YCbCr color space, (g)-(i) LAB color space

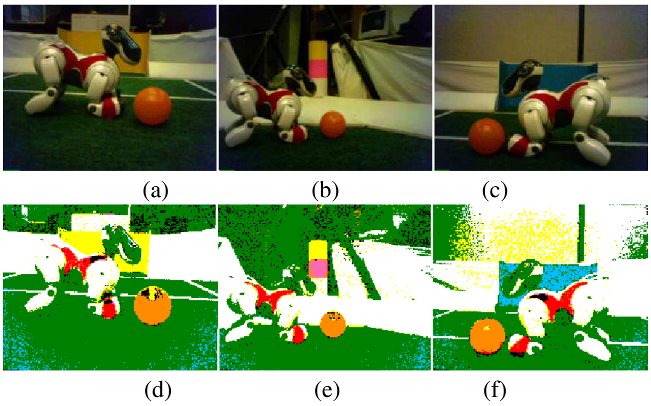


Figure 6: Sample Images with Opponent and Ball. (a)-(c) Original Images, (d)-(f) Segmented images

ing in the *LAB* color space, we still do not want to transform each pixel in the test image from *YCbCr* (the native image format) to *LAB* as this would make the segmentation phase extremely time consuming on the robot. So, when updating the color map in the learning process (*UpdateColorMap()* in Algorithm 1), we assign the color label to each discrete cell in the *YCbCr* color map by determining the label assigned to the corresponding pixel values in *LAB*. This ensures that test image segmentation on the robot is still a table lookup and therefore takes the same time as operating in *YCbCr*. The additional pixel-level transformation is the cause of the increase in the training time, though it is substantially offset by the ability to segment better. The learning process takes $\approx 2.5 minutes$ in *YCbCr* while it takes $\approx 4.5 minutes$ in *LAB*. Note that both of these numbers are still much smaller than the time taken to generate *HLabel*, an hour or more.

The robot can perform the learning process under any given fixed illumination within a range of illuminations in our lab (varies over several 100 lux). Consider the results over the images shown in Figure 7. When the illumination changes, the original color map does not perform well (notice the green in the goal and orange in the beacon which could be detected as a ball). But the robot is able to learn a new (suitable) color map in a few minutes.

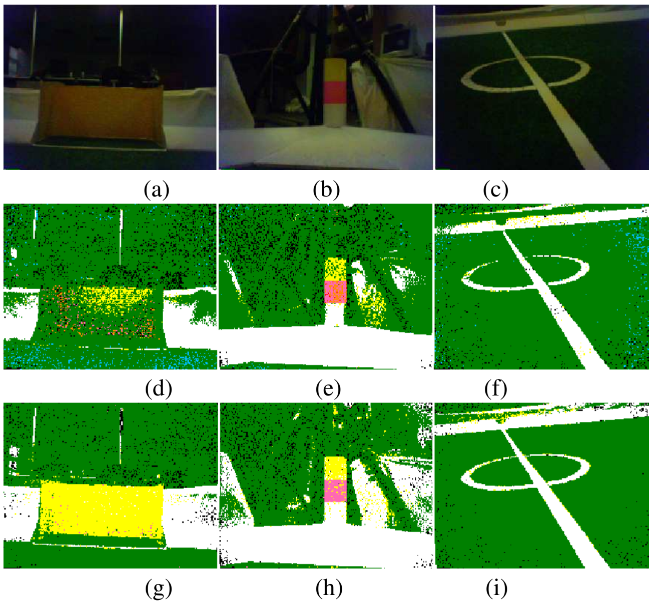


Figure 7: Sample Images under new illumination. (a)-(c) Original images, (d)-(f) Old Map, (g)-(i) New Map

Finally, to test the hypothesis that the algorithm is robust to color *re-mapping*, we started the learning process (unknown to the robot) with the robot in *Position-2*, facing the blue goal (see Figure 2). The robot ended up learning the color *blue* as *yellow* and vice versa. Figure 8 presents a sample of the corresponding segmentation results. This confirms our hypothesis that the process is totally dependent on shape and size and not on the particular color that is being learnt. For example, if the field were blue with yellow lines and the goals were green and white, the robot wouldn't notice the difference and no change in the algorithm would be required. This also implies that if such a robot is used in a house, repainting a wall would not pose a problem to the learning process.

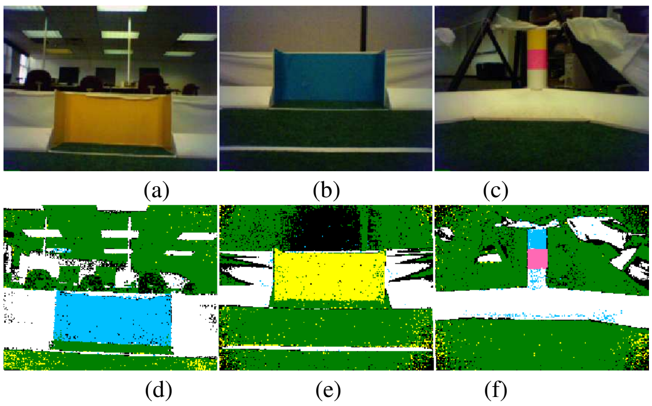


Figure 8: Sample Images with colors *reversed*. (a)-(c) Original images, (d)-(f) Colors reversed

It is essential to note that while learning color distributions, we are not attempting to model *absolute* colors – color labels do not hold much significance. We are more interested in making the robot learn the colors autonomously from its environment, using only the inherent structure.

To illustrate the learning process better, we provide the videos of the learning process, as seen from the robot's camera. (web-link:*www.cs.utexas.edu/∼AustinVilla/?p=research/auto_vis*)

Related Work

Color segmentation is a well-researched field in computer vision with several good algorithms, for example mean-shift (Comaniciu & Meer 2002) and gradient-descent based cost-function minimization (Sumengen, Manjunath, & Kenney 2003). But these involve computation that is infeasible to perform on autonomous robots given the computational and memory constraints. Even in the RoboCup domain, several algorithms have been implemented. The baseline approach involves creating mappings from the YCbCr values (ranging from $0 - 255$ in each dimension) to the color labels (Uther *et al.* 2001; Bruce, Balch, & Veloso 2000). Other methods include the use of decision trees (Chen *et al.* 2002) and the creation of axis-parallel rectangles in the color space (Cohen *et al.* 2004). Attempts to automatically learn the color map have been rarely successful. One such instance is (Jungel 2004), wherein the author presents a method to learn the color map using three layers of color maps with increasing precision levels, colors being represented as cuboids. But the generated map is reported to be not as accurate as the hand-labeled one and other domain specific constraints are introduced to disambiguate between object colors, during the object recognition phase. In (Schulz & Fox 2004), colors are estimated using a hierarchical bayesian model with *Gaussian* priors and a joint posterior on robot position and environmental illumination. Our approach on the other hand learns a color map using an efficient *Gaussian* representation for the color classes with no prior color knowledge. It involves very little storage and the resultant color map is comparable in accuracy to the hand-labeled one. In addition, this is done using a very small set of images in less than five minutes. Note that not only is our approach differentiated in that it learns the colors autonomously, thus dramatically reducing human effort, but it also yields good qualitative performance.

Conclusions/Future Work

We have presented an approach to automating the color learning and segmentation process on-board a legged robot with limited computational and storage resources. In spite of the relatively low-resolution images with inherent noise and distortion, the algorithm enables the robot to autonomously generate its color map in a very short period of time. The corresponding segmentation accuracy after about five minutes is shown to be comparable to the that obtained by hand-labeling several images over a period of an hour or more.

Though we have tested our approach in a single constrained environment, in principle, it applies much more generally. All that is needed is an environmental model with the locations of distinctive features labeled. In our work, we use colors as the distinctive features. But in environments with features that aren't constant-colored, other feature representations, such as those used by SIFT (Lowe 2004), could be used. As long as the *locations* of the objects remain as indicated on the map, the robot could robustly re-learn how to detect them.

In the domain considered here, both *YCbCr* and *LAB* color spaces are reasonably good for learning the marker colors. Including ball color (*orange*) causes a significant difference in performance between the two color spaces considered - *LAB* performs much better here (and even after *red* is included). The algorithm is dependent only on the structure inherent in the environment and a *re-mapping* of the colors does not prevent the robot from learning them. Further, the color map can be learnt in several fixed illumination conditions between a minimum and maximum on the field. The learning can be easily repeated if a substantial variation in illumination is noticed. This variation could be detected as a function of the shift in the means of the various colors.

Currently, the color map is learnt from a known fixed starting position without any prior knowledge of colors. An extension that we are currently working on is to learn from any given starting position on the field.

Acknowledgements

We would like to thank the members of the UT Austin Villa team for their efforts in developing the soccer-playing software mentioned in this paper. This work was supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

Bruce, J.; Balch, T.; and Veloso, M. 2000. Fast and inexpensive color image segmentation for interactive robots. In *The International Conference on Intelligent Robots and Systems (IROS)*.

Chen, S.; Siu, M.; Vogelgesang, T.; Yik, T. F.; Hengst, B.; Pham, S. B.; and Sammut, C. 2002. *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Berlin: Springer Verlag.

Cohen, D.; Ooi, Y. H.; Vernaza, P.; and Lee, D. D. 2004. *RoboCup-2003: The Seventh RoboCup Competitions and Conferences*. Berlin: Springer Verlag.

Comaniciu, D., and Meer, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(5):603–619.

et al., P. S. 2004. UT Austin Villa 2004: Coming of Age, AI TR 04-313. Technical report, Department of Computer Sciences, UT-Austin.

Gonzalez, R. C., and Woods, R. E. 2002. *Digital Image Processing*. Prentice Hall.

Hyams, J.; Powell, M. W.; and Murphy, R. R. 2000. Cooperative navigation of micro-rovers using color segmentation. *In Journal of Autonomous Robots* 9(1):7–16.

Jungel, M. 2004. Using layered color precision for a self-calibrating vision system. In *The Eighth International RoboCup Symposium*.

Kitano, H.; Asada, M.; Noda, I.; and Matsubara, H. 1998. Robot world cup. *IEEE Robotics and Automation Magazine* 5(3):30–36.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)* 60(2):91–110.

Minten, B. W.; Murphy, R. R.; Hyams, J.; and Micire, M. 2001. Low-order-complexity vision-based docking. *IEEE Transactions on Robotics and Automation* 17(6):922–930.

Schulz, D., and Fox, D. 2004. Bayesian color estimation for adaptive vision-based robot localization. In *The IEEE International Conference on Intelligent Robots and Systems (IROS)*.

Sony. 2004. The Sony Aibo robots. `http://www.sonystyle.com`.

Sridharan, M., and Stone, P. 2004. Towards illumination invariance in the legged league. In *The Eighth International RoboCup Symposium*.

Sumengen, B.; Manjunath, B. S.; and Kenney, C. 2003. Image segmentation using multi-region stability and edge strength. In *The IEEE International Conference on Image Processing (ICIP)*.

Uther, W.; Lenser, S.; Bruce, J.; Hock, M.; and Veloso, M. 2001. Cm-pack'01: Fast legged robot walking, robust localization, and team behaviors. In *The Fifth International RoboCup Symposium*.