

PLANNING ACTIONS TO ENABLE COLOR LEARNING ON A MOBILE ROBOT

MOHAN SRIDHARAN AND PETER STONE

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.

Key Words. Color Learning, Robot Vision.

1. 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, 18], 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 on board a mobile robot 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, as addressed here, 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 input 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 [16], 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 learn colors 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 also able to autonomously plan a motion sequence suitable for learning the desired colors corresponding to any given configuration of objects, based on environmental knowledge and heuristic constraints on its motion sequence.

This paper therefore 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.

2. Problem Description

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

2.1. 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:

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

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

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

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 ($P(\omega_i) = 1/N$, $i \in [0, N - 1]$), each color's *aposteriori* probability is then given by:

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

The Gaussian model for color distributions functions well inside the lab. In addition, it generalizes well with limited samples when the color distributions are actually unimodal. It is hence 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 the colors effectively using 3D Gaussians.

Color histograms provide an excellent alternative when colors have multi-modal distributions in the color space [19]. 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 (such that values in the bins sum to 1) to provide the equivalent of the probability density function (Equation 2):

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

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 the color (ω). 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, histograms 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 involve a parameter estimation scheme that are computationally expensive to run in real-time on mobile robots. Hence, we use a hybrid representation with Gaussians and histograms. Furthermore, in tests conducted offline on sample data collected from the robot, we found that our method results in performance that is comparable to the other more sophisticated models.

2.2. 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° horz., 45.2° 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).



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 values¹ 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 (see Section 5) 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, while still operating at frame rate ($30Hz$). 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 [9]. This puts added emphasis on learning the color map in a short period of time.

3. Algorithm

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

Our previous algorithm [16] (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 color distributions are now multi-modal and can no longer be modeled properly as

¹We 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, θ) .

- 1: $i = 0, N = \text{MaxColors}$
- 2: $\text{Time}_{st} = \text{CurrTime}, \text{Time}[]$ — the maximum time allowed to learn each color.
- 3: **while** $i < N$ **do**
- 4: $\text{Color} = \text{BestColorToLearn}(i)$;
- 5: $\text{TargetPose} = \text{BestTargetPose}(\text{Color})$;
- 6: $\text{Motion} = \text{RequiredMotion}(\text{TargetPose})$
- 7: Perform *Motion* {Monitored using visual input and localization}
- 8: **if** $\text{TargetRegionFound}(\text{Color})$ **then**
- 9: Collect samples from the candidate region, $\text{Observed}[]$ [3].
- 10: **if** $\text{PossibleGaussianFit}(\text{Observed})$ **then**
- 11: $\text{LearnGaussParams}(\text{Colors}[i])$
- 12: *Learn Mean and Variance from samples*
- 13: **else** { 3D Gaussian not a good fit to samples }
- 14: $\text{LearnHistVals}(\text{Colors}[i])$
- 15: *Update the color’s 3D histogram using the samples*
- 16: **end if**
- 17: $\text{UpdateColorMap}()$
- 18: **if** $\text{!Valid}(\text{Color})$ **then**
- 19: $\text{RemoveFromMap}(\text{Color})$
- 20: **end if**
- 21: **else**
- 22: Rotate at target position.
- 23: **end if**
- 24: **if** $\text{CurrTime} - \text{Time}_{st} \geq \text{Time}[\text{Color}]$ or $\text{RotationAngle} \geq \text{Ang}_{th}$ **then**
- 25: $i = i + 1$
- 26: $\text{Time}_{st} = \text{CurrTime}$
- 27: **end if**
- 28: **end while**
- 29: Write out the color statistics and the Color Map.

Gaussians. In addition, it is time consuming to provide the appropriate motion sequence for each configuration of objects in the robot’s environment. The current algorithm significantly extends the previous approach in two ways. It automatically selects the best model by choosing between two different representations for each color (Gaussians, Histograms) to allow for color learning outside the lab. Furthermore, it *automatically* generates the motion sequence suitable for learning colors for any given starting pose and configuration of objects in its environment.

The robot starts off at a known field location without any color knowledge i.e. all images are segmented black. It has a list of colors (*Colors*) to be learned and a list of object descriptions (*Regions*). For each useful region that exists in the robot’s environment, the object description provides information on the size, shape, color

label and global location (x, y, z) . Both the robot's starting pose and the object descriptions 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, the robot has to exploit the available information to generate a suitable motion sequence while keeping itself well localized so that it can move to the desired locations. To generate the motion sequence, the robot essentially 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. We model these factors as a set of three weights, which the robot computes for each color-object combination (c, i) :

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

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, modeling the fact that we would like the robot to move smaller distances to learn the colors. The function $f_s(s(c, i))$ assigns larger weights to larger candidate objects, as larger objects provide the robot with a better opportunity to obtain more samples to learn the color parameters.

There are objects in the robot's environment that consist of more than one color and conversely there are colors that only exist in objects with more than one color. The function $f_u(o(c, i))$ incorporates this factor. If a particular object i consists of color c and other color(s), the color-object combination (c, i) is given a larger weight *iff* the other color(s) have already been learned. Also, if a particular object i for 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, (c, i) is given a larger weight.

Essentially the weights are used to dynamically determine the *value* of each color-object combination and the robot, during each decision cycle, chooses the combination that provides the highest value. The *BestColorToLearn* (line 4 in Algorithm 1) is then chosen according to the function described below:

$$(6) \quad \arg \max_{c \in [0, 9]} \left(\max_{i \in [0, N_c - 1]} (f_d(d(c, i)) + f_s(s(c, i)) + f_u(o(c, i))) \right)$$

where the robot parses through the different objects (N_c) available for each color ($c \in [0, N - 1]$) and calculates the weights. The color that provides the maximum value is chosen to be learned 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, i.e. the robot can learn the relative *value* of

each color-object combination over successive trials. This model could then be used to plan the globally optimal path instead of the current greedy approach.

Once a color is chosen, the robot determines the best target object to learn that color from, using the minimum motion and maximum size constraints, based on the following equation:

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

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 the desired color from this target object. The robot aims to position itself such that the entire target object is in its visual field. This can be easily achieved by considering the geometry of the objects (available from the object description) and the field-of-view of the robot's camera. It then determines the motion sequence, if any, necessary to reach the target pose (*RequiredMotion()* – line 6). This motion sequence is executed to move to the target pose. The current knowledge of colors is used to recognize objects, which are used as markers to localize using Particle Filtering [15]. This provides *visual feedback* for the desired motion to the target pose. In the initial stages, the robot does not have useful color information but as the robot learns colors, the visual feedback helps correct for inaccuracies in the motion model and/or slippage.

Once it gets close to the target location, the robot searches for candidate image regions satisfying a set of constraints based on the 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 goodness-of-fit with a 3D Gaussian (*PossibleGaussianFit()* – line 10). We use the statistical technique known as *bootstrapping* to check the goodness-of-fit of a 3D Gaussian for the given verification samples corresponding to a particular color. KL-divergence is used as a distance measure in this process as described in Algorithm 2. More details on the theory and motivation behind bootstrapping, along with several examples, can be found in [6].

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 in Algorithm 1). If not, the same candidate pixels are used to populate a 3D histogram using *LearnHistVals()* (line 14). The learned distributions, Gaussians and/or histograms, are used to generate the *Color Map*,

the mapping from pixel values to color labels, which is the output desired from the color learning process. This is done using the function *UpdateColorMap()* (line 17). Each cell in the color map is assigned a label corresponding to the color which has the largest *a posteriori probability* for that set of pixel values, as determined using Equation 3. This computationally intensive part of the learning process is performed only once every five seconds or so.

The updated color map is used to segment subsequent images and detect objects for two main reasons. First, it helps validate the learned parameters (*Valid()* – lines 18). If the target object corresponding to the color just learned is not found over several consecutive frames, it is taken as an indication of the fact that the color has been learned from an incorrect object. This can cause major problems in learning other colors. Hence, the corresponding color’s statistics are removed (*RemoveFromMap()* – line 19) from the color map. Second, it helps the robot *localize* to suitable locations [15] to learn the other desired colors. Remember that the robot initially starts out with absolutely no color knowledge and hence has to depend entirely on the motion model to localize. But, once a few colors are learned, known objects in the environment can be detected and used as markers to determine the pose of the robot in its environment. Essentially, our algorithm *bootstraps*, the knowledge available at any instant being exploited to plan and execute the subsequent tasks efficiently.

If, on the other hand, the candidate region is not found at the expected location, it is attributed to slippage and motion model errors. Given the limited field-of-view of the robot’s camera, a reasonably small error can lead to a candidate object not being visible at the final position. In such cases, the robot turns in place, searching for the candidate region, i.e. threshold angle $Ang_{th} = 360^\circ$. To ensure that the robot does not pick an incorrect candidate from the objects around the target object, only a certain amount of error in motion is accepted. We still let the robot turn one full circle, rather than have it turn a certain value in each direction, to avoid changes in the direction of motion. We are working on making the system more robust to motion errors. If the candidate region is found, the color learning process proceeds as described above. But, if the robot has turned through the threshold angle 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.

4. 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 (see Figure 2).

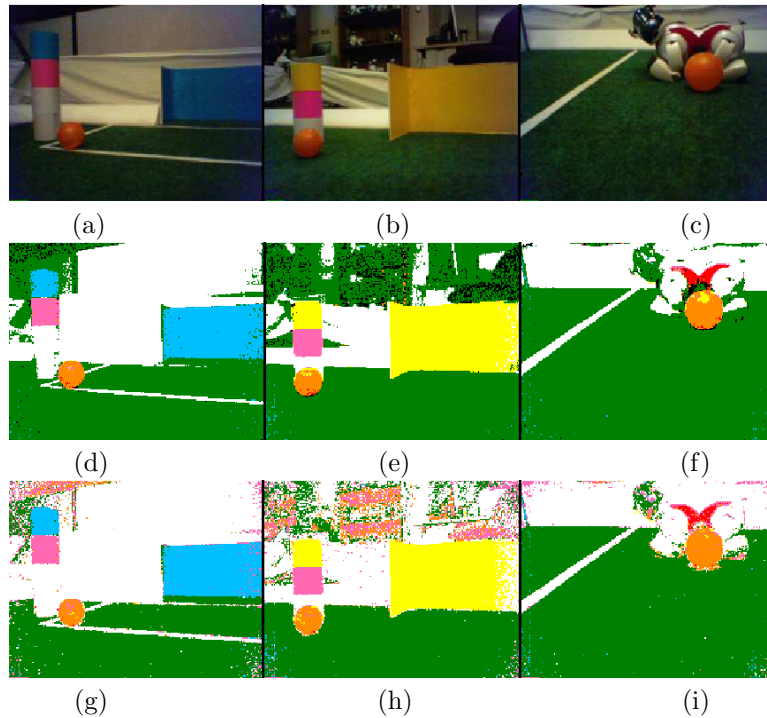


Figure 2: Images inside the lab. (a)-(c) Original, (d)-(f) *AllGauss*, (g)-(i) *AllHist*. Note that *AllHist* performs as well as *AllGauss*.

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).

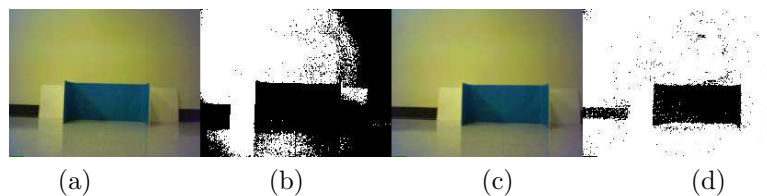


Figure 3: Segmentation using: (a)-(b) 3D Gaussians, (c)-(d) 3D Histograms. Gaussians do not model ground/walls well but Histograms do.

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*.

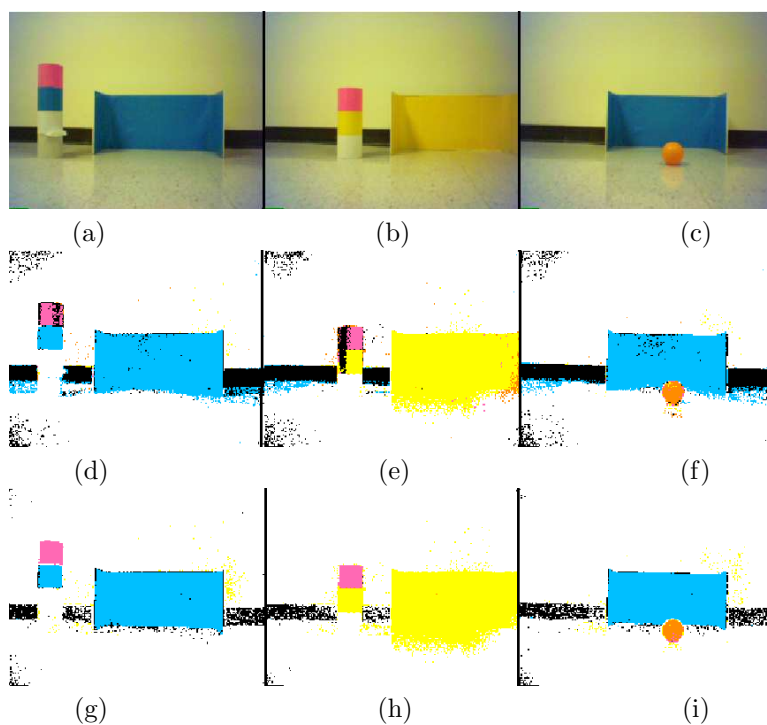


Figure 4: Images outside the lab (Case 1): (a)-(c) Original, (d)-(f) *AllHist*, (g)-(i) *GaussHist*. *GaussHist* performs better under minor illumination changes.

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. For example, when the color *red* is to be learned in addition to other overlapping colors such as *orange* and *pink*, the robot is unable to identify suitable candidate regions leading to false positives as seen in (e), (f) in Figure 5.

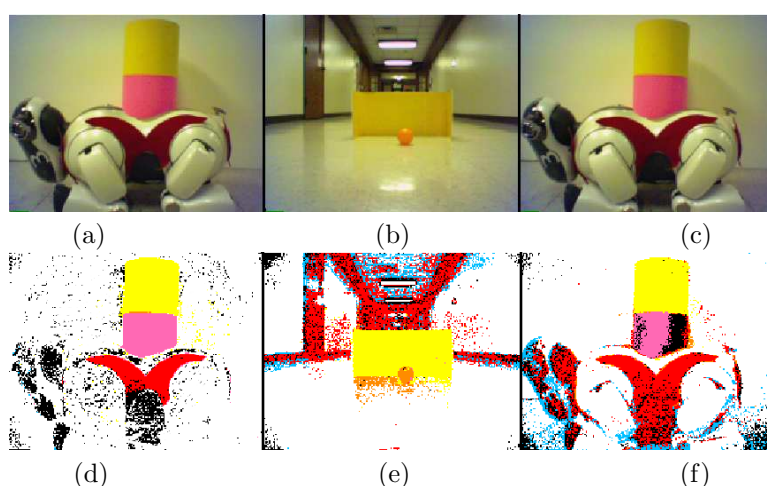


Figure 5: Images with opponent color in map: (a)-(c) Original, (d) *GaussHist*, (e)-(f) *AllHist*. *GaussHist* models overlapping colors better.

With Gaussians, the robot has the option of varying the spread of the known overlapping colors. Hence *GaussHist* enables the robot to learn all the colors using the good features of both models ((d) in Figure 5). Several other sample images can be viewed online: 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 walls were of a different color but again had varying shades as a result of the overhead illumination. Sample image results are shown in Figure 6.

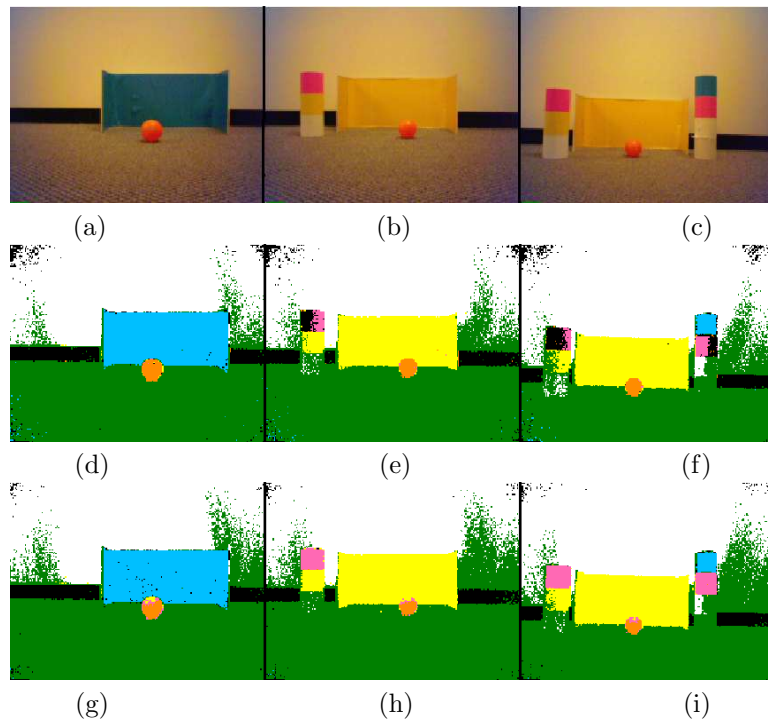


Figure 6: Images outside the lab (Case 2): (a)-(c) Original, (d)-(f) *AllHist*, (g)-(i) *GaussHist*. *GaussHist* generalizes better to handle minor illumination changes.

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 (see Row 2 in Figure 6). But *GaussHist* was able to learn all the desired colors (Row 3 of Figure 6) by choosing a suitable representation for all the colors in the robot’s environment.

Table 1 documents some numerical results corresponding to the two test cases in the indoor corridors. Here we compare the segmentation accuracy and the storage requirements of the two color representations (*AllHist* and *GaussHist*).

The storage requirements reflect the number of colors represented as histograms instead of Gaussians. We observe that *GaussHist* provides better accuracy than the *AllHist* and requires much less storage. The results reported in the table are statistically significant at the 95% significance level.

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

Table 1: Accuracies and storage requirements of models in two different indoor corridors. The results are statistically significant.

In addition to the images and videos corresponding to the experiments reported in this paper, our web site² also has images to show that the planned color learning scheme can be applied to different illumination conditions and can handle repaintings - 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). A group of seven graduate students with experience working with the Aibos were invited 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, where the robot has very little color knowledge, 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 configurations, with 10 trials for each configuration, are shown in Table 2.

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.

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. As seen in Table 2, 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 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 θ).

One configuration where the robot performs worst is shown in Figure 7. 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

²www.cs.utexas.edu/users/AustinVilla/?p=research/gen_color

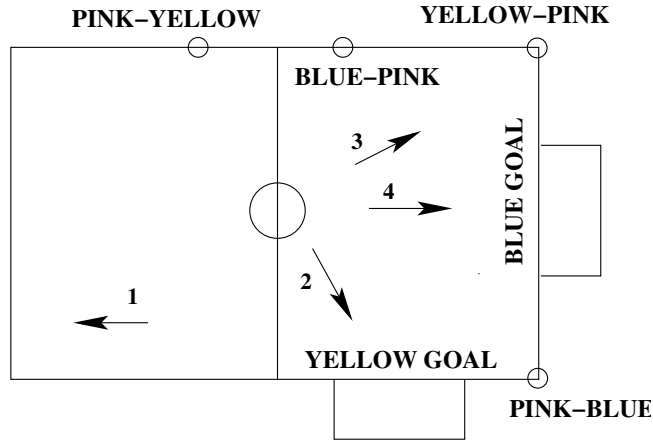


Figure 7: Sample Configuration where robot performs worst.

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.

5. Related Work

Color segmentation is a well-researched field in computer vision with several good algorithms, for example mean-shift [4] and gradient-descent based cost-function minimization [18]. The mean-shift algorithm is a very general non-parametric technique for the analysis of complex multi-modal feature spaces and the detection of arbitrarily shaped clusters. The feature space is modeled as an empirical probability density function (p.d.f) using a density estimation-based non-parametric clustering approach. Dense regions in the feature space correspond to local maxima, i.e. the modes of the unknown p.d.f. Once the modes are found, the associated clusters can be separated based on the local structure of the feature space. Mean-shift is a procedure that determines vectors aligned with the local gradient estimates, defining a path to the desired modes. Though it provides very good performance on several vision tasks such as segmentation and tracking, it is computationally expensive to perform on mobile robots with computational and memory constraints.

Another set of popular methods for image segmentation are the active contours (curve evolution) that define initial contours and then deform them towards the object boundaries. These can be classified into three groups: edge-based, region-based and hybrid. In [18], the authors describe a region-based method that segments images into multiple regions and integrates an edge-flow vector field-based edge function for segmenting precise boundaries. Their method allows the user to specify the similarity measure based on any image characteristic, such as color or texture. Also, the algorithm is not sensitive to the initial curve estimates. It provides good segmentation results on a wide variety of images but this again is computationally expensive to implement on mobile robots.

Even in the RoboCup domain, several algorithms have been implemented for color segmentation. The baseline approach involves creating mappings from the YCbCr values (ranging from 0 – 255 in each dimension) to the color labels [21]. Other methods include the use of decision trees [2] and the creation of axis-parallel

rectangles in the color space [3]. All these approaches involve the hand-labeling of several (≈ 30) images over a period of an hour or more before the classification scheme (decision-tree, color map) can be generated. Our approach on the other hand has the robot learn the colors autonomously in less than five minutes.

Attempts to learn colors or make them independent to illumination changes have produced reasonable success [10, 7] but the approaches either involve the knowledge of the spectral reflectances of the objects under consideration and/or require additional transformations that are computationally expensive to perform in the mobile robot domain.

Gevers and Smeulders [7] evaluate several color spaces to determine their suitability for recognizing multicolored objects invariant to significant changes in viewpoint, object geometry and illumination. They present a detailed theoretical and experimental analysis of the following models: RGB, Intensity I, normalized color rgb , saturation S, Hue H, and the newly proposed models $c_1c_2c_3$, $l_1l_2l_3$, $m_1m_2m_3$. They show that assuming dichromatic reflection and white illumination, normalized rgb , saturation S and Hue H, and the newly proposed models $c_1c_2c_3$, $l_1l_2l_3$ and $m_1m_2m_3$ are all invariant to the viewing direction, object geometry and illumination. Hue H and $l_1l_2l_3$ are also invariant to highlights, while $m_1m_2m_3$ is independent of the illumination color and inter-reflections under the narrow-band filter assumption. This provides a good reference on the choice of color space based on the application and the associated constraints.

Lauziere et al. [10] describe an approach for learning color models and recognizing objects under varying illumination using the prior knowledge of the spectral reflectances of the objects under consideration. In [11] they further explain the process of camera characterization. The color camera sensitivity curves are measured and used to recognize objects better under daylight illumination conditions.

Schulz and Fox [14], estimate colors using a hierarchical bayesian model with *Gaussian* priors and a joint posterior on position and environmental illumination. Ulrich and Nourbakhsh [20] recognize obstacles by modeling the ground using color histograms and assuming non-ground regions to represent obstacles. Our approach to color learning on the other hand uses an efficient *hybrid* representation for color classes that combines Gaussians and Histograms and works autonomously in *real-time* with no prior color knowledge, using the *structure* inherent in the environment.

Attempts to automatically learn the color map in the legged league have rarely been successful. One attempt based on scene geometry [1] involved detecting edges in the image and constructing closed figures to find image regions that correspond to certain known environmental features. The color information extracted from these regions is used to build the color classifiers, using the Earth Mover's distance (EMD) [13] as the cluster similarity metric. The dynamic changes introduced by illumination changes are tracked by associating the current classifiers with the previous ones, using the symbolic color labels. This approach to color learning is time consuming even with the use of offline processing. But the idea of using domain knowledge to learn colors is appealing and our online approach based on a similar idea is described in Section 3.

Jungel [8] presented another approach with reasonable success, where the color map is learned using three layers of color maps with increasing precision levels. Colors in each level are represented as cuboids, but colors that are *close* to each other are disambiguated using the world knowledge. In the object recognition phase, other domain specific constraints are introduced to disambiguate between object colors. Further, the colors are defined relative to a reference color (field *green* in

the robot soccer domain) and with minor illumination changes, the reference color is tracked and all the other colors are classified by displacing their regions in the color space by the same amount as the reference color. But different colors do not actually shift by the same amount with illumination changes. Also, the generated map is reported to be not as accurate as the hand-labeled one.

Our prior work [16] 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.

6. 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 [16] we had presented an algorithm that enabled the robot to learn colors in under five minutes, in the controlled setting of the lab, modeling colors as 3D Gaussians. Here, the robot used a motion sequence specified by a human observer. The hybrid representation for color distributions presented in this paper enables the robot to autonomously learn colors and localize in unengineered 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. This algorithm is still sensitive to significant changes in illumination and it needs human supervision to recognize such changes. In more recent work, we have demonstrated that the robot can detect changes in illumination automatically [17] and use that in conjunction to this planned color learning scheme to learn colors and adapt to illumination changes without any human intervention. We are currently working on making the robot learn the colors from any *unknown* location in its environment. This is challenging because the robot has to be able to deal with a lot of uncertainty, especially in the initial stages when it does not have any information on color or its pose. A future research direction is to have the robot *learn* the heuristic functions used in the current motion planning stage and do global motion planning instead of doing it in a greedy fashion.

The results presented in this paper indicate that the robot can 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 color-coded, 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.

Acknowledgments

Special thanks to Suresh Venkatasubramanian for his helpful discussions on the color learning algorithm. The authors would also like to thank Mazda Ahmadi for helping set up the environment for running some of the experiments. Thanks are

due to the members of the UT AustinVilla team for developing the code base used in the experiments. 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 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] T. Gevers and A. W. M. Smeulders. Color based object recognition. In *Pattern Recognition*, 32(3):453–464, 1999.
- [8] M. Jungel. Using layered color precision for a self-calibrating vision system. In *The Eighth International RoboCup Symposium*, Lisbon, Portugal, 2004.
- [9] H. Kitano, M. Asada, I. Noda, and H. Matsubara. Robot world cup. *Robotics and Automation*, 5(3):30–36, 1998.
- [10] Y. B. Lauziere, D. Gingras, and F. P. Ferrie. Autonomous physics-based color learning under daylight. In *The EUROPTO Conference on Polarization and Color Techniques in Industrial Inspection*, volume 3826, pages 86–100, June 1999.
- [11] Y. B. Lauziere, D. Gingras, and F. P. Ferrie. Color camera characterization with an application to detection under daylight. In *Vision Interface (VI)*, pages 280–287, May 1999.
- [12] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.
- [13] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover’s distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
- [14] D. Schulz and D. Fox. Bayesian color estimation for adaptive vision-based robot localization. In *The IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.
- [15] 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 (ICRA)*, April 2005.
- [16] M. Sridharan and P. Stone. Autonomous color learning on a mobile robot. In *The Twentieth National Conference on Artificial Intelligence (AAAI)*, 2005.
- [17] M. Sridharan and P. Stone. Color learning on a mobile robot: Towards full autonomy under changing illumination. In *The International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, January 2007.
- [18] B. Sumengen, B. S. Manjunath, and C. Kenney. Image segmentation using multi-region stability and edge strength. In *The IEEE International Conference on Image Processing (ICIP)*, September 2003.
- [19] M. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [20] I. Ulrich and I. Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *The National Conference on Artificial Intelligence (AAAI)*, July-August 2000.
- [21] 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.

The University of Texas at Austin, Austin, TX 78712, USA.

E-mail: smohan@ece.utexas.edu and pstone@cs.utexas.edu

URL: www.ece.utexas.edu/~smohan/ and www.cs.utexas.edu/~pstone/