In *The International Joint Conference on Artificial Intelligence* (IJCAI 07),
Hyderabad, India, January 2007.

# Color Learning on a Mobile Robot: Towards Full Autonomy under Changing Illumination

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

## Abstract

A central goal of robotics and AI is to be able to deploy an agent to act autonomously in the real world over an extended period of time. It is commonly asserted that in order to do so, the agent must be able to *learn* to deal with unexpected environmental conditions. However an *ability* to learn is not sufficient. For true extended autonomy, an agent must also be able to recognize *when* to abandon its current model in favor of learning a new one; and *how* to learn in its current situation. This paper presents a fully implemented example of such autonomy in the context of color map learning on a vision-based mobile robot for the purpose of image segmentation. Past research established the ability of a robot to learn a color map in a single fixed lighting condition when manually given a "curriculum," an action sequence designed to facilitate learning. This paper introduces algorithms that enable a robot to i) devise its own curriculum; and ii) recognize when the lighting conditions have changed sufficiently to warrant learning a new color map.

## 1 Motivation

Mobile robotic systems have recently been used in fields as diverse as medicine, rescue, and surveillance [1; 10]. One key enabler to such applications has been the development of powerful sensors such as color cameras and lasers. However, with these rich sensors has come the need for extensive sensor calibration, often performed manually, and usually repeated whenever environmental conditions change significantly.

Here, we focus on the visual sensor (camera), arguably the richest source of sensory information. One important subtask of visual processing is *color segmentation*: mapping each image pixel to a color label. Though significant advances have been made in this field [3; 6], most of the algorithms are computationally expensive to implement on a mobile robot and/or involve a time consuming off-line preprocessing phase. Furthermore, the resulting segmentation is typically quite sensitive to illumination variations. A change in illumination could require a repetition of the entire training phase.

Past research established that a robot can efficiently train its own color map based on knowledge of the locations of colored objects in the environment, but only when manually given a sequence of actions to execute while learning (a *curriculum*) [19]. Separately, it has been shown that a robot can recognize illumination changes and switch among color maps at appropriate times, given a fixed set of pre-trained color maps [18]. The prior work was also limited to controlled environments with only solid-colored objects.

This paper significantly extends these results by enabling a robot i) to recognize when the illumination has changed sufficiently to require a completely new color map rather than using one of the existing ones; and ii) to plan its own action sequence for learning the new color map on-line. Furthermore, we introduce a hybrid color-map representation that enables the robot to learn in less controlled environments, including those with textured surfaces. All algorithms run in real-time on the physical robot enabling it to operate autonomously in an uncontrolled environment with changing illumination over an extended period of time.

## 2 Problem Specification

Here, we formulate the problem and describe our solution. Section 2.1 presents the hybrid color-map representation used for autonomous color learning. Section 2.2 describes our approach to detecting significant illumination changes.

### 2.1 What to learn: Color Model

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

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

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

We start out modeling each color as a three-dimensional (3D) Gaussian with mutually independent color channels. Using empirical data and the statistical technique of bootstrapping [5], we determined that this representation closely approximates reality. The Gaussian model simplifies calculations and stores just the mean and variance as the statistics for each color, thereby reducing the memory requirements and making the learning process feasible to execute on mobile robots with constrained resources.

The *a priori* probability density functions (color $\omega \in [0, N-1]$) are then given by:

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

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 $\mu_{C_i}$ and $\sigma_{C_i}$ represent the corresponding means and standard deviations.

Assuming equal priors ($P(\omega) = 1/N,\ \forall\omega \in [0, N-1]$), each color's *a posteriori* probability is then given by:

$$p(\omega|c_1, c_2, c_3) \propto p(c_1, c_2, c_3|\omega) \tag{3}$$

The Gaussian model for color distributions, as described in our previous work [19], performs well 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 illumination variations cause the color distributions to be multi-modal; the robot is now unable to model colors properly using Gaussians.

In order to extend the previous work to less controlled settings, we propose a hybrid color representation that uses Gaussians and color histograms. Histograms provide an excellent alternative when colors have multi-modal distributions [20]. Here, the possible color values (0–255 along each channel) are discretized into bins that store the count of pixels that map into that bin. A 3D histogram can be normalized to provide the probability density function:

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

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 $SumHistVals_\omega$ is the sum of the values in all the bins of the histogram for color $\omega$. The *a posteriori* probabilities 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. Resource constraints prevent the implementation of operations more sophisticated than smoothing. Also, histograms require more storage, wasteful for colors that can be modeled as Gaussians. We 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 goodness-of-fit decision is made online, for each color.

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 [4]. But most of these methods involve parameter estimation schemes that are computationally expensive to perform on mobile robots. Hence, we use a *hybrid* representation with Gaussians and histograms that works well and requires inexpensive computation. In addition, the robot automatically generates the *curriculum* (action sequence) based on the object configuration, as described in Section 3.

## 2.2 When to learn: Detecting illumination changes

To detect significant changes in illumination, we need a mechanism for representing illumination conditions and for differentiating between them.

We hypothesized that images from the same lighting conditions would have measurably similar distributions of pixels in color space. The original image is in the YCbCr format, with values ranging from [0-255] along each dimension. To reduce storage, but still retain the useful information, we transformed the image to the normalized RGB space, $(r, g, b)$:

$$r = \frac{R+1}{R+G+B+3}, \quad g = \frac{G+1}{R+G+B+3}, \quad b = \frac{B+1}{R+G+B+3} \tag{5}$$

Since $r + g + b = 1$, any two of the three features are a sufficient statistic for the pixel values. We represent a particular illumination condition with a set of distributions in $(r, g)$ space, quantized into $N$ bins in each dimension, corresponding to several images captured by the robot.

Next, we need a well-defined measure capable of detecting the correlation between discrete distributions. Based on experimental validation, we use *KL-divergence*, an entropy-based measure. For two distributions A and B in the 2D $(r, g)$ space, $N$ being the number of bins along each dimension:

$$KL(A, B) = -\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(A_{i,j} \cdot ln\frac{B_{i,j}}{A_{i,j}}) \tag{6}$$

The more similar two distributions are, the smaller is the KL-divergence between them. Since KL-divergence is a function of the log of the observed color distributions, it is reasonably robust to large peaks in the observed color distributions and is hence less affected by images with large amounts of a single color. The lack of symmetry in KL-divergence is eliminated using the Resistor-Average KL-divergence (RA-KLD) [8].

Given a set of distributions corresponding to $M$ different illumination conditions, we have previously shown [18] that it is possible to effectively classify the distribution corresponding to a test image into one of the illumination classes. A major limitation was that we had to know the illumination conditions in advance and also had to provide manually trained color maps for each illumination. Here, we make a significant extension in that *we do not need to know the different illumination conditions ahead of time*.

For every illumination condition $i$, in addition to a set of $(r, g)$ distributions ($rg_{samp}[i]$), we calculate the RA-KL distances between every pair of $(r, g)$ distributions to get a distribution of distances, ($D_i$), which we model as a Gaussian. When the illumination changes significantly, the average RA-KL distance between a test $(r, g)$ distribution and $rg_{samp}[i]$ maps to a point well outside the $95\%$ range of the intra-illumination distances ($D_i$). This feature is used as a measure of detecting a change in illumination.

# 3 Algorithms: When, What, How to Learn

Our algorithms for color learning and adaptation to illumination change are summarized in Algorithm 1 and Algorithm 2.

Algorithm 1 enables the robot to decide *when to learn*. The robot first learns the color map for the current illumination by generating a curriculum using the world model, as described in Algorithm 2. Next, it represents this illumination condition

**Algorithm 1** Adapting to Illumination Change – *When to learn?*

**Require:** For each illumination $i \in [0, M-1]$, color map and distribution of RA-KLD distances $D_i$.
1: Begin: $M = 0$, $current = M$.
2: Generate *curriculum* and learn all colors - Algorithm 2.
3: Generate $rg_{samp}[current][]$, $N$ $(r, g)$ space distributions, and distribution of RA-KLD distances, $D_{current}$.
4: Save color map and image statistics, $M = M + 1$.
5: **if** $currentTime - testTime \geq time_{th}$ **then**
6: $rg_{test} =$ sample $(r, g)$ test distribution.
7: **for** $i = 0$ to $M - 1$ **do**
8: $dAvg_{test}[i] = \frac{1}{N}\sum_j KLDist(rg_{test}, rg_{samp}[i][j])$
9: **end for**
10: **if** $dAvg_{test}[current]$ lies within the threshold range of $D_{current}$ **then**
11: Continue with current color map.
12: **else if** $dAvg_{test}[i]$ lies within the range of $D_i, i \neq current$ **then**
13: Use corresponding color map, $current = i$.
14: **else if** $\forall i \in [0, M-1], dAvg_{test}[i]$ lies outside the range of $D_i$ **then**
15: Re-learn color map autonomously: Algorithm 2.
16: Save $(r, g)$ distributions for new illumination.
17: Transition to the new color map for subsequent operations.
18: $current = M, M = M + 1$.
19: **end if**
20: $testTime = currentTime$.
21: **end if**

by collecting sample image distributions in $(r, g)$ and computing the distribution of RA-KL distances, $D_{currIll}$.

Periodically ($time_{th} = 0.5$), the robot generates a test distribution, $rg_{test}$, and computes its average distance to each set of previously stored distributions, $rg_{samp}[i]$. If $dAvg_{test}[i]$ lies within the threshold range (95%) of the corresponding $D_i$, the robot transitions to the corresponding illumination condition. But, if it lies outside the threshold range of all known distribution of distances, the robot learns a new color map and collects image statistics, which are used in subsequent comparisons. Changing the threshold changes the resolution at which the illumination changes are detected but the robot is able to handle minor illumination changes using the color map corresponding to the closest illumination condition (see Section 4.2). With transition thresholds to ensure that a change in illumination is accepted *iff* it occurs over a few frames, it also smoothly transitions between the learned maps. The algorithm requires no manual supervision.

Next, we briefly describe the planned color learning algorithm, Algorithm 2, used in lines 2 and 15 of Algorithm 1. Our previous algorithm [19] (lines $11, 12, 17-20$) had the robot move along a prespecified motion sequence, and model each color as a 3D Gaussian. But, outside the controlled lab setting, some color distributions are multi-modal and cannot be modeled effectively as Gaussians. The current algorithm significantly extends the previous approach in two ways. It automatically chooses between two representations for each

**Algorithm 2** Autonomous Color Learning – *How to learn?*

**Require:** Known initial pose and color-coded model of the robot's world - objects at known positions. These can change between trials.
**Require:** Empty Color Map; List of colors to be learned.
**Require:** Arrays of colored *regions*, rectangular shapes in 3D; A list for each color, consisting of the properties (size, shape) of the regions of that color.
**Require:** Ability to *approximately* 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 samples*
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 to facilitate color learning outside the lab: it decides *what to learn*. It also automatically determines *how to learn*, i.e. it generates the *curriculum* for learning colors, for any robot starting pose and object configuration.

The robot starts off at a known location without any color knowledge. It has a list of colors to be learned and a list of object descriptions corresponding to each color (size, shape, location of regions). Though this approach does require some human input, in many applications, particularly when object locations change less frequently than illumination, it is more efficient than hand-labeling several images. To generate the curriculum, the robot has to decide 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 without actually planning for the subsequent steps. 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 $(c, i)$:

$$w_1 = f_d(\ d(c,i)\ ),\ w_2 = f_s(\ s(c,i)\ ),\ w_3 = f_u(\ o(c,i)\ ) \tag{7}$$

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. Function $f_d(\ d(c, i)\ )$ assigns larger weights to smaller distances, $f_s(\ s(c, i)\ )$ assigns larger weights to larger candidate objects, and $f_u(\ o(c, i)\ )$ assigns larger weights *iff* the object $i$ can be used to learn color $c$ without having to wait for any other color to be learned or object $i$ consists of color $c$ and other colors that have already been learned.

The *BestColorToLearn()* (line 4) is then given by:

$$arg \max_{c\in[0,9]} \Big( \max_{i\in[0,N_c-1]} \big( f_d(\ d(c,i)\ ) + f_s(\ d(c,i)\ ) + f_u(\ o(c,i)\ )\ \big) \Big) \tag{8}$$

where the robot parses through the different objects available for each color ($N_c$) and calculates the weights. Once a color is chosen, the robot determines the best target for the color, using the minimum motion and maximum size constraints:

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

For a chosen color, the best candidate object is the one with the maximum weight for the given heuristic functions. The robot chooses the *BestTargetPose()* (line 5) to learn color from this object and moves there (lines 6,7). It 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 pixels in the region are used as samples, $Observed$, to verify goodness-of-fit with a 3D Gaussian (line 10). The test is done using *bootstrapping* [5] using KL-divergence as the distance measure, as described in Algorithm 3.

If the samples generate a good Gaussian fit, they are used to determine the mean and variance of the color distribution (*LearnGaussParams()* – line 11). If not, they are used to populate a 3D histogram (*LearnHistVals()* – line 14). The learned distributions are used to generate the *Color Map*, the mapping from the pixel values to color labels (line 17). The robot uses the map to segment subsequent images and find objects. The objects help the robot localize to positions suitable for learning other colors, and to validate the learned colors and remove spurious samples (lines 18,19).

To account for slippage and motion model errors, if a suitable image region is not found, the robot turns in place to find it. If it has rotated in place 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. Instead of providing a color map and/or the action sequence each time the environment or the illumination changes, we now just provide the positions of objects in the robot's world and have it plan its *curriculum* and learn colors autonomously. The adaptation to illumination changes makes the entire process autonomous. A video of the robot learning colors can be seen online: *www.cs.utexas.edu/∼AustinVilla/?p=research/auto_vis*.

**Algorithm 3** PossibleGaussianFit(), line 10 Algorithm 2 – *What to learn?*

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

# 4 Experiments

We first provide a brief overview of the robotic platform used, followed by the experimental results.

## 4.1 Experimental Platform

The SONY *ERS-7* Aibo is a four legged robot whose primary sensor is a CMOS camera located at the tip of its nose, with a limited field-of-view ($56.9^o$ horz., $45.2^o$ vert.). The images, captured in the *YCbCr* format at 30Hz with a resolution of $208 \times 160$ 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 IR sensors and wireless LAN for inter-robot communication. The legged as opposed to wheeled locomotion results in jerky camera motion. All processing for vision, localization, motion and action selection is performed on-board using a 576MHz processor.

One major application domain for the Aibos is the RoboCup Legged League [15], a research initiative in which teams of four robots play a competitive game of soccer on an indoor field $\approx 4m \times 6m$. But applications on Aibos and mobile robots with cameras typically involve an initial calibration phase, where the color map is produced by hand-labeling images over a period of an hour or more (Section 5). Our approach has the robot autonomously learning colors in *less than five minutes* and adapting to illumination changes.

## 4.2 Experimental Results

We tested our algorithm's ability to answer three main questions: *When to learn* - the ability to detect illumination changes, *How to learn* - the ability to plan the action sequence to learn the colors, and *How good is the learning* - the segmentation and localization accuracy in comparison to the standard human-supervised scheme.

**When to Learn?**

First, we tested the ability to accurately detect changes in illumination. The robot learned colors and $(r, g)$ distributions corresponding to an illumination condition and then moved around in its environment chasing a ball. We changed the lighting by controlling the intensity of specific lamps and the robot identified significant illumination changes.

| (%) | Change | $Change^c$ |
|---|---|---|
| Change | 97.1 | 2.9 |
| $Change^c$ | 3.6 | 96.4 |

**Table 1**: Illumination change detection: few errors in 1000 trials.

Table 1 presents results averaged over 1000 trials with the rows and columns representing the ground truth and observed values respectively. There are very few false positives or false negatives. The errors due to highlights and shadows are removed by not accepting a change in illumination unless it is observed over a few consecutive frames.

To test the ability to transition between known illuminations, the robot learned color maps and statistics for three conditions: *Bright(1600lux)*, *Dark(450lux)*, *Interim(1000lux)*.

| Illum. | Transition Accuracy | |
|---|---|---|
| | Correct (%) | Errors |
| Bright | 97.3 | 4 |
| Dark | 100 | 0 |
| Interim | 96.1 | 6 |

**Table 2**: Illumination transition accuracy: few errors in $\approx 150$ trials.

The intensity of the overhead lamps was changed to one of the three conditions once every $\approx$ 10 sec. Table 2 shows results averaged over $\approx$ 150 trials each. The few false transitions, due to shadows or highlights, are quickly corrected in the subsequent tests. When tested in conditions in between the known ones, the robot finds the closest illumination condition and is able to work in the entire range.

**How to Learn?**

In previous work [19], fixed object locations resulted in a single curriculum to learn colors. To test the robot's ability to generate curricula for different object and robot starting positions, we invited a group of seven graduate students with experience working with the Aibos to suggest challenging configurations. It is difficult to define challenging situations ahead of time but some examples that came up include having the robot move a large distance in the initial stages of the color learning process, and to put the target objects close to each other, making it difficult to distinguish between them. The success ratio and the corresponding localization accuracy over 15 trials are shown in Table 3.

| Config | Success (%) |
|---|---|
| Worst | 70 |
| Best | 100 |
| Avg | $90 \pm 10.7$ |

**Table 3**: Planning Accuracy in challenging configurations.

A trial is a success if all colors are learned successfully. The localization error is the difference between the robot's estimate and the actual target positions, measured by a human using a tape measure. We observe that the robot is mostly able to plan a suitable motion sequence and learn colors. In the cases where it fails, the main problem is that the robot has to move long distances with very little color knowledge. This, coupled with slippage, puts it in places far away from the target location and it is unable to learn the colors. The motion planning works well and we are working on making the algorithm more robust to such failure conditions. The localization accuracy with the learned map is comparable to that with a hand-labeled color map ($\approx 8cm, 10cm, 6deg$ in comparison to $6cm, 8cm, 4deg$ in $X$, $Y$, and $\theta$).

**How good is the learning?**

To test the accuracy of learning under different illuminations, we had the robot learn colors under controlled lab conditions and in indoor corridors outside the lab, where the overhead fluorescent lamps provided non-uniform illumination (between 700-1000lux) and some of the colors (floor, wall etc) could not be modeled well with 3D Gaussians. We observed that the robot automatically selected the Gaussian or Histogram model for each color and successfully learned all the colors.

| Config | Localization Error | |
|---|---|---|
| | Dist (cm) | $\theta$ (deg) |
| Lab | $9.8 \pm 4.8$ | $6.1 \pm 4.7$ |
| Indoor | $11.7 \pm 4.4$ | $7.2 \pm 4.5$ |

**Table 4**: Localization accuracy: comparable to that with a hand-labeled map.

Table 4 shows the localization accuracies under two different illumination conditions (lab, indoor corridor) based on the learned color maps. We had the robot walk to 14 different points and averaged the results over 15 trials. The differences were not statistically significant. The corresponding segmentation accuracies were $95.4\%$ and $94.3\%$ respectively, calculated over 15-20 images, as against the $97.3\%$ and $97.6\%$ obtained with a hand-labeled color map (differences not statistically significant). The learned maps are as good as the hand-labeled maps for object recognition and high-level task competence. But, our technique takes 5 minutes of robot time instead of an hour or more of human effort. Sample images under different testing conditions and a video of the robot localizing in a corridor can be seen online: *www.cs.utexas.edu/∼AustinVilla/?p=research/gen_color*.

To summarize, our algorithm enables the robot to plan its motion sequence to learn colors autonomously for any given object configuration. It is able to detect and adapt to illumination changes without manual training.

## 5 Related Work

Color segmentation is a well-researched field in computer vision with several effective algorithms [3; 6]. Attempts to learn colors or make them robust to illumination changes have produced reasonable success [16; 13]. But they are computationally expensive to perform on mobile robots which typically have constrained resources.

On Aibos, the standard approaches for creating mappings from the YCbCr values to the color labels [7; 11; 12] require hand-labeling of images ($\approx$ 30) over an hour or more. There have been a few attempts to automatically learn the color map on mobile robots. In one approach, closed figures are constructed corresponding to known environmental features and the color information from these regions is used to build color classifiers [2]. The algorithm is time consuming even with the use of offline processing and requires human supervision. In another approach, three layers of color maps, with increasing precision levels are maintained; colors being represented as cuboids [14]. The generated map is not as accurate as the hand-labeled one. Schulz and Fox [17] estimate colors using a hierarchical Bayesian model with Gaussian priors and a joint posterior on robot position and environmental illumination. Ulrich and Nourbakhsh [21] model the ground using color histograms and assume non-ground regions to represent obstacles. Anzani et. al [9] model colors using mixture of Gaussians and compensate for illumination changes by modifying the parameters. But, prior knowledge of color distributions and suitable initialization of parameters are required. Our approach does not require prior knowledge of color distributions. Instead, it uses the world model to automatically learns colors by generating a suitable curriculum, and adapts to illumination changes.

## 6 Conclusions

Robotic systems typically require significant amount of manual sensor calibration before they can be deployed. We aim to make the process more autonomous. We propose a scheme that achieves this goal with regard to color segmentation, an important subtask for visual sensors.

In our previous work [19], the robot learned colors within the controlled lab setting using a pre-specified motion sequence. In other work [18], we demonstrated the ability to transition between discrete illumination conditions when appropriate color maps and image statistics were trained offline. But the robot was given a lot of information manually, including the object positions, the action sequence for learning colors, and color maps for each illumination condition.

With the current method only the object locations need to be specified. A *hybrid representation* for color enables the robot to generate a *curriculum* to learn colors and localize both inside the lab and in much more uncontrolled environments with non-uniform overhead illumination and background clutter that can be confused with the objects of interest. Other robots may use cameras of higher quality but color maps are still needed. For full autonomy there are always computational constraints at some level, irrespective of the robot platform being used. This paper lays the groundwork for the next step of testing the same algorithm on other robot platforms that work outdoors.

In the end, the robot is able to detect changes in illumination robustly and efficiently, *without prior knowledge of the different illumination conditions*. When the robot detects an illumination condition that it had already learned before, it smoothly transitions to using the corresponding color map. Currently, we have the robot re-learn the colors when a significant change from known illumination(s) is detected. One direction of future work is to have the robot adapt to minor illumination changes by suitably modifying specific color distributions. Ultimately, we aim to develop efficient algorithms for a mobile robot to function autonomously under uncontrolled natural lighting conditions.

## Acknowledgment

Special thanks to Suresh Venkat for discussions on the color learning experiments, and to the UT AustinVilla team. This work was supported in part by NSF CAREER award IIS-0237699 and ONR YIP award N00014-04-1-0545.

## References

[1] M. Ahmadi and P. Stone. A multi-robot system for continuous area sweeping tasks. In *ICRA*, 2006.

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

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

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

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

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

[7] D. Cohen et. al. *UPenn TDP, RoboCup-2003: The Seventh RoboCup Competitions and Conferences*. 2004.

[8] D. H. Johnson et. al. Information-theoretic analysis of neural coding. *Journal of Computational Neuroscience*, 2001.

[9] F. Anzani et. al. On-line color calibration in non-stationary environments. In *The International RoboCup Symposium*, 2005.

[10] J. Pineau et. al. Towards robotic assistants in nursing homes: Challenges and results. *RAS Special Issue on Socially Interactive Robots*, 2003.

[11] S. Chen et. al. *UNSW TDP, RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. 2002.

[12] William Uther et al. Cm-pack'01: Fast legged robot walking, robust localization, and team behaviors. In *The Fifth International RoboCup Symposium*, 2001.

[13] T. Gevers and A. W. M. Smeulders. Color based object recognition. *In Pattern Recognition*, 1999.

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

[15] H. Kitano, M. Asada, I. Noda, and H. Matsubara. Robot world cup. *Robotics and Automation*, 1998.

[16] Y. B. Lauziere, D. Gingras, and F. P. Ferrie. Autonomous physics-based color learning under daylight. In *Conf. on Color Techniques and Polarization in Industrial Inspection*, 1999.

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

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

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

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

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