Autonomous Planned Color Learning on a Mobile Robot Without Labeled Data

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 yet integral subtask of mobile robot systems that use visual sensors, especially since they 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.

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 (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 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. But the process is constrained to work within the limited memory and processing resources of the robot and it should be able to cope with the rapid motion of the limited-field-of-view camera, with the associated noise and image distortions.

We build on our previous work (Sridharan & Stone 2005), where the robot learnt 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.

Problem Description

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

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:

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

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

In our previous color learning approach (Sridharan & Stone 2005), each color was modeled as a three-dimensional (3D) Gaussian with mutually independent color channels (no correlation among the values along the color channels). Using empirical data and the statistical technique of bootstrapping (Efron & Tibshirani 1993), we determined that this closely approximates reality. In addition to simplifying calculations, with the Gaussian model, the mean and variance are the only statistics that need to be stored for each color. This reduces the memory requirements and also makes the learning process feasible to execute on the mobile robots with constrained processing power.

For the 3D Gaussian model with independent channels, 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 (no assumptions on specific colors being confined to specific regions in the color space), 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 perfectly inside the lab, and it generalizes well with limited samples when the color distributions are actually unimodal; it is able to handle minor illumination changes. But 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 (Swain & Ballard 1991). 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.

But 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. In this paper, 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) using methods such as Expectation-Maximization (EM). But most of these models/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.

Experimental Platform

The SONY Aibo, *ERS-7*, is a four legged robot whose primary sensor is a CMOS camera located at the tip of its nose, providing the robot with a limited view (56.9^o (hor), 45.2^o (ver)) 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 (dof), 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).

Figure 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 9 different colors that appear in its environment (pink, yellow, blue, orange, red, dark blue, white, green, and black). Almost all known approaches in this scenario (see related work section) 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 itself on the field 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 (Kitano *et al.* 1998). This puts added emphasis on learning and adapting the color map in a short period of time.

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 (Sridharan & Stone 2005) (lines $11, 12, 17-20$) had the robot learn colors by moving along a prespecified motion sequence, and it modeled each color as a 3D Gaussian. This assumption fails to work outside the controlled setting of the lab because all the color distributions can no longer be modeled as Gaussians (they are sometimes multi-modal). The current algorithm automatically chooses between two different representations for each color and also *automatically* generates a motion sequence suitable for learning colors for any given starting pose and object configuration in its world.

The robot starts off at a known pose in its world model with the locations of various color-coded objects known. It has no initial color information (images are segmented *black*). It has two lists: the list of colors to be learnt (*Colors[]* and an array of structures ($Regions[][]$) — a list for each color. Each structure corresponds to an object of a particular color and stores a set of properties, such as its size (length and width) and its three-dimensional location (x,y,z) in the world model. 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. Though this approach does seem to require a fair amount of human input, in many applications, particularly when object locations change less frequently than illumination, it 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.

[1]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 learnt - $Colors[]$.
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, θ).

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.

These heuristic constraints depend on the problem domain. We need to make two decisions in our problem domain: the order in which the colors are to be learnt and then 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. Note that the details of the algorithm and the corresponding heuristics are presented primarily for the replicability of our work. Our main aim is to demonstrate that such autonomous color learning can be accomplished in a setting where it is typically done manually.

In our task domain, the factors that influence these choices are:

1. The amount of motion (distance) that is required to place the robot in a location suitable to learn the color.
2. The size of the candidate region the color can be learnt from.
3. The existence of a region that can be used to learn that color independent of the knowledge of any other (as of yet) unknown color.

These factors are used by the robot in a set of heuristic functions, to learn the colors with minimal motion and increase the chances of remaining well-localized. If a color can be learnt with minimal motion and/or is visible in large quantities around the robot's current location, it should be learnt first. If a maximum-sized region (number of pixels) of a particular color exists, it would be a good candidate for that color. Sometimes a color does not have a maximum sized region and can be learnt more reliably by associating it with another color around it: in our default configuration, *pink* has regions of the same size associated with either *blue* or *yellow*. Here, the robot attempts to learn one of those two colors before it attempts to learn *pink*. The relative importance weights assigned to the individual factors are used to resolve the conflicts, if any, between the factors.

The robot therefore computes three weights for each object-color combination (c, i) in its world:

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

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, while $f_s(\ s(c,i)\ )$ assigns larger weights to larger candidate objects. The function $f_u(\ o(c,i)\ )$ assigns larger weights iff the particular object (i) for a particular color (c) is unique - it can be used to learn the color without having to wait for any other color to be learnt.

The best color to learn (*BestColorToLearn*—line 4) is chosen as:

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

where the robot parses through the different objects available for each color (N_c) and calculates the weights. For each color the object that provides the maximum weight is determined. Then, the color that results in the largest value among these values is chosen to be learnt first. 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-candidate-region* for the color, using the minimum motion and maximum size constraints, based on:

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

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 pose best suited to detect this candidate region – *BestTargetPose()* (line 5). Specifically it attempts to move to a position where the entire candidate object would be in its field of view; based on the known world model. Using its navigation function – *RequiredMotion()* (line 6) – the robot determines and executes the motion sequence to place it at the target position. The current knowledge of colors is used to recognize objects, localize (we use Particle Filtering - see (Sridharan, Kuhlmann, & Stone 2005) for complete details) and provide *visual feedback* for the motion.

Once it gets close to the target location, the robot searches for image regions that satisfy the heuristic shape and size constraints for the candidate region. The structure *Regions[Color][best-candidate-region]* provides the actual world model definitions – the (x,y,z) location, width and height, which are dynamically modified by the robot, based on its pose and geometric principles, to arrive at suitable constraints. The robot stops when either an image region satisfying the constraints is found (*TargetRegionFound()*, line 8) or its pose estimate corresponds to the target position.

If the candidate region is not found, it is attributed to slippage and the robot turns in place, searching for the candidate region. Geometric sanity checks, based on the known locations of the objects in the robot's environment, are used to resolve conflicts with other regions that satisfy the heuristic constraints - the candidate region should satisfy the geometric checks with other known objects in its field of view.

If such a region is found, the robot stops with the region at the center of its visual field, and uses the pixel values (each pixel being a vector with three elements corresponding to the three color channels) in the region as *verification samples*, *Observed[][3]* to check if a 3D Gaussian is a good fit for this color (*PossibleGaussianFit()* — line 10). We use the statistical *bootstrap* technique with KL-divergence (Johnson *et al.* 2001) as the distance measure, as described in Algorithm 2.

If the 3D Gaussian is a good fit, the robot executes *LearnGaussParams()* (line 11). Each pixel of the candidate region (currently *black*) that is sufficiently distant from the *means* of the other known color distributions is selected and the *mean* and *variance* of these pixel values represent the *density function* of the color under consideration. If the 3D Gaussian is not a good fit for the samples, the robot models the color as a 3D histogram, the same candidate pixels now being used to populate the histogram (*LearnHistVals()* — line 14). To suppress the inherent image noise, the robot repeatedly extracts information from several images at the same target pose.

Once the color has been learnt, the function *UpdateColorMap()* (line 17) takes the learnt distributions and gener-

Algorithm 2 Verify goodness-of-fit

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 $Estimated$ – $Data$, 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 $Dist_i$.

ates the mapping from the pixel values to color labels, the *Color Map*. The assignment of color labels to each cell in the $128 \times 128 \times 128$ map, the most computationally intensive part of the learning process, is performed only once every five seconds or so. The histograms are normalized (Equation 4) to make them correspond to the Gaussian density functions. 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.

By definition, Gaussians have a non-zero value throughout the color space. During the learning process, the robot could classify all the color map cells into one of the colors currently included in the map, resulting in no candidate regions for the other colors. So, a cell is assigned a particular color label *iff* its distance from the mean of the corresponding color lies within an integral multiple of the color's standard deviation. Histograms do not have this problem.

The updated map is used to segment subsequent images and detect objects. This helps validate the learnt parameters, *Valid()* (line 18) – if a suitable object is not found, the color's statistics (Gaussian/Histogram) are deleted and it is removed from the map (*RemoveFromMap()* — line 19), and helps the robot *localize* and move to suitable locations to learn the other colors. Our learning algorithm essentially *bootstraps*, the knowledge available at any given instant being exploited to plan and execute the subsequent tasks efficiently.

If the robot has rotated in place (searching for a candidate) 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. We prefer to let the robot turn a complete circle rather than turn a certain angle in each direction to avoid abrupt transitions. The process continues until the robot has attempted to learn all the colors. Then, it saves the color map and the statistics. 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*.

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 and have it plan its motion sequence and learn colors autonomously. The robot can be deployed a lot faster, especially in domains where object locations change less frequently than illumination conditions.

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. Qualitatively, both representations produced similar results (see Figure 2). We then quantitatively compared the two color maps with the labels provided by a human observer, over $\approx$ 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 *AllHist* performs as well as *AllGauss* but requires more storage.

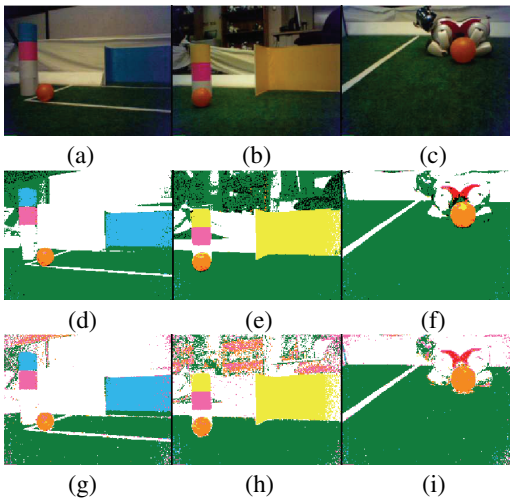
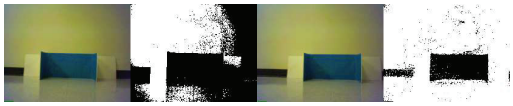


Figure 2: Images inside the lab. (a)-(c) Original, (d)-(f) $AllGauss$, (g)-(i) $AllHist$.

A main goal of this work is to make it applicable to less-controlled settings. We next tested the robot in two indoor corridors, where the natural setting consisted of a series of overhead fluorescent lamps placed a constant distance apart, resulting in non-uniform illumination conditions and a lot of highlights/shadows on the objects and the floor. In the first corridor, the floor was non-carpeted and of a similar color as the walls. The robot was provided with a world model with color-coded objects of interest, but because of the non-uniform illumination the textured floor and the walls had

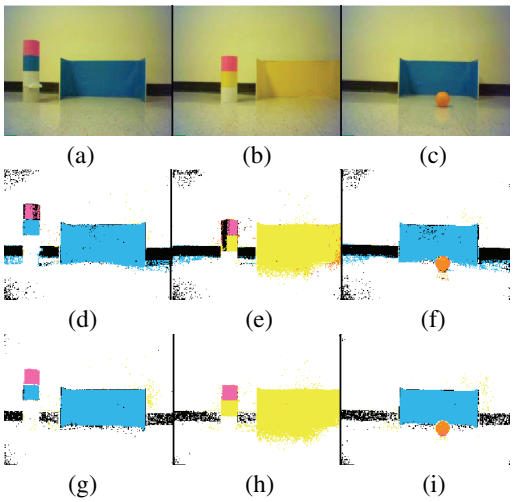
multi-modal color distributions. As a result *AllGauss* could not determine a suitable representation for the ground/walls, causing problems with finding candidates for the other colors (see Figure 3).



(a) (b) (c) (d)

Figure 3: Segmentation using: (a)-(b) 3D Gaussians, (c)-(d) 3D Histograms.

With the hybrid color representation, *GaussHist*, the robot, based on the statistical tests, ended up modeling one color (wall and ground) as histogram and the other colors as Gaussians. Figure 4 compares *AllHist* with *GaussHist*.



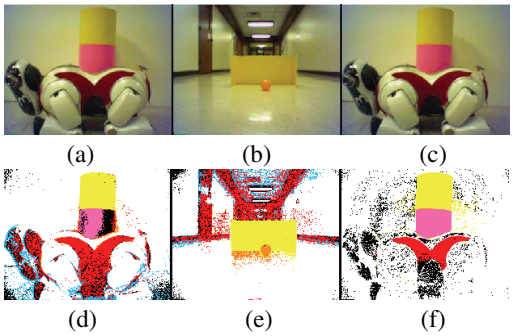
(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 4: Images outside the lab - walls and floors similar. (a)-(c) Original, (d)-(f) $AllHist$, (g)-(i) $GaussHist$.

The *AllHist* model does solve the problem of modeling ground color better. But, while Gaussians are robust to slight illumination changes, histograms, in addition to requiring more storage, do not generalize well to such situations (errors row 2 of Figure 4). This inability to generalize well also causes problems in resolving conflicts between overlapping colors. For example, when the robot attempts to learn *red* (opponent's uniform color) after learning other colors, it is unable to identify a suitable candidate region. As seen in Figure 5, this leads to false positives and bad performance over other colors ((d),(e)).

With Gaussians, the robot has the option of varying the spread of the known overlapping colors, such as *orange* and *pink*. Hence *GaussHist* lets the robot successfully learn the total set of colors using the good features of both models.

Next, we ran the color learning algorithm in a different corridor, where the floor had a patterned carpet with varying shades and the illumination resulted in multi-modal distributions for the ground and the walls. Once again, *AllGauss* did not model the multi-modal color distributions well while



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

Figure 5: Images with opponent color in map: (a)-(c) Original, (d)-(e) $AllHist$, (f) $GaussHist$.

AllHist had problems when faced with minor illumination conditions (inevitable) during testing. But *GaussHist* is able to exploit the advantages of both systems and the robot successfully learns the desired colors.

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 1: Accuracies and storage requirements of models in two different indoor corridors.

Table 1 documents the numerical results for the two situations - the results are statistically significant. The storage requirements reflect the number of colors that were represented as histograms instead of Gaussians. Sample images for this setting 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.

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 each, subject to the constraints listed earlier, 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 all the (five) colors needed for localization are learnt.

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 2: Successful Planning and Localization Accuracy.

Table 2 tabulates the performance of the robot in its plan-

ning task over 15 configurations, with 10 trials for each configuration. It also shows the localization accuracy of the robot using the learnt color map. We observe that the robot is able to plan its color learning task and execute it successfully in most of the configurations (designed to be adversarial) and the corresponding localization accuracy is comparable to that obtained with the hand-labeled color map ($\approx 6cm, 8cm, 4deg$ in X, Y, and θ).

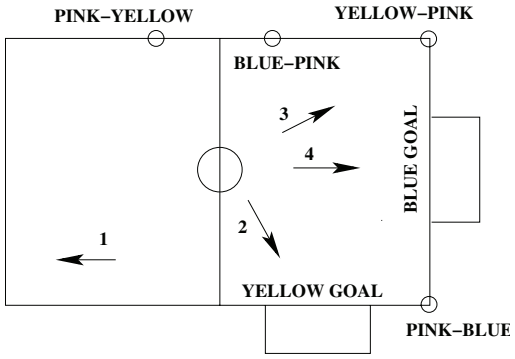


Figure 6: Sample Configuration where robot performs worst.

One configuration where the robot performs worst is shown in Figure 6. 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 yellow goal. Currently, failure in this initial stage strands the robot without any chance of recovery: a suitable recovery mechanism using additional geometric constraints is an important area for future work. Note that 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 learnt color map to localize to points in an indoor corridor can be viewed online: *www.cs.utexas.edu/users/AustinVilla/?p=research/gen_color*.

Related Work

Color segmentation is a well-researched field in computer vision with several effective algorithms (Comaniciu & Meer 2002; Sumengen, Manjunath, & Kenney 2003). Attempts to learn colors or make them independent to illumination changes have produced reasonable success (Lauziere, Gingras, & Ferrie 1999; Gevers & Smeulders 1999). 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 (Uther *et al.* 2001; Chen *et al.* 2002; Cohen *et al.* 2004) require hand-labeling of several images (≈ 30) over an hour or more. Attempts to automatically learn the color map on the Aibos/robots 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 (Cameron & Barnes 2003). This approach is time consuming even with the use of offline processing and requires human supervision. In another approach, a color map is learnt using three layers of color maps, with increasing precision levels; colors being represented as cuboids (Jungel 2004). 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 (Schulz & Fox 2004) estimate colors using a hierarchical Bayesian model with *Gaussian* priors and a joint posterior on robot position and environmental illumination. Ulrich and Nourbakhsh (Ulrich & Nourbakhsh 2000) recognize obstacles by modeling the ground using color histograms and assuming non-ground regions to represent obstacles.

Our prior work (Sridharan & Stone 2005) 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 uncontrolled settings outside it.

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 (Sridharan & Stone 2005) we had presented an algorithm to learn colors autonomously (within 5 minutes) in the controlled setting of the lab, modeling colors as 3D Gaussians. Our hybrid representation for color distributions 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 (for example SIFT (Lowe 2004)) 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.

References

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

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. *PAMI* 24(5):603–619.

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

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

Johnson, D. H.; Gruner, C. M.; Baggerly, K.; and Seshagiri, C. 2001. Information-theoretic analysis of neural coding. *In Journal of Computational Neuroscience* 10:47–69.

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

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

Lauziere, Y. B.; Gingras, D.; and Ferrie, F. P. 1999. Autonomous physics-based color learning under daylight. In *The EUROPTO Conf. on Polarization and Color Tech. in Industrial Inspection*.

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

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

Sridharan, M., and Stone, P. 2005. Autonomous color learning on a mobile robot. In *The Twentieth National Conference on Artificial Intelligence (AAAI)*.

Sridharan, M.; Kuhlmann, G.; and Stone, P. 2005. Practical Vision-based Monte Carlo Localization on a Legged Robot. In *The International Conference on Robotics and Automation*.

Sumengen, B.; Manjunath, B. S.; and Kenney, C. 2003. Image segmentation using multi-region stab. and edge strength. In *ICIP*.

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

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

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*.