In *The IEEE Ninth International Conference on Control, Automation, Robotics and Vision* (ICARCV 06), Singapore, December 2006.

Autonomous Planned Color Learning on a Mobile Robot Without Labeled Data

Mohan Sridharan
Electrical and Computer Engineering
The University of Texas at Austin, USA
smohan@ece.utexas.edu

Peter Stone
Department of Computer Sciences
The University of Texas at Austin, USA
pstone@cs.utexas.edu

Abstract*— Color segmentation is a challenging yet integral subtask of mobile robot systems that use visual sensors, especially since such systems typically have limited computational and memory resources. We present an online approach for a mobile robot to autonomously learn the colors in its environment without any explicitly labeled training data, thereby making it robust to re-colorings in the environment. The robot plans its motion and extracts structure from a color-coded environment to learn colors autonomously and incrementally, with the knowledge acquired at any stage of the learning process being used as a bootstrap mechanism to aid the robot in planning its motion during subsequent stages. With our novel representation, the robot is able to use the same algorithm both within the constrained setting of our lab and in much more uncontrolled settings such as indoor corridors. The segmentation and localization accuracies are comparable to that obtained by a time-consuming offline training process. The algorithm is fully implemented and tested on SONY Aibo robots.** ***Keywords: Color Learning, Robot Vision.

I. Motivation

Integrated robotic systems need to sense the world they operate in. One way to do that is through vision, a rich source of information. A principal subtask of visual processing is *color segmentation*: mapping each image pixel to a color label. Though significant advances have been made in this field [4], [7], most of the algorithms are computationally expensive and/or involve a time consuming off-line preprocessing phase. In addition, the resulting 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 with limited computational resources. A key defining feature of the algorithm is that there is *no* labeled training data or apriori bias regarding the labels of points in color space. This makes the algorithm suitable for use under different lighting conditions and even changes of entire colors (e.g. repainting all red objects as blue and vice versa).

The problem of color segmentation takes as input the color-coded model of the world with a representation of the size, shape, position and color labels of objects of interest. A stream of input images are provided and the robot's initial position (and its joint angles over time) are known. The desired output is a *Color Map* that assigns a *Color Label* to each point in the color space. This problem is challenging because the process is constrained to work within the limited memory and processing resources of the robot. Furthermore, it should be able to cope with the rapid motion of the limited-field-of-view camera, and with the associated noise and image distortions.

We build on our previous work [15], where the robot learned colors within the controlled lab setting with solid colors and constant, uniform illumination conditions, executing a motion sequence provided by a human observer. Vision research on mobile robots is often conducted in such settings, which makes algorithm development easier but typically makes assumptions that are not true of the real world. Here we enable the robot to work outside the controlled lab setting, which required algorithmic changes to deal with the non-uniformity of the surroundings, such as with textured surfaces. The robot is able to autonomously plan its motion sequence for any given configuration of objects, based on environmental knowledge and heuristic constraints on its motion sequence.

This paper makes two main contributions. First, it presents a novel hybrid generalization of our previous color representation scheme such that the robot is able to learn colors efficiently and effectively both in the controlled lab setting and in uncontrolled indoor settings. Second, it enables the robot to autonomously plan a motion sequence that puts it in positions suitable to learn the desired colors. The robot simultaneously learns colors and localizes, and incrementally performs better at both these tasks.

II. Problem Description

In this section, we formally describe the problem, our proposed hybrid color learning model, and the robot platform.

A. Color Representation

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

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

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

In our previous color learning approach [15], each color was modeled as a three-dimensional (3D) Gaussian with mutually independent color channels, i.e. no correlation among the values along the color channels. Using empirical data and the statistical technique of bootstrapping [6], we determined that this representation closely approximates reality. In addition to simplifying calculations, the Gaussian model requires us to store just the mean and variance as the statistics for each color. This reduces the memory requirements and also makes the learning process feasible to execute on mobile robots with constrained processing power.

For this 3D Gaussian model, the *apriori* probability density functions (color $\omega \in [0, N-1]$) are given by:

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

where, $c_i \in [C_{i_{min}} = 0, C_{i_{max}} = 255]$ 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 standard deviations.

Assuming equal priors, each color's *aposteriori* probability is then given by:

$$p(\omega|c_1, c_2, c_3) \propto p(c_1, c_2, c_3|\omega) \quad (3)$$

The Gaussian model for color distributions works inside the lab. In addition, it generalizes well with limited samples when the color distributions are actually unimodal; it is able to handle minor illumination changes. However, in settings outside the lab, factors such as shadows and larger illumination changes cause the color distributions to be multi-modal. The robot is now unable to model colors properly using Gaussians.

Color histograms provide an excellent alternative when colors have multi-modal distributions in the color space [16]. Here, the possible color values (0–255 along each channel) are discretized into a specific number of bins that store the count of pixels that map into that bin. The 3D histogram of a color can be normalized (values in the bins sum to 1) to provide the equivalent of the probability density function (Equation 2):

$$p(c_1, c_2, c_3|\omega) \equiv \frac{Hist_\omega(b_1, b_2, b_3)}{SumHistVals} \quad (4)$$

where b_1, b_2, b_3 represent the histogram bin indices corresponding to the color channel values c_1, c_2, c_3, and $SumHistVals$ is the sum of the values in all the bins of the histogram for that color. The *aposteriori* probabilities for each color are then given by Equation 3.

Unfortunately, histograms do not generalize well with limited training data, especially for samples not observed in the training set, such as with minor illumination changes. Constrained computational and memory resources prevent the implementation of operations more sophisticated than smoothing. Also, they require more storage, which would be wasteful for colors that can be modeled as Gaussians. We propose to combine the two representations such that they complement each other: *colors for which a 3D Gaussian is not a good fit are modeled using 3D histograms*. The decision is made online by the robot, for each color, based on image pixel samples.

Samples for which a 3D Gaussian is a bad fit can still be modeled analytically using other distributions (e.g. mixture of Gaussians, Weibull) through methods such as Expectation-Maximization [5]. But most of these methods do not offer an efficient parameter estimation scheme that can be implemented to work in real-time on mobile robots. Hence, we use a hybrid representation with Gaussians and histograms.

B. Experimental Platform

The SONY *ERS-7* Aibo is a four legged robot with a CMOS camera, providing the robot with a limited view (56.9^o horz., 45.2^o vert.) of its environment. The images, captured in the *YCbCr* format at $30Hz$ with a resolution of 208×160 pixels, possess common defects such as noise and distortion. The robot has 20 degrees-of-freedom, three in each leg, three in its head, and a total of five in its tail, mouth, and ears. It has noisy touch sensors, IR sensors, and wireless LAN for inter-robot communication. The legged (as opposed to wheeled) locomotion results in jerky camera motion.

The RoboCup Legged League is a research initiative in which teams of four robots play a competitive game of soccer on an indoor field of size $\approx 4m \times 6m$ (see Figure 1).

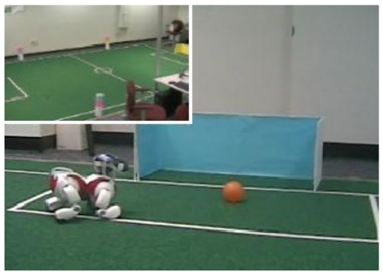
Fig. 1: An Image of the Aibo and the field.

Visual processing on the robot typically begins with an off-board training phase that generates the color map from the space of $128 \times 128 \times 128$ possible pixel values[1] to one of the colors that appear in its environment (pink, yellow, blue, orange, red, dark blue, white, green, and black). Almost all known approaches in this scenario (Section V) produce the color map by hand-labeling several ($\approx 20 - 30$) images over a period of at least an hour. This map is used to segment the images and construct connected constant-colored *regions* out of the segmented images. The regions are used to detect useful objects (e.g. markers and the ball). The robot uses the markers to localize and coordinates with its team-mates 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 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 [11]. This puts added emphasis on learning and adapting the color map in a short period of time.

III. Algorithm

Algorithm 1 describes a method by which the robot *autonomously plans* to *learn* the colors in its environment using the known positions of color-coded objects. Underlined function names are described below.

Our previous algorithm [15] (lines $11, 12, 17-20$) had the robot learn colors by moving along a prespecified motion sequence, and modeled each color as a 3D Gaussian. This fails to work outside the controlled setting of the lab because some

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

Algorithm 1 Planned Autonomous General Color Learning

```
Require: Known initial pose (can be varied across trials).
Require: Color-coded model of the robot's world - objects at
    known positions, which can change between trials.
Require: Empty Color Map; List of colors to be learned -
    $Colors$.
Require: Arrays of colored *regions*, rectangular shapes in 3D;
    $Regions$. A list for each color, consisting of the properties
    (size, shape) of the regions of that color.
Require: Ability to navigate to a target pose $(x, y, \theta)$.
1: $i = 0, N = MaxColors$
2: $Time_{st} = CurrTime$, $Time[]$ — the maximum time
   allowed to learn each color.
3: while $i < N$ do
4:    $Color$ = BestColorToLearn( $i$ );
5:    $TargetPose$ = BestTargetPose( $Color$ );
6:    $Motion$ = RequiredMotion( $TargetPose$ )
7:    Perform $Motion$ {Monitored using visual input and
      localization}
8:    if TargetRegionFound( $Color$ ) then
9:       Collect samples from the candidate region,
         $Observed[][3]$.
10:      if PossibleGaussianFit($Observed$) then
11:         LearnGaussParams( $Colors[i]$ )
12:          *Learn* Mean *and* Variance *from samples*
13:      else { 3D Gaussian not a good fit to samples }
14:         LearnHistVals( $Colors[i]$ )
15:          *Update the color's 3D histogram using the sam-
             ples*
16:      end if
17:      UpdateColorMap()
18:      if !Valid( $Color$ ) then
19:         RemoveFromMap( $Color$ )
20:      end if
21:   else
22:      Rotate at target position.
23:   end if
24:   if $CurrTime - Time_{st} \geq Time[Color]$ or
      $RotationAngle \geq Ang_{th}$ then
25:      $i = i + 1$
26:      $Time_{st} = CurrTime$
27:   end if
28: end while
29: Write out the color statistics and the Color Map.
```

color distributions are now multi-modal and can no longer be modeled as Gaussians. The current algorithm significantly extends the previous approach in two ways. It automatically chooses between two representations for each color to allow color learning outside the lab and also *automatically* generates the motion sequence suitable for learning colors for any given starting pose and object configuration.

The robot starts off at a known field location without any color knowledge. It has a list of colors ($Colors[]$) to be learned and a list of object descriptions ($Regions[][]$) corresponding to each color (size, shape, location). Both the robot's starting pose and the object locations can be varied between trials, which causes the robot to also modify the list of candidate regions for each color. Note that we are not entirely removing the human input. Instead of providing a color map and/or the motion sequence each time the environment or the illumination conditions change, we now just provide the positions of various objects in the robot's world. In many applications, particularly when object locations change less frequently than illumination, this is more efficient than hand-labeling several images.

Due to the inaccuracy of the motion model and the initial lack of visual information, geometric constraints on the position of objects in the robot's environment are essential to resolve conflicts that may arise during the learning process. To generate the motion sequence, the robot needs to make two decisions: the order in which the colors are to be learned and the best candidate object for learning a particular color. The algorithm currently makes these decisions *greedily* and heuristically, i.e. it makes these choices one step at a time. The aim is to get to a large enough target object while moving as little as possible, especially when not many colors are known. The robot computes three weights for each object-color combination (c, i):

$$w_1 = f_d(\ d(c,i)\ ),\ w_2 = f_s(\ s(c,i)\ ),\ w_3 = f_u(\ o(c,i)\ ) \tag{5}$$

where the functions $d(c, i)$, $s(c, i)$ and $o(c, i)$ represent the distance, size and object description for each color-object combination in the robot's world. The function $f_d(\ d(c,i)\ )$ assigns a smaller weight to distances that are large, $f_s(\ s(c,i)\ )$ assigns larger weights to larger candidate objects, while $f_u(\ o(c,i)\ )$ assigns larger weights *iff* the particular object (i) for a particular color (c) is unique, i.e. it can be used to learn the color without having to wait for any other color to be learned.

The *BestColorToLearn* (line 4) is chosen as:

$$arg \max_{c\in[0,9]} \Big(\max_{i\in[0,N_c-1]} \big(f_d(\ d(c,i)\ ) + f_s(\ d(c,i)\ ) + f_u(\ o(c,i)\ ) \big) \Big) \tag{6}$$

where the robot parses through the different objects available for each color (N_c) and calculates the weights. The functions are currently experimentally determined based on the relative importance of each factor, though once estimated they work across different environments. One future research direction is to estimate these functions automatically as well.

Once a color is chosen, the robot determines the best target for the color, using the minimum motion and maximum size constraints:

$$arg \max_{i\in[0,N_c-1]} \Big(f_d(\ d(c,i)\ ) + f_s(\ d(c,i)\ ) + f_u(\ o(c,i)\ ) \Big) \tag{7}$$

For a chosen color, the best candidate object is the one that provides the maximum weight for the given heuristic functions.

Next, the robot calculates the *BestTargetPose()* (line 5) to learn from this target object. It then determines (*Required-Motion()* – line 6) and executes the motion sequence to get there. The current knowledge of colors is used to recognize objects, localize using Particle Filtering [14], and provide *visual feedback* for the motion.

Once it gets close to the target location, the robot searches for candidate image regions satisfying a set of constraints based on current robot location and target object description. If a suitable image region is found (*TargetRegionFound()* – line 8), the robot stops with the region at the center of its visual field, and uses the pixel values in the region as *verification samples*, $Observed$, to verify quality of fit with a 3D Gaussian (*PossibleGaussianFit()* – line 10). We use Bootstrapping [6] with the KL-divergence measure as described in Algorithm 2.

Algorithm 2 PossibleGaussianFit(), Line 10, Algorithm 1

```
1: Determine Maximum-likelihood estimate of Gaussian parameters from samples, $Observed$.
2: Draw N samples from Gaussian – $Estimated$, N = size of $Observed$.
3: $Dist = KLDist(Observed, Estimated)$.
4: Mix $Observed$ and $Estimated$ to get $Data$, 2N items.
5: for $i = 1$ to $NumTrials$ do
6:     Sample N items with replacement from $Data$ – $Set_1$, remaining items – $Set_2$.
7:     $Dist_i = KLDist(Set_1, Set_2)$
8: end for
9: Goodness-of-fit by p-value: where $Dist$ lies in the distribution of $Dist_i$.
```

If the 3D Gaussian is a good fit, the pixels in the candidate region are used to compute the *mean* and *variance* of the 3D Gaussian representing this color using *LearnGaussParams()* (line 11). If not, the candidate pixels are used to populate a 3D histogram using *LearnHistVals()* (line 14). The learned distributions are used to generate the *Color Map*, the mapping from pixel values to color labels. Each cell in the color map is assigned a label corresponding to the color which has the largest *aposteriori probability* (Equation 3) for that set of pixel values. This computationally intensive part of the learning process is performed only once every five seconds or so. The updated map is used to segment subsequent images and detect objects. This helps validate the learned parameters (lines 18, 19) and helps the robot *localize* to suitable locations to learn the other colors. Essentially, our algorithm *bootstraps*, the knowledge available at any instant being exploited to plan and execute the subsequent tasks efficiently.

If the candidate region is not found, it is attributed to slippage and the robot turns in place, searching for the candidate region. If the robot has turned for more than a threshold angle ($Ang_{th} = 360^o$) and/or has spent more than a threshold amount of time on a color ($Time[Color] \approx 20sec$), it transitions to the next color in the list. A video of the color learning process and images at various intermediate stages can be viewed online: *www.cs.utexas.edu/users/AustinVilla/?p=research/auto_vis*.

IV. Experimental Setup and Results

We are concerned with both the color learning and the planning components of the algorithm. We hypothesized that the hybrid color learning scheme should allow the robot to automatically choose the best representation for each color and learn colors efficiently both inside and outside the lab. Our goal is for the hybrid representation to work outside the lab while not resulting in a reduction in accuracy in the controlled lab setting. We proceeded to test that as follows.

We first compared the two color representations, Gaussians (*AllGauss*) and Histograms (*AllHist*), for all the colors, inside the controlled setting of the lab. We quantitatively compared the two color maps with the labels provided by a human observer, over ≈ 15 images. Since most objects of interest are on or slightly above the ground (objects above the horizon are automatically discarded), only suitable image regions were hand-labeled (on average 6000 of the total 33280 pixels). The average classification accuracies for *AllHist* and *AllGauss* were 96.7 ± 0.85 and 97.1 ± 1.01 while the corresponding storage requirements were $3000Kb$ and $0.15Kb$. Note that, qualitatively, *AllHist* performs as well as *AllGauss* but requires more storage (Figure 2).

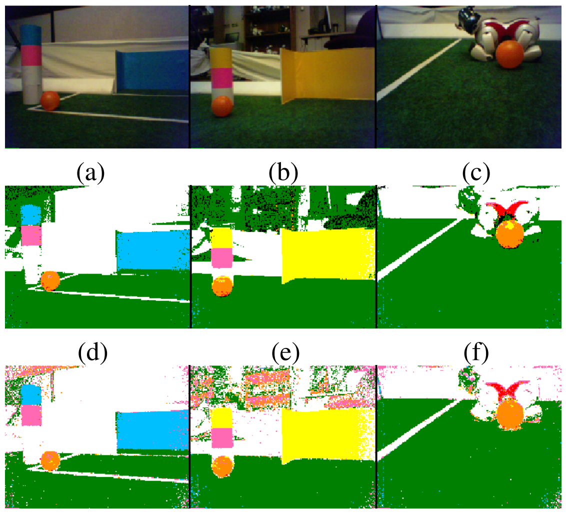

Fig. 2: Images inside the lab. (a)-(c) Original, (d)-(f) $AllGauss$, (g)-(i) $AllHist$. Note that $AllHist$ performs as well as $AllGauss$.

A main goal of this work is to make it applicable to less-controlled settings. We tested the robot in two indoor corridors with overhead fluorescent lamps placed a constant distance apart, resulting in non-uniform illumination conditions and a lot of highlights and shadows on the objects and the floor. In the first corridor, the floor was non-carpeted and of a similar color as the walls. As a result of the non-uniform illumination the floor and the walls had multi-modal color distributions. *AllGauss* could not determine a suitable representation for the ground/walls, causing problems with finding candidates for the other colors (see Figure 3).
With the hybrid color representation, *GaussHist*, the robot, based on the statistical tests, ended up modeling one color

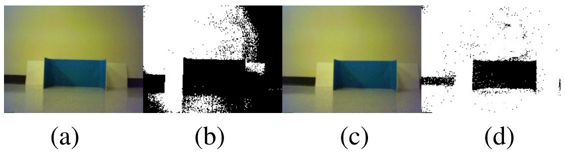
Fig. 3: Segmentation using: (a)-(b) 3D Gaussians, (c)-(d) 3D Histograms. Gaussians do not model ground/walls well but Histograms do.

(wall and ground) as histogram and the other colors as Gaussians. Figure 4 compares *AllHist* with *GaussHist*.

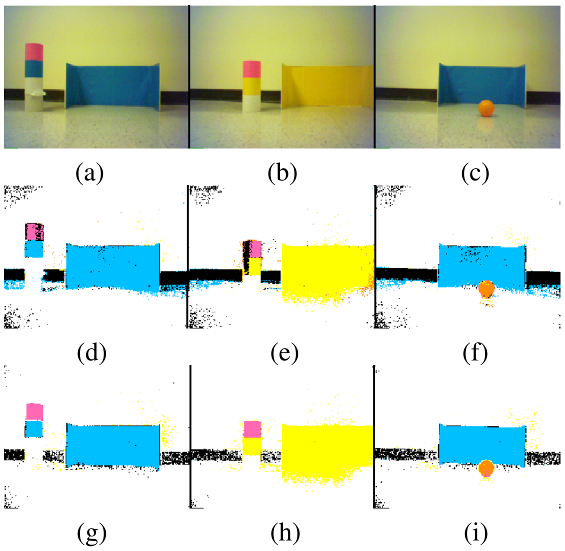
Fig. 4: Images outside the lab: (a)-(c) Original, (d)-(f) $AllHist$, (g)-(i) $GaussHist$. $GaussHist$ performs better under minor illumination changes.

The *AllHist* model does model the ground color better. But histograms require more storage and do not generalize well to minor illumination changes (errors in row 2 of Figure 4), causing problems in resolving conflicts between overlapping colors. The robot is unable to identify suitable candidate regions leading to false positives. With Gaussians, the robot has the option of varying the spread of the known overlapping colors. Hence *GaussHist* lets the robot learn all the colors using the good features of both models. Sample images at: *www.cs.utexas.edu/users/AustinVilla/?p=research/auto_vis*.

Next, we ran the algorithm in a different corridor, where the floor had a patterned carpet with varying shades. The illumination resulted in multi-modal distributions for the color of ground and walls. *AllGauss* did not model these colors well and *AllHist* had problems with the inevitable minor illumination variations during testing. But *GaussHist* was able to learn all the desired colors.

Table I documents some numerical results. The storage requirements reflect the number of colors represented as histograms instead of Gaussians. Sample images can be seen online:*www.cs.utexas.edu/~AustinVilla/?p=research/auto_vis*. We also provide images to show that the planned color learning scheme can be applied to different illumination conditions and can handle re-paintings - changing all *yellow* objects to *white* and vice versa poses no problem.

Type	Accuracy (%)	(KB)
$AllHist-1$	89.53 ± 4.19	3000
$GaussHist-1$	97.13 ± 1.99	440
$AllHist-2$	91.29 ± 3.83	3000
$GaussHist-2$	96.57 ± 2.47	880

TABLE I: Accuracies and storage requirements of models in two different indoor corridors. The results are statistically significant.

One challenge in experimental methodology was to measure the robot's planning capabilities in qualitatively *difficult* setups (objects configurations and robot's initial position). We described our algorithm to seven graduate students with experience working with the robots and asked them to pick a few test configurations which they thought would challenge the algorithm. For each configuration, we measured the number of successful learning attempts: an attempt is deemed a success if the five colors needed for localization are learned.

Config	Success (%)	Localization Error		
		X (cm)	Y (cm)	θ (deg)
Worst	70	17	20	20
Best	100	3	5	0
avg	90 ± 10.7	8.6 ± 3.7	13.1 ± 5.3	9 ± 7.7

TABLE II: Successful Planning and Localization Accuracy.

Table II tabulates the performance of the robot over 15 configurations, with 10 trials for each configuration. It also shows the localization accuracy of the robot using the learned color map. The robot is able to plan its motion sequence and learn colors in most of the configurations that are designed to be adversarial. The corresponding localization accuracy is comparable to that obtained with the hand-labeled color map ($\approx 6cm, 8cm, 4deg$ in X, Y, and θ).

One configuration where the robot performs worst is shown in Figure 5. Here, it is forced to move a large distance to obtain its first color-learning opportunity (from position 1 to 2). This sometimes leads the robot into positions quite far away from its target location (position 2) and it is then unable to find any candidate image region that satisfies the constraints for the target. Currently, failure in the initial stages strands the robot without any chance of recovery: a suitable recovery mechanism is an important area for future work. The failure is largely due to external factors such as slippage: the color-learning plan generated by the robot is quite reasonable. A video of the robot using a learned color map to localize in an indoor corridor can be viewed online: *www.cs.utexas.edu/users/AustinVilla/?p=research/gen_color*.

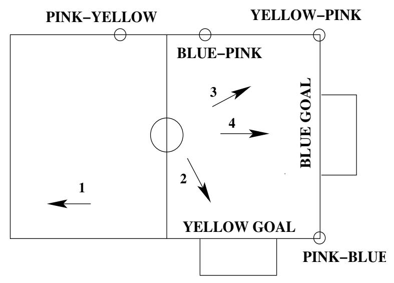

Fig. 5: Sample Configuration where robot performs worst.

V. Related Work

Color segmentation is a well-researched field in computer vision with several effective algorithms [4], [7]. Attempts to learn colors or make them independent to illumination changes have produced reasonable success [8], [9]. But these approaches either involve computations infeasible to perform on mobile robots which typically have constrained resources and/or require the knowledge of the spectral reflectances of the objects under consideration.

On Aibos, the standard approaches for creating mappings from the YCbCr values to the color labels [2], [3], [18] require hand-labeling of several images over an hour or more. Attempts to automatically learn the color map have rarely been successful. In one approach, edges are detected, closed figures are constructed corresponding to known environmental features and the color information from these regions is used to build color classifiers [1]. This approach is time consuming even with the use of offline processing and requires human supervision. In another approach, a color map is learned using three layers of color maps, with increasing precision levels; colors being represented as cuboids [10]. The generated map is not as accurate as the hand-labeled one and additional higher level constraints during the object recognition phase are required to disambiguate the colors. Schulz and Fox [13] estimate colors using a hierarchical Bayesian model with Gaussian priors and a joint posterior on robot position and environmental illumination. Ulrich and Nourbakhsh [17] recognize obstacles by modeling the ground using color histograms and assuming non-ground regions to represent obstacles.

Our prior work [15] enabled the robot to autonomously learn the color map, modeling colors as Gaussians. Here, we present a novel approach that uses a *hybrid representation* for color, works online with *no prior knowledge of color* by *planning* a suitable motion sequence, and enables the robot to learn colors and localize both inside the lab and in less controlled environments.

VI. Conclusions

Color segmentation is a challenging problem, even more so on mobile robots that typically have constrained processing and memory resources. In our prior work [15] we had presented an algorithm to learn colors autonomously within 5 minutes, in the controlled setting of the lab, modeling colors as 3D Gaussians. The hybrid representation for color distributions presented in this paper enables the robot to autonomously learn colors and localize in uncontrolled indoor settings, while maintaining the efficiency in the constrained lab environment. We have also provided a scheme for the robot to autonomously generate the appropriate motion sequence based on the world model so that it simultaneously learns colors and localizes. The color map provides segmentation and localization accuracy comparable to that obtained by previous approaches. The algorithm is dependent only on the structure inherent in the environment and can be quickly repeated if a substantial variation in illumination is noticed. The robot could automatically detect the changes in illumination and adapt to them without human intervention. We are also working on making the robot learn the colors from any *unknown* location in its environment.

The results indicate that the robot should be able to learn the colors even in a natural outdoor setting as long as reasonable illumination is available. We use colors as the distinctive features. But in environments where features aren't constant-colored, other representations such as SIFT [12] could be used. As long as the *locations* of the features are as indicated in the world model, the robot can robustly re-learn how to detect them. This flexibility could be exploited in applications such as surveillance where multiple robots patrol the corridors. Ultimately, we aim to develop efficient algorithms for a mobile robot to function autonomously under completely uncontrolled natural lighting conditions.

Acknowledgment

Special thanks to Suresh Venkat for his helpful discussions on the color learning experiments. The authors would also like to thank the members of the UT Austin Villa team. This work was supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

References

[1] D. Cameron and N. Barnes. Knowledge-based autonomous dynamic color calibration. In *The Seventh International RoboCup Symposium*, 2003.

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

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

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Publishers, 2nd edition, 2000.

[6] B. Efron and R. J. Tibshirani. *An Introduction to Bootstrap*. Chapman and Hall Publishers, 1993.

[7] B. Sumengen et. al. Image segmentation using multi-region stability and edge strength. In *ICIP*, 2003.

[8] Y. B. Lauziere et. al. Autonomous physics-based color learning under daylight. In *Conf. on Color Techniques and Polarization in Industrial Inspection*, 1999.

[9] T. Gevers and A. W. M. Smeulders. Color based object recognition. *In Pattern Recognition*, 32(3):453–464, 1999.

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

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

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

[13] D. Schulz and D. Fox. Bayesian color estimation for adaptive vision-based robot localization. In *IROS*, 2004.

[14] M. Sridharan, G. Kuhlmann, and P. Stone. Practical vision-based monte carlo localization on a legged robot. In *The International Conference on Robotics and Automation*, April 2005.

[15] M. Sridharan and P. Stone. Autonomous color learning on a mobile robot. In *AAAI*, 2005.

[16] M. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[17] I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *AAAI*, 2000.

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