To appear in *Autonomous Robots Journal*, 2007.

Structure-Based Color Learning on a Mobile Robot under Changing Illumination

Mohan Sridharan and Peter Stone

***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. To operate in the real world, autonomous robots rely on sensory information. Despite the potential richness of visual information from on-board cameras, many mobile robots continue to rely on non-visual sensors such as tactile sensors, sonar, and laser. This preference for relatively low-fidelity sensors can be attributed to, among other things, the characteristic requirement of real-time operation under limited computational resources. Illumination changes pose another big challenge. For true extended autonomy, an agent must be able to recognize for itself *when* to abandon its current model in favor of learning a new one; and *how* to learn in its current situation. We describe a self-contained vision system that works on-board a vision-based autonomous robot under varying illumination conditions. First, we present a baseline system capable of color segmentation and object recognition within the computational and memory constraints of the robot. This relies on *manually* labeled data and operates under *constant* and reasonably uniform illumination conditions. We then relax these limitations by introducing algorithms for i) Autonomous planned color learning, where the robot uses the knowledge of its environment (position, size and shape of objects) to automatically generate a suitable motion sequence and learn the desired colors, and ii) Illumination change detection and adaptation, where the robot recognizes for itself when the illumination conditions have changed sufficiently to warrant revising its knowledge of colors. Our algorithms are fully implemented and tested on the Sony ERS-7 Aibo robots.**

***Index Terms*— Color Learning, Illumination Invariance, Real-time Vision.**

I. Introduction

MOBILE robotic systems have recently been used in fields as diverse as medicine, surveillance, rescue, and autonomous navigation [1]–[3]. One key enabler to such applications has been the development of powerful sensors such as color cameras and lasers. Visual input, in the form of color images from a camera, can be a rich source of information, considering the sophisticated algorithms recently developed in the field of computer vision, for extracting information from images. Even so, most robots continue to rely on non-visual sensors such as tactile sensors, sonar, and laser [4].

This preference for relatively low-fidelity sensors rather than vision can be attributed to three major discrepancies between the needs of robots and the capabilities of state-of-the-art vision algorithms.

1) Most state-of-the-art approaches to challenging computer vision problems, such as segmentation [5], [6], blob clustering [7], object recognition [8]–[10] and illumination invariance [11]–[13] require a substantial amount of computational and/or memory resources. However, mobile robotic systems typically have strict constraints on the computational and memory resources available, but still demand real-time processing.
2) Most mobile robot platforms are characterized by a rapid nonlinear motion of the camera, especially legged robots. But most vision algorithms assume a stationary or slowly moving camera [10], [14].
3) Most current vision algorithms require extensive manual color calibration, making them inapplicable in domains with changing illumination conditions; robots, while moving around the world, often go into places with changing illumination. The very same pixel values corresponding to a color under one illumination may correspond to a different color in another illumination. Many current mobile robot applications ignore color because of this sensitivity to illumination, thereby overlooking potentially useful information.

This paper aims to address these challenges by exploiting the *structure* that is often present in a robot's environment. We define *structure* as the objects of unique shapes and colors that exist at known locations – a *color-coded world model*. We show that a robot can use this structure to model the color distributions, thereby achieving efficient color segmentation. Specifically, knowing that it is looking at an object of known color allows it to treat certain image pixels as *labeled training samples*. The domain knowledge also helps develop object recognition algorithms that can be used by the robot to localize and navigate in its complex world towards additional sources of color information.

We have developed a mobile robot vision system that *learns* colors using the uniquely color-coded objects at known locations, and *adapts* to illumination changes. Specifically, this article makes the following contributions:

- First, it describes a baseline vision system that tackles color segmentation and object recognition on-board a robot with constrained computational and memory resources. The baseline system is robust to jerky nonlinear camera motion and noisy images. However, it relies on *manually* labeled training data and operates in *constant* and uniform illumination conditions.
- Second, it exploits the structure inherent in the environment to eliminate the need for manual labeling. The image regions corresponding to known objects are used as *labeled training samples*. The learned color distributions are used to better identify the objects, thereby localizing and possibly moving to other sources of color information. We introduce a *hybrid* color representation that allows for color learning both within the controlled lab settings and in un-engineered indoor corridors.
- Third, it provides robustness to changing illumination

Both Authors are with the University of Texas at Austin.

conditions. We introduce an algorithm that enables the robot to *detect* significant changes in illumination. When a change in illumination is detected, the robot autonomously adapts by revising its current representation of color distributions. As a result, the robot is able to function over a wide range of illuminations.

The focus of this article is on the design of efficient robot vision algorithms that address challenging problems such as color segmentation, object recognition, color learning and illumination invariance. Using our algorithms the robot is able to operate autonomously in an uncontrolled environment with changing illumination over an extended period of time. The vision system is fully implemented and tested on a commercial off-the-shelf four-legged robot, the Sony ERS-7 Aibo [15]. We also illustrate the general applicability of our algorithms with the running example of a vision-based autonomous car on the road; we refer to it as the *car-on-the-road* task.

The remainder of the article is organized as follows. After a brief description of our test platform (Section II), we present our baseline vision system (Section III), which tackles the problems of color segmentation, object recognition and line detection, in real-time. Section IV extends the baseline system by eliminating the offline color calibration phase: the robot uses the environmental *structure* to autonomously generate a suitable motion sequence to learn the desired colors. Section V further enables the robot to detect significant illumination changes and adapt to them. We compare our approaches to related work in Section VII and present our conclusions and directions for future research in Section VIII.

II. Test Platform

The experiments described in this paper were performed on the Sony ERS-7 Aibo four-legged robot [15]. It is $\approx 280mm$ tall and $\approx 320mm$ long. It has 20 degrees of freedom: 3 in its head, 3 in each leg, and 5 more in its mouth, ears and tail. Its primary sensor is a CMOS color camera with a limited field-of-view (56.9^o horz. and 45.2^o vert.). Images are captured at 30Hz in the YCbCr image format, with a resolution of 208×160 pixels. In addition to 64MB on-board memory, the robot has noisy touch sensors, IR sensors, and a wireless LAN card for inter-robot communication.

The Aibo is popular in part due to its use as the standard platform in the RoboCup Legged League[1], where teams of four Aibos play a competitive game of soccer on an indoor field of size $\approx 4m \times 6m$ (Figure 1). The goal is to direct a ball into the opponents' goal while preventing the other team from scoring a goal. All processing for vision, localization, locomotion and action-selection, is performed on board using a 576MHz processor. Not operating at frame rate places the robot at a severe disadvantage in terms of reaction time. Games are currently played under constant and reasonably uniform illuminations, but the ultimate goal of the RoboCup initiative 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 [16]. The computational (and memory) constraints and the rapid nonlinear camera motion make the Aibo a challenging representative test platform. Other robot platforms may have more (or less) computational resources, and different camera parameters. But, in all mobile robot domains there are *some* hard constraints on these properties, within which the robot has to operate. Though we use the Aibo as a case study, our algorithms are described in general terms and are hence applicable to other mobile robot domains as well.

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

III. Baseline Vision System

We first present a real-time vision system that runs on a mobile robot platform with limited computational and memory resources, and rapid camera motions. Within these constraints that are characteristic of mobile robots we tackle the tasks of *color segmentation*, *object recognition*, and *line detection*.

Our baseline vision system takes as input a stream of limited-field-of-view images, the robot's initial position, and its joint angles over time, including the tilt, pan and roll of the camera. Additional sensory inputs, if available, can also be considered. On the Aibo, accelerometer values can be used to determine the body tilt and roll. The desired outputs are the distances and angles, with an associated probability measure, to a set of color-coded objects. In order to operate at frame rate (30Hz), each complete cycle of operation, including localization, locomotion, and decision-making, can take a maximum of 33msec. Throughout this section, we provide timing data for our algorithms. Though motivated by the robot soccer domain, this problem formulation is characteristic of other common mobile robot vision applications. In the *car-on-the-road* task for example, the camera mounted on a rapidly moving car has to deal with a noisy, distorted stream of images and detect color-coded objects such as stop and yield signs.

Our vision algorithm proceeds in two stages: i) color map generation and region/blob formation (Section III-A) ii) marker and line recognition (Section III-B). Figure 2 shows four representative images from the robot soccer environment, which we use to illustrate the results of each stage of our vision system (Figures 3–6). Sample videos showing the robot's view, after each stage of processing, are available online.[2]

[1]www.tzi.de/4legged

[2]www.cs.utexas.edu/users/AustinVilla/?p=research/robust_vision

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

Fig. 2: Sample Images in the RGB color space.

A. Color Segmentation and Region Merging

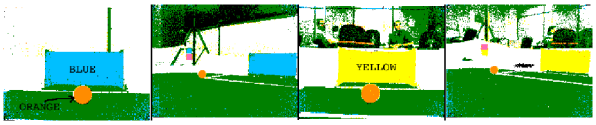
The first step in our baseline vision system is color segmentation, mapping image pixels to color labels. [3] A complete mapping identifies a label for each point in *YCbCr* space:

$$\forall p, q, r \in [0, 255], \{Y_p, Cb_q, Cr_r\} \mapsto l\,|_{l \in [0,8]} \quad (1)$$

Though prior research has produced several good segmentation algorithms [5], [17], [18], they are computationally expensive to perform on robots such as the Aibo, given its computational constraints. A variety of approaches have been implemented in the RoboCup domain, including decision trees [19] and axis-parallel rectangles in color space [20].

Our approach is motivated by the desire to create mappings from each YCbCr pixel value to a color label [21]. We represent this mapping as a *color map*, or *color cube*, created via an off-board training phase. A set of images ($\approx$ 25) captured by the robot's camera are hand-labeled such that the robot learns the range of pixel values that map to each color. The hand-labeled data labels only $\approx$ 3% of the color space, and to generalize from this labeling, each cell in the color map is assigned a color label that is the weighted average of the cells a certain *Manhattan distance* away (a form of *Nearest Neighbor-NNr*). As a result, *holes* and edge effects are removed, and a good representation is created for colors with overlap. To reduce memory requirements we subsample the color space to have values ranging from 0–127 in each dimension. The resulting color map, $\approx$ 2MB in size, is loaded on the robot and is used to segment subsequent images.

The segmentation in the *YCbCr* color space is sensitive to minor illumination changes, such as with shadows or highlights. Previous research in rescue robotics has suggested that a spherically distributed color space, *LAB*, inherently provides some robustness to illumination changes [2]. To take advantage of *LAB*'s properties without incurring the overhead of on-line conversion, the initial labeling and the NNr operation are done in *LAB*. Then, each cell in the *YCbCr* color map is labeled based on the label of the corresponding cell in the *LAB* color map. The on-line pixel-level segmentation process remains a table lookup process taking $\approx$ 0.1msec per image.



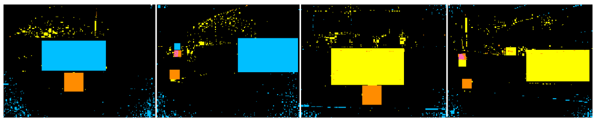
(a) (b) (c) (d)

Fig. 3: Sample Segmented Images — Compare with Figure 2.

We compared the segmentation accuracy of the two color spaces over $\approx$ 30 images captured to reflect small changes in illumination. The classification accuracies (%) were 81.2 ± 4.4 and 92.7 ± 2.5 for *YCbCr* and *LAB* respectively (statistically significant at 95% confidence level). Figure 3 shows the segmentation performance on the images in Figure 2.

The next step is to find contiguous *regions* of constant colors, i.e. *cluster* pixels of the same color into meaningful groups. Our approach is modeled after previous approaches [21]. As the image pixels are segmented they are organized into run-lengths [22] represented as the start point and length in pixels of a contiguous color strip. As an optimization, we only encode colors that identify objects of interest – we omit the colors of the field (*green*) and the lines (*white*). Lines are detected by an efficient line-detection algorithm described in Section III-B.

Next, we use an implementation of the Union-Find algorithm [23] to merge run-lengths of the same color that are within a threshold *Euclidean* distance from each other. We also progressively build *bounding boxes* i.e. rectangular boundaries around the regions. This abstraction provides a set of bounding boxes, one for each region in the current image, and a set of properties corresponding to each region, such as the number of pixels it envelopes. Our technical report [24] has complete details on the thresholds and properties used in this process. Figure 4 shows the result of region formation.



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

Fig. 4: Sample Regions — Compare with Figure 2.

Color segmentation and region formation, which constitute the low-level vision module, take $\approx$ 18msec per image. Though presented in the context of the Aibo, the algorithms presented here generalize to other mobile robot applications. In the case of the *car-on-the-road* task, we would still need to recognize colored regions in varying backgrounds, e.g. red for the stop sign, yellow for the yield sign, and white for the lines on the road.

B. Object Recognition and Line Detection

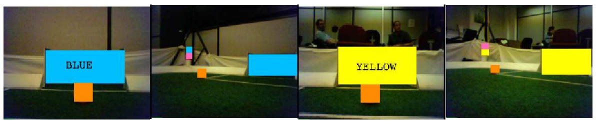
Once we have candidate regions, the next step is to recognize the relevant objects in the image. Segmentation errors due to noise and/or irrelevant objects (people, chairs, walls, computers) can lead to the formation of spurious regions (Figure 4) and make object recognition challenging. Though several successful approaches have been proposed for object recognition [9], [10], they typically involve extensive computation of object features or large amounts of storage in the form of object templates corresponding to different views.

Most robot application environments are structured and this domain knowledge can be exploited to recognize useful objects. The domain knowledge that gets incorporated as geometric and heuristic constraints depends on the application. Here, all the objects in the robot's environment (fixed markers used for localization and the moving objects that are tracked) are color-coded. In the *car-on-the-road* task, the objects could include the stop and yield signs, other vehicles and grass by the side of the road.

[3]pink, yellow, blue, orange, red, darkblue, white, green, black.

A set of geometric and heuristic constraints are designed to eliminate spurious regions that do not meet constraints on size, density and image position. For example, all objects of interest to the robots are on or a certain distance above the ground, and have bounding boxes with high densities. Full details of the heuristics are available in our technical report [24]. These heuristics are easy to apply since the required properties were stored in the region formation stage. The degree of conformity between expected and observed values of the properties is used to determine the probability of occurrence of each object. For example, if the known aspect ratio (height/width) of an object is 2.0 and observed aspect ratio in the image is 1.5, the probability of occurrence of the object is 0.75.

We tested the object recognition performance over eight sequences of ≈ 200 images each, ground truth provided by a human observer. We performed this test both without and with robot motion (objects stationary). The corresponding classification accuracies were 100% and 92.7% respectively (no false positives). The motion-based image distortion causes a decrease in accuracy. Figure 5 shows sample results.



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

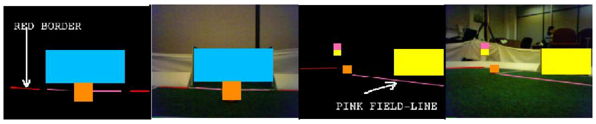
Fig. 5: Sample Object Recognition — Compare with Figure 2.

Once an object is recognized in the image, the relative distance and angle to the object are determined using trigonometric transforms and known sizes [24]. The vision module, up to the object recognition phase, takes ≈ 28msec per frame. A video of the robot's view, as it moves and recognizes objects, can be seen online.[4]

In addition to the objects, lines with known locations are important sources of information, especially since the robots' main focus (during a game) is the ball, and other robots may occlude the markers. In the *car-on-the-road* task, lines help recognize lanes and pedestrian crossing zones. Previous research has resulted in methods such as Hough Transforms, and edge detectors such as Canny and Sobel [22]. But given that the robot also has to localize, move, and cooperate with team-members using its limited computational resources, these algorithms are computationally expensive to use.

Our approach builds on a previous approach in the RoboCup domain [25]. It utilizes environmental knowledge: edges of interest on the field involve a white-green or green-white-green transition corresponding to the borders and the field lines respectively. A series of vertical scans are performed on the segmented image, the scan lines spaced $4-5$ pixels apart to speed up the scanning, and to eliminate noisy lines that extend only a few pixels across. The observation of lines closer to the robot provides more reliable (less noisy) information. The robot therefore scans the image from the bottom of the top, and once an edge pixel is detected along a scan line, the algorithm proceeds to the next scan line even though it prevents it from finding line pixels further along the scan line. The scan lines are suitably oriented to compensate for the camera motion-based image rotation. The candidate edge pixels are filtered through a set of heuristic filters whose parameters were determined experimentally [24].

Instead of using the detected edge pixels as localization inputs, as in previous approaches [25], lines are fit to the edge pixels using the Least Square Estimation procedure [26]. Although line pixels (or lines) provide useful information, the line intersections, though still not unique, involve much less ambiguity, which can be resolved using prior robot pose. In order to determine the line intersections, a pair of lines are considered at a time. Line intersections are accepted only if they satisfy experimentally determined heuristic thresholds [24]. Figure 6 shows a set of images with field lines in pink and border lines in red.



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

Fig. 6: Sample Line Recognition — Pink Field Lines and Red Borders.

We also analyzed the performance over ≈ 2000 images both for a stationary and moving robot. The corresponding classification accuracies were 100% and 93.3% respectively, with no false positives in either case. We noticed a significant improvement in our localization accuracy $(10-15\%)$ once we used lines/line intersections as inputs to our particle filtering localization algorithm [27]. The entire baseline system operates at ≈ 27msec per frame so that the robot is able to operate at frame-rate (≈ 33msec per frame).

C. Summary - Baseline System

We have described a baseline vision system that works in real-time on the robot, performing color segmentation, object recognition and line detection. This work is also described in [28]. Though fully implemented and effective in the robot soccer domain, the system suffers from two major drawbacks that prevent autonomous operation. First, even for a fixed illumination, the vision system requires time-consuming manual color calibration. Second, the system is highly dependent on constant and uniform illumination for its operation, something that is not representative of a typical mobile robot environment. In the next two sections we present our solutions to these problems: *autonomous color learning* and *illumination invariance*.

IV. Planned Color Learning

The baseline system described above (Section III-A) involved hand-labeling several ($\approx 20-30$) images, leading to *more than an hour* of manual effort before the robots can be deployed. The calibration has to be repeated each time the illumination changes significantly. Here, we present a novel approach that enables the robot to *autonomously learn* the desired colors, suitably planning its motion sequence based on known positions of color-coded objects. Using a *hybrid color*

[4] www.cs.utexas.edu/users/AustinVilla/?p=research/robust_vision

representation the entire color map is learned autonomously in *less than five minutes*, both within controlled lab settings and in the less controlled settings outside it. The segmentation and localization accuracies are comparable to those from a hand-labeled color map.

A key defining feature of the algorithm is that there is *no a priori color knowledge* or labeled color data. The method depends *only* on the known positions, shapes and colors of objects. It is independent of the actual color labels (blue, yellow etc.) assigned to each object, and is hence robust to different illuminations and even changes of entire colors (e.g. repainting all red objects as blue and vice versa). Even in the *car-on-the-road* task, we could use objects of known colors: red-stop sign, yellow-yield sign, white/yellow lines on the road, to learn the desired colors. Note that we are not entirely removing the human input: we still provide the positions of useful objects. But, in many applications, particularly when object locations change less frequently than illumination, it is more efficient than hand-labeling several images.

As in the case of the baseline vision system (Section III), the problem of color segmentation can be characterized by a set of inputs, outputs and constraints.

1. *Inputs:*

- A color-coded model of the world that the robot inhabits. The model contains a representation of the size, shape, position, and colors of all objects of interest. *We did not use this in the baseline system.*
- A stream of limited-field-of-view images. The images present a view of the world with many useful color-coded objects, but also many unpredictable elements.
- The initial position of the robot and its joint angles over time, particularly those specifying the camera motion.

2. *Output:*

- A *Color Map* that assigns a *color label* to each point in the input color space.

3. *Constraints:*

- Limited computational and memory resources with all processing being performed on-board the robot.
- Rapid motion of the limited-field-of-view camera with the associated noise and image distortions.

We aim to generate a reliable mapping from the inputs to the outputs, while operating within the constraints imposed by the test platform. In our approach, the robot uses the color-coded world model to plan a motion sequence that puts it in positions appropriate to learn the desired colors. At each position, the robot selects suitable image regions and models colors using a *hybrid* color representation. The learned colors are used to recognize objects, localize and hence move to positions suitable for learning other colors. We begin by formally describing the color segmentation problem (Section IV-A), a generalization of the description in Section III-A. Section IV-B provides details of the actual algorithm. A description of the experimental setup and the experimental results (Section IV-C) is followed by a summary (Section IV-D).

A. Problem Specification

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

$$\forall i, j, k \in [0, 255], \quad \Pi_E : \{c_{1,i}, c_{2,j}, c_{3,k}\} \mapsto l|_{l \in [0,N-1]} \tag{2}$$

where c_1, c_2, c_3 are the color channel (e.g. RGB) values, and E depicts the dependence on the current illumination.

In our preliminary color learning approach [29], each color was represented using a three-dimensional (3D) Gaussian with mutually independent color channels, i.e. we assume that there is very little correlation between the values along the three color channels for any color. In practice this assumption does not hold perfectly, depending on the color space under consideration. For example, in the lab, the correlation coefficients between Cb and Cr color channels in the $YCbCr$ color space, $\rho_{cbcr} = -0.71, -0.67$ for yellow and orange respectively. But for most colors in the color space we use for segmentation (LAB), the correlation is small enough to justify the independence assumption, which simplifies the computation considerably. Using empirical data and the statistical technique of bootstrap [30], we determined that the Gaussian representation closely approximates reality. In addition, the Gaussian model only requires the computation and storage of the mean and covariance matrix for each color, thereby reducing the memory requirements.

For the 3D Gaussian model, the *apriori* probability density functions (pdfs) for a color ($l \in [0, N-1]$) is given by:

$$p(\mathbf{c}|l) \sim \frac{1}{(2\pi)^{3/2}|\mathbf{\Sigma}_l|^{1/2}} \cdot e^{\{-\frac{1}{2}(\mathbf{c}-\boldsymbol{\mu}_l)^t \mathbf{\Sigma}_l^{-1}(\mathbf{c}-\boldsymbol{\mu}_l)\}} \tag{3}$$

where the random variable $\mathbf{c} = c_1, c_2, c_3$ represents the distribution of color's values along the three color channels, while $N(\boldsymbol{\mu}_l, \mathbf{\Sigma}_l)$ defines the 3D Gaussian for color l. Assuming equal priors ($P(l) = 1/N, \ \forall l \in [0, N-1]$), each color's *aposteriori* probability is given by:

$$p(l|\mathbf{c}) \propto p(\mathbf{c}|l) \tag{4}$$

The Gaussian model for color distributions performs well inside the lab, generalizing with limited samples and handling minor illumination changes when the color distributions are actually unimodal. However, in un-engineered settings outside the lab, factors such as shadows and larger illumination changes result in multi-modal color distributions which cannot be modeled properly using Gaussians.

Color histograms provide an excellent alternative to Gaussians when colors have multi-modal distributions in the color space [31]. 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 to provide the pdf for that color (Equation 3):

$$p(\mathbf{c}|l) \equiv \frac{Hist_l(b_1, b_2, b_3)}{\sum Hist_l} \tag{5}$$

where b_1, b_2, b_3 represent the histogram bin indices corresponding to the color channel values c_1, c_2, c_3. The *aposteriori* probabilities for each color are then given by Equation 4.

Unfortunately, histograms do not generalize well with limited training data, for instance for new samples produced by minor illumination changes. Constrained 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. Here, 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 decision is made online by the robot, for each color, based on pixel samples.

Other distribution models were also found to be feasible (exponential, mixture-of-Gaussians etc), but the Gaussian and histogram constitute a minimal set of models that provide the required capability to model the desired distributions, and they perform as well as the other more sophisticated models. In addition, the parameters of these two models can be easily determined in real-time within the computational constraints of mobile robot platforms.

B. Algorithm

Algorithm 1 describes our approach. Underlined function names are described below.

Our preliminary algorithm [29] (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. As mentioned above, the Gaussian assumption may not hold outside the constrained lab setting. The current algorithm uses a *hybrid representation* that automatically chooses between two different models for each color and *automatically* generates a motion sequence suitable for learning colors for any given robot starting pose and object configuration.

The robot starts off at a known pose with the locations of various color-coded objects known. It has no prior color information (images segmented *black*). It has a list of colors to be learned ($Colors$) and an array of structures ($Regions$), where each structure corresponds to an object of a particular color and stores a set of properties, such as its size (length, 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 modify the list of candidate regions for each color. Though this approach does require human input, in many applications, particularly when object locations change less frequently than illumination, it is more efficient than hand-labeling images.

Due to the noise in the motion model and the initial lack of visual information, constraints need to be imposed on the robot's motion and the position of objects, in order to resolve conflicts that may arise during the learning process. These heuristic constraints depend on the problem domain. Here, two decisions need to be made: 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 analyzes one step at a time without actually planning for the subsequent steps. The details of the algorithm and the corresponding heuristics are presented primarily for the replicability of our work. Our aim is to demonstrate that such autonomous color

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: $i = 0, N = MaxColors$
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.

learning can be accomplished in a setting where it is typically done manually.

In our task domain, the following three factors influence these choices:

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 learned 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.

Specifically, if a color can be learned with minimal motion and/or is visible in large quantities around the robot's current location, it should be learned first. Sometimes a color can be learned more reliably by associating it with another color around it. For example, in our default configuration, *pink* has regions of the same size associated with either *blue* or *yellow*. The robot attempts to learn one of those two colors before it attempts to learn *pink*. Essentially, 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. The relative importance weights assigned to the individual factors are used to resolve the conflicts, if any, between the factors.

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

$$\begin{aligned} w_1 &= f_d(\ d(l,i)\ ) \\ w_2 &= f_s(\ s(l,i)\ ) \\ w_3 &= f_u(\ o(l,i)\ ) \end{aligned} \tag{6}$$

where the functions $d(l,i)$, $s(l,i)$ and $o(l,i)$ represent the distance, size and object description for each color-object combination. The function $f_d(\ d(l,i)\ )$ assigns a smaller weight to distances that are large, while $f_s(\ s(l,i)\ )$ assigns larger weights to larger candidate objects. The function $f_u(\ o(l,i)\ )$ assigns larger weights if the particular object (i) for a particular color (l) is 'unique', which here implies that it is not composed of any color, in addition to (l), that is currently unknown.

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

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

where the robot parses through the different objects available for each color (N_l) and calculates the weights. For each color, the object that provides the maximum weight is determined. Next, the color that results in the maximum among these values 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.

Once a color is chosen, the robot determines the target object to learn the color from (*best-candidate-object*):

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

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

Next, the robot calculates the *BestTargetPose()* (line 5) to detect this target object. Specifically, using the known world model, it attempts to move to a pose where the entire candidate object would be in its field of view. Using its navigation function – *RequiredMotion()* (line 6) – the robot determines and executes the motion sequence to place it at the target pose. The current knowledge of colors is used to recognize objects and localize using particle filtering [27] thereby providing *visual feedback* for the motion.

Once it gets close to the target pose, the robot searches for image regions that satisfy the heuristic constraints for the target object. The structure *Regions[Color][best-candidate-object]* provides the actual properties of the target object such as its (x,y,z) location, width and height. Based on its pose and geometric principles, the robot uses these properties to dynamically compute suitable constraints. The robot stops when either *TargetRegionFound()* (line 8) is *true* or its pose estimate corresponds to the target position.

If a suitable region is found, the robot stops with the region at the center of its visual field. The pixel values in the region, which satisfy simple out-lier checks, are used as *verification samples*, $Observed$, to check goodness-of-fit with a 3D Gaussian (*PossibleGaussianFit()* — line 10). The statistical *bootstrap* technique is used, with KL-divergence [32] as the distance measure (Algorithm 2). Appendix II describes the bootstrap test and shows that the 3D Gaussian is a good fit for the color distributions within controlled lab settings.

Algorithm 2 PossibleGaussianFit(), line 10 of 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 $Estimated$ – $Data$, 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 robot executes *LearnGaussParams()* (line 11). Each pixel of the candidate region (currently *black*, i.e. unlabeled) that is sufficiently distant from the *means* of the other known color distributions is selected. The *mean* and *covariance* of these pixel values represent the pdf 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).

Next, the function *UpdateColorMap()* (line 17) uses the learned distributions to generate the *Color Map*. Assigning color labels to each cell in the $128 \times 128 \times 128$ map is computationally expensive and is performed only once every five seconds. Histograms are normalized (Equation 5) to generate pdfs. Each cell in the color map, which corresponds to a particular 3D vector of pixel values, is assigned a label corresponding to the color which has the largest *aposteriori probability* (Equation 4) for that vector 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. Therefore, 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 covariance. Histograms do not have this problem.

The updated map is used to segment subsequent images and detect objects. This helps validate the learned parameters and remove erroneous color statistics (Gaussian/Histogram) if necessary (line 18, 19). Furthermore, it 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 candidate object is not found at the target location, it is attributed to slippage and the robot turns in place, searching for the candidate region with slightly relaxed constraints. The robot turns a complete circle rather than turning a certain amount in each direction to avoid abrupt transitions. The constraints on size and location prevent the selection of a wrong target image region under most cases, and the validation process handles the other cases. If the robot has turned 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. The process continues until the robot has tried to learn all the colors. Then the color map and statistics are saved. A video of the color learning process can be seen online.[5]

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 object descriptions 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.

C. Experimental Results

Our algorithm is successful if the robot is able to plan a suitable motion sequence and learn all the desired colors in its environment. Hence, we test 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 lab setting. Qualitatively, both representations produced similar results (Figure 7). 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 $0.15Kb$ respectively i.e. *AllHist* performs as well as *AllGauss* but requires more storage.

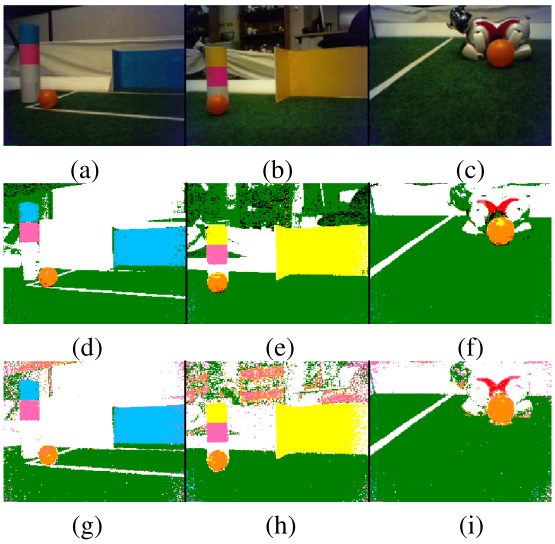


Fig. 7: Images inside the lab. (a)-(c) Original, (d)-(f) $AllGauss$, (g)-(i) $AllHist$. $AllHist$ performs as well as $AllGauss$.

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 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. The robot was provided with a world model with color-coded objects of interest, but because 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 and walls, causing problems with finding candidates for the other colors – Figure 8).

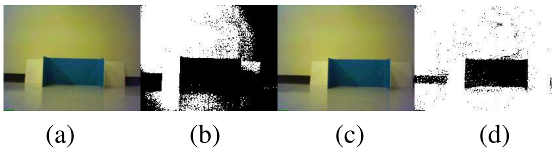


Fig. 8: Segmentation with: (a)-(b) 3D Gaussians, (c)-(d) 3D Histograms. Histograms model ground or wall colors better.

With the hybrid color representation, *GaussHist*, the robot, based on the statistical tests, ended up modeling one color (walls and ground) as a histogram and the others as Gaussians. Figure 9 compares *AllHist* with *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 tackle minor illumination changes that are inevitable during testing (errors in row 2 of Figure 9). The 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

[5] www.cs.utexas.edu/users/AustinVilla/?p=research/auto_vis

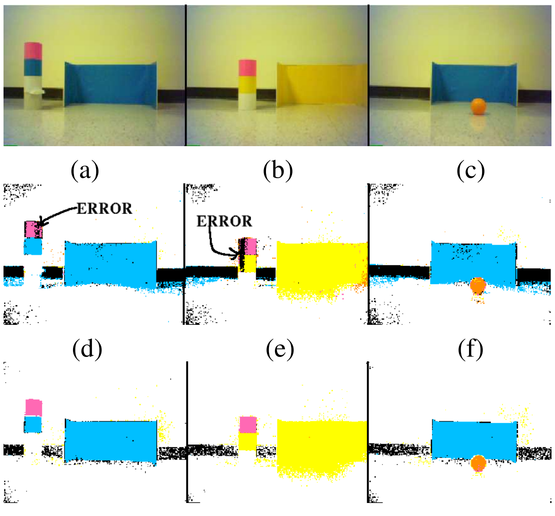


Fig. 9: Images outside the lab: (a)-(c) Original, (d)-(f) *AllHist*, (g)-(i) *GaussHist*. *GaussHist* performs better under minor illumination changes.

identify a suitable candidate region. As seen in Figure 10 (e, f), it leads to false positives and the segmentation performance over other colors deteriorates.

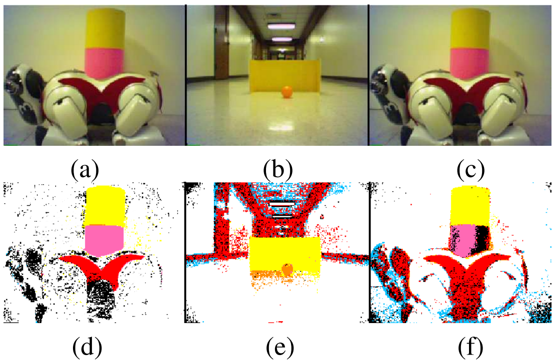


Fig. 10: Images with opponent color in map: (a)-(c) Original, (d) *GaussHist*, (e)-(f) *AllHist*. *GaussHist* models overlapping colors better.

With Gaussians, the robot has the option of varying the spread of the known overlapping colors, such as *orange* and *pink*. Hence *GaussHist* successfully learns 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 *AllHist* had problems when faced with the inevitable minor illumination variations during testing. But *GaussHist* enabled the robot to successfully learn the desired colors. We also ran the color learning experiments with other objects instead of those on the robot soccer field (trash cans, boxes etc.). These objects were not uniform-colored, resulting in multi-modal color distributions. But the robot successfully learned those colors as well, as a result of the hybrid color representation.

Table I documents numerical results for the two test cases outside the controlled lab setting. The storage requirements reflect the number of colors represented as histograms instead of Gaussians. Sample images for this setting can be seen online.[6] We also provide images to show that the planned color learning scheme can be applied to different illuminations, and can handle re-paintings — changing all *yellow* objects to *white* and vice versa poses no problem.

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

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

One challenge in experimental methodology was to measure the robot's planning capabilities in qualitatively *difficult* setups (objects configurations and robot's initial position). We described our algorithm to seven graduate students with experience working with the robots and asked them to pick a few test configurations each, 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 five colors needed for localization (*pink, yellow, blue, white, green*) are learned. Table II tabulates the performance of the robot in its planning task over 15 configurations, with 10 trials for each configuration.

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

TABLE II: Successful Planning and Localization Accuracy.

Table II also shows the localization accuracy of the robot using the learned color map. The robot is able to plan its motion to learn colors and execute it successfully in most of the configurations that were designed to be adversarial. The corresponding localization accuracy is comparable to that obtained with the hand-labeled color map ($\approx 6cm, 8cm, 4deg$ in X, Y, and θ).

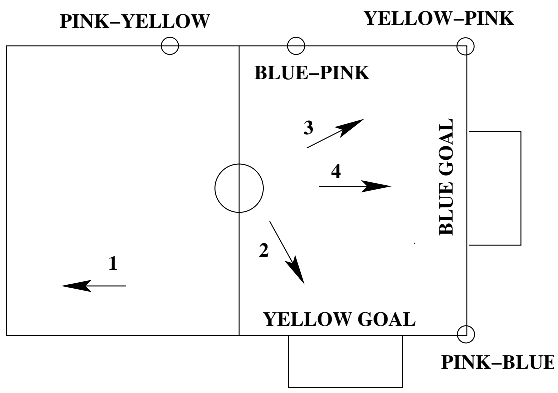


Fig. 11: Sample Configuration where robot performs worst.

[6]www.cs.utexas.edu/∼AustinVilla/?p=research/auto_vis

One configuration where the robot performs worst is shown in Figure 11. Here, it is forced to move a large distance to obtain its first color-learning opportunity (from position 1 to 2). The large motion without visual feedback sometimes leads the robot into positions quite far away from its target location and it is unable to find any candidate image region that satisfies the target object's constraints. Currently, failure in the initial stages 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 learned color map to localize to points in an indoor corridor can be seen online. [7]

D. Summary - Color Learning

One major drawback of our baseline vision system (Section III) was the need for elaborate manual sensor calibration before deployment. Here, we have described an algorithm that enables the robot to use the known world model (*structure*) to *autonomously* plan a suitable motion sequence and learn colors. Using our *hybrid representation* allows for color learning both within the controlled lab environment and in less controlled settings outside it, such as indoor corridors. The algorithm *bootstraps* – the color map available at any stage is used to detect objects, thereby localizing better to locations suitable for learning other colors. The segmentation and localization accuracies with the learned color map are comparable to that with the hand-labeled color map. In addition, the robot is able to plan its motion sequence for several different object configurations that were specifically designed to be adversarial. This approach is also described in [29], [33].

In our robot soccer domain the objects of interest are known markers that are color-coded. In the *car-on-the-road* problem, the vision system would be able to learn a yield sign colored *yellow* and a stop sign painted *red*. Hence we use colors as the distinctive features. But in environments with features that are not constant-colored, other feature representations such as SIFT [8], could be used. As long as the *locations* of the objects remain as indicated on the map, the robot could robustly relearn how to detect them.

We have made it possible to quickly train a new color map whenever illumination changes significantly. But it does not provide a mechanism for automatically detecting and adapting to illumination changes. In the next section, we tackle this limitation of the baseline vision system, its sensitivity to illumination changes.

V. Adapting to Illumination Changes

A robot operating in the real world is subjected to illumination changes, such as between day and night. When illumination changes, it causes a nonlinear shift in the color distributions in the color space [12], and the previously trained color map ceases to be useful. On robots with color cameras, this typically requires the repetition of the training phase that generates the color map. In real world tasks such as the *car-on-the-road* example, lack of proper color information can lead to rather disastrous consequences.

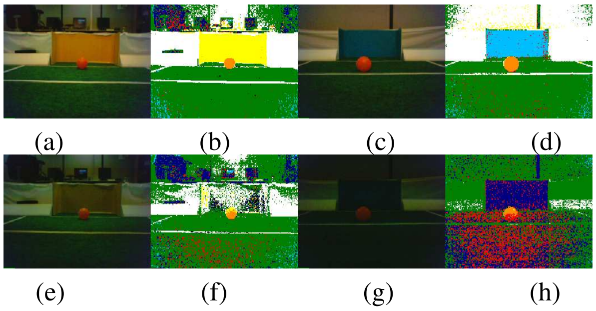


Fig. 12: Sample Images showing Illumination Sensitivity.

As shown in Figure 12 the color map trained for an illumination condition works fine for minor changes in illumination but results in very bad segmentation when the illumination changes significantly. The top row shows segmentation results when a color map is trained and tested on images captured under the same illumination condition. The bottom row shows the segmentation obtained when the same map is used to segment images captured under a different illumination — note that almost entire objects are segmented incorrectly.

Our autonomous color learning algorithm, described in Section IV-B, enables the robot to learn the color map but continuous human supervision is still required to enable the detection of illumination changes. Stated differently, we have enabled the robot to decide *What to learn* (choosing between Gaussian and Histogram for color distributions) and *How to learn* (planning motion sequence). But the robot still cannot decide *When to learn*. In order to work over a range of illuminations, the robot must be able to:

1. Detect a change in illumination conditions by extracting suitable statistics from its input images;
2. Automatically learn a new color map if it is put in an illumination condition which it has never seen before;
3. Transition to an appropriate color map if it is placed in an illumination condition that it has learned a color map for, and use that for subsequent vision processing;
4. Perform all the necessary computation efficiently without having an adverse effect on its task performance.

We formally describe the problem and our solution in Section V-A, followed by the algorithm (Section V-B), the experimental results (Section V-C) and a summary (Section V-D).

A. Problem Specification

In order to detect significant changes in illumination, a mechanism for representing different illuminations and differentiating between them is needed.

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

[7] www.cs.utexas.edu/users/AustinVilla/?p=research/gen_color

$$r = \frac{R+1}{R+G+B+3}, \quad g = \frac{G+1}{R+G+B+3}, \quad b = \frac{B+1}{R+G+B+3} \tag{9}$$

since $r + g + b = 1$, any two of the three features are a sufficient statistic. An illumination is represented by a set of (r, g) histograms (pdfs), quantized into N bins in each dimension, corresponding to images captured by the robot.

We then need a well-defined measure capable of detecting the correlation between discrete distributions. Based on experimental validation (see Appendix I-C), we use the popular entropy-based measure: *KL-divergence*. For two 2D (r, g) histograms A and B with N bins along each dimension:

$$\mathcal{KL}(A, B) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (A_{i,j} \cdot ln \frac{A_{i,j}}{B_{i,j}}) \tag{10}$$

The more similar two distributions are, the smaller is the KL-divergence (KLD) between them. Since KLD is a function of the log of the observed color distributions, it is reasonably robust to large peaks in the observed distributions, and hence to images with large regions of a single color. The lack of symmetry in KLD is eliminated using the Resistor-Average KLD (RA-KLD) (see Appendix II-A for details).

Given a set of pdfs of pixel values corresponding to M different illuminations, we have previously shown that it is possible to effectively classify the test image histogram into one of the illumination classes [34]. A major limitation to this approach was that the illumination conditions had to be known in advance, and color maps had to be trained for each illumination. Here we make a significant extension in that *we do not need to know the different illuminations ahead of time*.

In addition to a set of of (r, g) histograms corresponding to an illumination ($rg_{samp}[i]$), the robot calculates the RA-KLD between every pair of histograms. The resultant distribution of the distances between the histograms under a particular illumination, say D_i, is modeled as a Gaussian that provides a second order statistic representing the illumination. When the illumination changes significantly, the average RA-KLD distance between a test (r, g) pdf and $rg_{samp}[i]$ maps to a point well outside the 95% range of the intra-illumination distances (D_i), providing a measure for detecting a change in illumination conditions.

B. Algorithm

Our algorithm for detecting illumination changes is summarized in Algorithm 3 and described in the text below.

The robot begins by learning the color map for the current illumination, by generating a suitable motion sequence using the world model, as described in Algorithm 1 (line 2). The color learning process takes less than five minutes, and we implicitly assume that the illumination does not change significantly during this period. Next, it moves around its environment and collects sample image histograms in (r, g) that represent this illumination. It also computes the distribution of RA-KLD distances, modeling it as a Gaussian ($D_{currIll}$), i.e. the mean and standard deviation of the distances describe the distribution (line 3).

Algorithm 3 Illumination Change Detection

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 motion sequence and learn colors - Algorithm 1.
3: Generate $rg_{samp}[current][]$, N (r, g) space histograms, 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 histogram.
7:   **for** $i = 0$ to $M - 1$ **do**
8:     $dAvg_{test}[i] = \frac{1}{N} \sum_j \underline{KLDist}(rg_{test}, rg_{samp}[i][j])$
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 1.
16:     Save (r, g) pdfs for new illumination.
17:     Generate the distribution of RA-KLD distances.
18:     Transition to the new color map for subsequent operations.
19:     $current = M$, $M = M + 1$.
20:   **end if**
21:   $testTime = currentTime$.
22: **end if**

Periodically ($time_{th} = 0.5$, line 5), the robot generates a test image histogram (rg_{test}, line 6) and computes its average distance to each set of previously computed (r, g) pdfs ($rg_{samp}[i]$ — lines 7-9). If the average distance lies within the threshold range (95%) of a known distribution of distances (D_i) other than the current one, the robot transitions to the corresponding illumination. The corresponding color map is used for all subsequent operations (lines 12, 13). But, if the average distance lies outside the threshold range of all known distribution of distances (line 14), the robot represents the current environmental state as a *new* illumination. It then proceeds to learn a color map using the autonomous color learning approach in Algorithm 1 (line 15). It also collects image statistics, i.e. image histograms in (r, g) and the distribution of RA-KLD distances (lines 16, 17). The statistics are used in subsequent comparisons for change in illumination. Changing the threshold changes the resolution at which the illumination changes are detected but we found that the robot is able to handle minor illumination changes reasonably using the color map corresponding to the closest illumination. In more recent work, which we do not describe here, we have found that a Bayesian update can be used to smoothly track minor illumination changes and operate consistently at high accuracy levels. In practice, the robot ends up learning only three different illuminations over a range of illuminations intensities ($\approx 450lux - 1600lux$). With transition thresholds to ensure that a change in illumination is accepted *iff* it occurs at least 6 times in 10 tests, it smoothly transitions between the different color maps that it has learned. The entire process is performed *without manual supervision*.

C. Experimental Results

We are primarily interested in testing two facets of our algorithm: i) the ability to decide *When to learn*, i.e. the ability to detect illumination changes, and ii) the ability to quickly transition between illuminations for which a representation has already been learned.

1) When to Learn: In order to test the ability of the robot to detect illumination changes accurately, we had the robot learn colors and image histograms in (r, g) corresponding to a particular illumination. We then had the robot move in its environment chasing a color-coded ball, and randomly changed the illumination on the field by controlling the intensity of specific lamps. We repeated the experiment over different starting illuminations and tested the ability of the robot to detect significant illumination changes. Table III presents results averaged over 1000 trials. It is essentially the *confusion matrix* with the rows and columns representing the ground truth and observed values respectively.

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

TABLE III: Illumination change detection: few errors in 1000 trials.

We observe that the robot detects illumination changes with very few false positives (second row, first column) or false negatives (first row, second column). Highlights and shadows are the major reasons for the errors, which are handled by not accepting a change in illumination unless it is observed over a few consecutive frames. Whenever the robot decides to learn a new color map, it is able to do so using the planned color learning algorithm (Section IV-B). When the algorithm is run with the illumination varying between ($\approx 450lux - 1600lux$), the robot ended up learning color maps and image statistics for three different cases corresponding approximately to 1600 lux, 1000 lux and 450 lux.

2) Transitions between illumination conditions: To test the robot's ability to transition between known illuminations, we chose the three discrete illumination intensities corresponding to the color maps that the robot had previously learned *Bright(1600lux), Dark(450lux)* and *Intermediate(1000lux)*. The intensity of the overhead lamps was changed to one of these conditions once every ≈ 10 sec. Table IV shows the results averaged over ≈ 150 trials each.

Illumination	Transition Accuracy	
	Correct (%)	Errors
Bright	97.3	4
Dark	100	0
Intermediate	96.1	6

TABLE IV: Illumination transition accuracy: few errors in ≈ 150 trials.

The first column represents the transitions to the corresponding illumination. Once again the few false transitions, mainly due to shadows or highlights, are quickly detected and corrected in the subsequent tests.

Next, we tested the ability to transition between the three illuminations while performing the *find-and-walk-to-ball* task, wherein the robot, starting from a fixed position, turns in place to find the ball (also at a fixed position) and walks up to it. Without any change in illumination the robot takes $6.7(\pm 0.6)$ seconds to perform this task. The robot starts off under one illumination and after 1.5 seconds (the time it takes to turn and see the ball), the illumination is changed by adjusting the intensity of all the lamps. The robot is timed as it performs the task. With a single color map, when the illumination condition changes significantly, the robot is unable to see a ball that is right in front of it, and cannot complete the task even if given unlimited time. With our algorithm, when the illumination changes, the robot seems lost for a couple of seconds while it recognizes the change and then transitions to the suitable color map. It then functions as normal, finding the ball and walking to it again. The results are shown in Table V.

Lighting (start/after 1.5 seconds)	Time (seconds)
Bright / Interim	8.5 ± 0.9
Bright / Dark	11.8 ± 1.3
Interm / Bright	8.6 ± 1.0
Interm / Dark	9.6 ± 3.1
Dark / Interm	11.5 ± 1.4
Dark / Bright	10.7 ± 1.1

TABLE V: Time taken to *find-and-walk-to-ball* under changing illumination.

The increase in the time taken to perform the task is due to the time taken to detect the change in illumination and transition to the appropriate color map. The values in the table are different for different transitions because the corresponding transition thresholds (for noise filtering) are different to reflect the fact that different transitions have different likelihoods. For example, a sudden transition from *Bright* to *Dark* is less likely than a transition from *Bright* to *Interm*. Complete details on the actual threshold values and the experiments that determine their choice can be found in [34]. Videos showing the robots performing under varying illuminations are available online. [8]

In an attempt to explore the robustness of our approach, we finally tested the algorithm for illuminations in between the ones that the robot ended up learning color maps for. These test illuminations would not register as being significantly different from the known illumination representations, and the robot would not learn new color maps for them. To enable comparison of these results, we recorded the time taken by the robot to *find-and-walk-to-ball*. In Table VI we present the values corresponding to the case where the robot starts off under the *Bright* illumination. About 1.5 seconds later, the illumination is changed such that it is between the *Bright* and the *Interm* illuminations (we also tested for the illumination midway between *Interm* and *dark*).

[8] www.cs.utexas.edu/∼AustinVilla/?p=research/illumination_invariance

Lighting	Time (seconds)
bet. bright and interm	12.27 ± 0.5
bet. interm and dark	13.3 ± 2.0

TABLE VI: Time taken (in seconds) to *find-and-walk-to-ball*

We conclude that even when the illumination is changed to one that is in between those that were significant enough to learn a new color map for, the robot transitions to using the *closest* illumination representation and is able to perform its tasks well. The increase in the time taken to perform the task is, once again, a result of the time taken to detect the change in illumination and transition to the appropriate color map.

D. Summary - Illumination Invariance

We have presented an approach that enables the robot to *autonomously* detect changes in illumination robustly and efficiently, *without prior knowledge of the different illuminations*. Each discrete illumination is characterized by a color map and a set of image histograms in the (r, g) color space, both of which are generated by the robot. The image histograms are used to generate second-order statistics that represent a particular illumination. When the robot detects an illumination that it had already learned a representation for, it smoothly transitions to using the corresponding color map. If it detects a new illumination, it automatically learns a new suitable color map and collects image statistics to be used in subsequent comparisons for change in illumination. Even when presented with illuminations that are in between the ones that it already has learnt color maps for, and which, by definition, are not significantly different from the known illuminations, it smoothly selects the *closest* illumination and transitions to the appropriate color map. The whole process is autonomous and proceeds without human supervision.

The algorithm is also applicable to other domains. In the *car-on-the-road* task, the vision system could learn a color map for sunny conditions. When illumination changes, such as when the sun goes behind a cloud (or sets in the evening), the system would detect it and adapt to this change by learning a new color map. When the sun comes back, the system would automatically switch back to the previous color map.

VI. Summary - Overall

The main aim of our work is to enable a mobile robot to perform autonomously in its environment, using the structure that is inherent in its environment. We first developed a vision system that tackled challenging vision problems such as segmentation, object recognition and line detection in real-time, under rapid camera motions, though it required manual calibration and was sensitive to illumination changes. Next, we designed an algorithm that enabled the robot to autonomously plan its motion sequence and learn the colors using the structure inherent in its environment. Finally, we also presented a scheme for the robot to automatically detect significant changes in illumination and use the color learning scheme to adapt to these changes. The overall system learns colors autonomously, and detects and adapts to significant illumination changes, thereby recognizing objects and performing its tasks efficiently. Next, we describe some related approaches to color learning/segmentation and illumination invariance.

VII. Related Work

In this section, we review some related approaches to color learning and illumination invariance, comparing them with our algorithms to motivate our approach.

A. Segmentation and Learning

Color segmentation is a well-researched field in computer vision with several good algorithms, for example mean-shift [5] and gradient-descent based cost-function minimization [6]. The mean-shift algorithm is a 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 (pdf) using a density estimation-based clustering approach. Dense regions in the feature space correspond to local maxima, i.e. the modes of the unknown pdf. 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. It provides good performance on vision tasks such as segmentation and tracking, but its quadratic complexity makes it expensive to perform on mobile robots with computational constraints.

Active contours are another set of popular methods for image segmentation. The method defines initial contours and then deforms them towards object boundaries. The methods can be classified into three groups: edge-based, region-based and hybrid. Manjunath et al. describe a region-based method [6] that segments images into multiple regions and integrates an edge-flow vector field-based edge function for segmenting precise boundaries. The 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, and provides good segmentation results on a variety of images, but the iterative optimization makes it expensive to implement on mobile robots.

Even in the RoboCup domain, several algorithms have been implemented for color segmentation. The baseline approach creates mappings from the YCbCr values ($0 - 255$ in each dimension) to the color labels [21]. Other methods include the use of decision trees [19] and the creation of axis-parallel rectangles in the color space [20]. All these approaches involve the hand-labeling of several (≈ 30) images over a period of an hour or more before the decision-tree/color map can be generated. Our baseline approach for color segmentation is a variant of these approaches, with some additional features to make it more robust to shadows and highlights (Section III-A).

Attempts to learn colors or make them independent to illumination changes have produced reasonable success [35], [36] 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. Mobile robots typically require real-time operation and frequently operate under dynamically changing environments.

The choice of color space is an important consideration in color learning and segmentation. Gevers and Smeulders evaluate several color spaces to determine their suitability for recognizing multicolored objects invariant to significant changes in viewpoint, object geometry and illumination [36]. They present a detailed theoretical and experimental analysis of the following models: RGB, Intensity I, normalized color *rgb*, saturation S, Hue H, and three models that they propose $c_1c_2c_3$, $l_1l_2l_3$, $m_1m_2m_3$. They show that assuming dichromatic reflection and white illumination, normalized *rgb*, saturation S and Hue H, and the newly proposed $c_1c_2c_3$, $l_1l_2l_3$ 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. The work provides a good reference on the choice of color spaces.

Lauziere et al. 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 [35]. They further explain the process of camera characterization in [37]. The color camera sensitivity curves are measured and used to recognize objects better under daylight illumination conditions. Mobile robots operating the real world frequently need to be deployed in a short period of time in previously unseen locations.

Attempts to automatically learn the color map in the legged league have rarely been successful. Cameron and Barnes [38] present an approach that detected edges in the image and constructed closed figures to find image regions corresponding to known environmental features. The color information from these regions was used to build the color classifiers, using the Earth Mover's distance (EMD) [39] as the similarity metric. The changes introduced by illumination changes are tracked by associating the current classifiers with the previous ones. The edge detection, closed figure formation and clustering makes the approach time consuming even with the use of off-board processing. Our algorithm exploits domain knowledge but learns colors in real-time on-board the robot using an efficient color model.

Jungel presents another approach where the color map is learned using three layers of color maps with increasing precision levels [40]. Colors in each level are represented as cuboids, but colors that are *close* to each other are not disambiguated. 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 color regions are displaced in the color space by the same amount. But different colors do not actually shift by the same amount with illumination changes. The generated map is reported to be not as accurate as the hand-labeled one. Our algorithm learns a color map in *under five minutes of robot time*, and provides performance comparable to the hand-labeled map obtained after an hour or more of human effort. It works both within the constrained lab setting and in less controlled settings outside the lab.

B. Illumination Invariance

In its most general form, the problem of color constancy can be explained using the following equation [11]:

$$m_j^p = \int \left(E(\lambda) S^{\mathbf{p}}(\lambda) R_j(\lambda) \right) d\lambda \qquad (11)$$

Here, $E(\lambda)$ is the spectral power distribution of the illuminant, $S^{\mathbf{x}}(\lambda)$ is the surface reflectance at a scene point $\mathbf{x}$, while $R_j(\lambda)$ is the spectral response (relative) of the imaging device's *j*th sensor. The response of the *j*th sensor of the imaging device at pixel *p*, m_j^p, is the integral of the product of these three terms over the range of wavelengths. Changing either the surface reflectance function or the spectral power distribution of the illuminant can change the response at the sensor. Color constancy requires that we either transform the response m_j^p to correlate with $S(\lambda)$ independent of $E(\lambda)$, or equivalently, recover an estimate of $E(\lambda)$. Several approaches have been attempted to solve this problem, with varying levels of success. However, almost all of them have been applied to static images and most of them have high computational complexity.

The Retinex theory was one of the first attempts to explain human color constancy [41]. Based on the assumption that white reflection induces maximal *rgb* camera responses (since light incident on a white patch is spectrally unchanged after reflection), it suggested that measuring the maximum r, g, and b responses can serve as an estimate of the scene illuminant. When it was determined that the maximum *rgb* in an image is not the correct estimate for white, the technique was modified to be based on global or local image color averages. The "Gray World" algorithm by Buchsbaum [42] is also based on the same principle. Unfortunately, the image average, either local or global, has been shown to correlate poorly with the actual scene illuminant [43]. Also this method excludes the possibility of distinguishing between the actual changes in illumination and those as a result of a change in the collection of surfaces in the scene under consideration.

Forsyth proposed the *gamut mapping* algorithm for color constancy [12]. Based on the fact that surfaces can reflect no more light than is cast on them, he concluded that the illuminant color is constrained by the colors observed in the image and can hence be estimated using image measurements alone. The algorithm generated a set of mappings that transformed image colors (sensor values) under an unknown illuminant to the gamut of colors observed under a standard (canonical) illuminant using 3D diagonal matrices. Then a single mapping was chosen from the feasible set of mappings.

Realizing that the scene illuminant intensity cannot be recovered in Forsyth's approach, Finlayson modified the algorithm to work in 2D chromaticity space [44]. He then proved that the feasible set calculated by his 2D algorithm was the same as that calculated by Forsyth's original algorithm, when projected into 2D, and proposed the *median selection* method to include a constraint on the possible color of the illuminant into the gamut mapping algorithm [45]. More recently he presented a correlation framework [11], where instead of recovering a single estimate of the scene illuminant, he measured the likelihood that each of a possible set of illuminants is the scene illuminant. The range of sensor values that can occur under each of a possible set of illuminants is calculated and once the required likelihoods are obtained by correlating with the colors in a particular image, they are used to determine a single estimate of the scene illuminant. In addition to extensive computation, the approach requires prior knowledge of the illuminations which is not feasible in a mobile robot domain.

Brainard and Freeman tackle the problem using the Bayesian decision theory framework, which combines all available statistics such as gray world, subspace and physical realizability constraints [46]. They model the relation among illuminants, surfaces and photosensor responses and generate a priori distributions to describe the probability of existence of certain illuminants and surfaces. A maximum local mass (MLM) estimator integrates local probabilities and uses Bayes' rule to compute the posterior distributions for surfaces and illuminants, for a given set of photosensor responses. Similar to the above-mentioned methods, it requires significant prior knowledge and is computationally expensive.

Tsin et al. present a Bayesian MAP (*maximum a posteriori*) approach to achieve color constancy for outdoor object recognition with a static surveillance camera [47]. Static overhead high-definition color images, over several days, are used to learn statistical distributions for reflectance and the light spectrum. A linear iterative updating scheme is used to converge to the classification result on the test images. A mobile robot system needs to be robust to rapid camera motions and dynamic changes.

In contrast to the Bayesian methods, Rosenberg et al. present an approach where they develop models for sensor noise, canonical color and illumination [13]. Then the global scene illumination parameters are determined by an exhaustive search using KL-divergence as the metric. They present results to show that proper correction is achieved for changes in scene illumination and compare it with the results obtained using a MLE (maximum likelihood estimate) approach. Once again, the method requires extensive prior knowledge and is computationally expensive.

Schulz and Fox estimate colors using a hierarchical bayesian model with *Gaussian* priors and a joint posterior on position and environmental illumination [48]. Significant prior knowledge of color distributions and illuminations, in addition to extensive hand-labeling, are required even when tested under two distinct illuminations and a small set of colors. In addition, it requires almost a second of off-board processing per image. Our approach enables the robot to model overlapping colors with no apriori knowledge of color distributions, and detect and adapt to a range of illuminations using autonomously-collected image statistics.

Lenser and Veloso present a tree-based state description/identification technique [49], which they use for detecting changes in lighting on Aibo robots. They incorporate a time-series of average screen illuminance to distinguish between illumination conditions, using the absolute value distance metric to determine the similarity between distributions. We however believe that the color space distributions could function as a better discriminating feature. Also, their method is *not* run on-board the robot while it is performing other tasks.

Anzani et al. describe an attempt at illumination invariance in the RoboCup middle-size league [50], where teams are made up of mobile robots (wheeled locomotion). They use *Mixture of Gaussians* to generate multi-modal distributions for the various colors. The labeling of color classes and association with mixture components is done by human supervision, and the Bayesian decision rule is used during the classification stage. To adapt the model parameters to changing illumination conditions, the EM algorithm [51] is used with online adaptation of the number of mixture components too. The algorithm has been tested only over a few illuminations in the lab, while we model colors and adapt to a range of illuminations even in un-engineered indoor corridors.

In the domain of mobile robots, the problem of color constancy has often been avoided by using non-vision-based sensors such as laser range finders and sonar sensors [4]. Even when visual input is considered, the focus has been on recognizing just a couple of well-separated colors [2], [52]. There has been relatively little work on illumination invariance with a moving camera in the presence of shadows and artifacts caused by the rapid movement in complex problem spaces. Further, with few exceptions (e.g. [49], [50]), the approaches that do exist for this problem cannot function in real-time with the limited processing power that we have at our disposal.

In the DARPA grand challenge, Thrun et al. [3] model colors as MoG and attempt to add additional Gaussians and/or modify the parameters of the existing Gaussians in response to the changes in illuminations. But, not all colors are modeled well using MoG. Furthermore, they were interested only is distinguishing safe regions on the ground from the unsafe regions and did not have to model overlapping color classes separately. Our approach (Section V) has the robot use the autonomously built representations for illumination to detect and adapt to significant changes in illumination, thereby performing its tasks over a range of illuminations.

VIII. Conclusions and Future Work

In this article we have introduced algorithms that address some challenging mobile robot vision problems. We first presented a prototype vision system that performs *color segmentation* and *object recognition* in real-time, under rapid camera motion and image noise. Here we used manual color calibration and assumed a fixed and uniform illumination. Next, we drastically reduced the color calibration time from an *hour* or more of *human effort* to *less than five minutes of robot time* by making the robot autonomously plan its motion sequence and learn the color distributions by efficiently utilizing the structure of the environment - known descriptions of color coded objects. The algorithm *bootstraps*, with the learned colors being used to segment and recognize objects, thereby localizing better to locations suitable for learning other colors. Finally, we also made the vision system robust to illumination changes by autonomously detecting and adapting to significant illumination changes. The illuminations were represented by color-space distributions and image statistics, and the robot transitions between the learned color maps, or learns new ones, as required.

Our on-going research includes extending the work reported in this article in three ways. First, we are working on the design of an algorithm that enables the robot to learn colors outdoors. This is a much more challenging problem where the robot may have to use other features, in addition to color, to represent objects of interest because the objects are less likely to be constant-colored and the range of illuminations can be much larger. In addition, we also plan to make the planning parts more robust to failures based on motion model errors, especially in the initial stages of learning where the lack of visual information makes the robot very vulnerable. The planning aspects can also be improved by having the robot learn the optimal functions, based on approaches such as reinforcement learning, instead of the current *greedy* approach of minimizing heuristics.

Second, the current illumination adaptation scheme has the robot detect significant illumination changes and re-learn the entire color map when necessary. But this approach can be made more robust by having the robot continuously modify its color map for minor illumination changes as well. The algorithm needs to have some means of detecting minor shifts in the color distributions and then adapting to these changes by selectively updating specific color distributions.

Third, we aim to enable the robot to learn colors from an *unknown* initial position. This is a challenging problem because the robot has to reason under a lot of uncertainty. It would need efficient error detection and correction mechanisms.

The problems in robot vision are very challenging and far from being solved. This work represents a step towards towards solving the daunting problem of developing efficient algorithms that enable a mobile robot to function autonomously under completely uncontrolled natural lighting conditions, with all its associated variations.

APPENDIX I
COMPARISON MEASURES

In order to compare image distributions, we need a well-defined measure capable of detecting the correlation between distributions under similar illumination conditions. Here we propose and examine two such measures: a correlation measure and the popular KL-divergence measure [53].

A. Correlation measure

Consider the case where we have two distributions A and B with N bins along each dimension, the correlation between the two can be computed as,

$$Cor(A,B) = \sum_{i=0}^{N-1} (A_i \cdot B_i) \tag{12}$$

The more similar the two distributions are, the higher is the correlation between them. This is a simple probabilistic representation of the similarity between two distributions.

B. KL-divergence (KLD) measure

The Kullback-Liebler divergence is a popular entropy-based measure for comparing distributions [32]. For the two distributions A and B mentioned above, we have:

$$\mathcal{KL}(A,B) = \sum_{i=0}^{N-1} (A_i \cdot ln\frac{A_i}{B_i}) \tag{13}$$

As mentioned in Section V-A the more similar two distributions are, the smaller is the KL-divergence between them. Since the KLD measure is a function of the log of the observed color distributions, it is reasonably robust to large peaks in the observed distributions, and hence to images with large regions of a single color.

C. Correlation vs. KL-divergence

In order to compare the performance of the two measures, sample images were collected from the robot's camera at four different positions with the robot standing upright. At each position, seven different illuminations were considered, ordered from the brightest to the darkest, resulting in 28 samples. The illuminations were generated by adjusting the intensity of the lamps in specific patterns. For the histogram corresponding to each image in this set, both measures were used to compute the *closest* image histogram among the others in the set. Table VII shows the results.

Method	*Correct*	*Off-by-one*	*Off-by-two*	*Incorrect*
Correlation	8	9	8	3
KL-divergence	15	13	0	0

TABLE VII: Classification results using Correlation and KLD

The *Off-by-one* column refers to the case where an image is classified as being from an illumination class that is one illumination away from the true class. *Incorrect classification* represents the cases where the classification result is 3 or more classes away from the true illumination class. The results are grouped in this manner because during task execution, when the illumination is changed to conditions similar to those under training, being off by one illumination class does not make any significant difference in color segmentation. The results are a lot different though when the robot is off by several illumination classes. For the seven class problem, the Correlation-based classification is off by two classes or incorrect in several cases. The classification based on KLD is correct in many cases and even when it is wrong, it is off only by one illumination class. Based on these experiments and the robustness to large peaks of a single color, the KLD measure was chosen for comparing distributions. Since the measure is not symmetric, some modifications were made, as described in Section II-A.

APPENDIX II
VALIDATION OF GAUSSIAN ASSUMPTION

Here we present the validation for the Gaussian assumption made in our approach to autonomous color learning on the robot (Section IV).

We need to analyze the *goodness-of-fit* of the *Gaussian* to the color distributions. To do so, we chose the method of *bootstrapping* [30] using *Resistor-Average Kullback-Liebler divergence* (RA-KLD) [53] as the distance measure.

Figure 13 shows the estimated and actual sample points for one of the colors in the robot's environment. The *Maximum Likelihood Estimate* (MLE) [51] of the actual samples are used

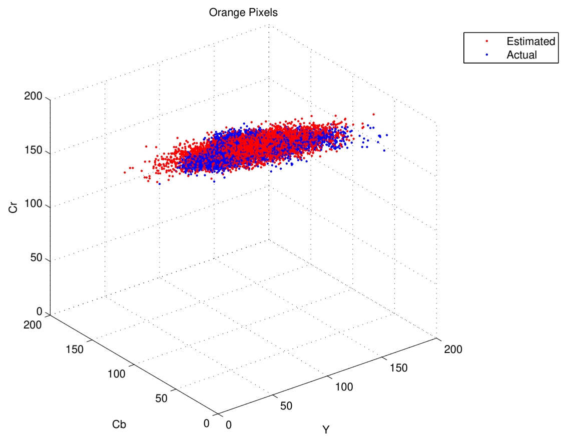


Fig. 13: Estimated and Actual samples of Orange

to define a 3D Gaussian distribution. The estimated points are obtained by drawing random samples from this Gaussian.

We observe that the estimated *Gaussian* function does do a good job of approximating the actual distributions.

A. Resistor-Average KLD

As described in Appendix I-B, KLD is a robust information theoretic measure that has been used extensively to compare distributions. The Krichevsky-Trofimov correction [54] is used to handle the fact that a zero-value in one distribution with a non-zero value in the same bin in the other distribution would result in a KLD value of infinity. In addition, KLD is not a true metric because it is neither symmetric nor does it satisfy the triangle-inequality. Hence, in our experiments we use the Resistor-Average KLD, defined as:

$$\mathcal{R}(A,B) = \frac{\mathcal{KL}(A,B)\cdot\mathcal{KL}(B,A)}{\mathcal{KL}(A,B)+\mathcal{KL}(B,A)} \tag{14}$$

As mentioned in [53], one-half of the RA-KLD measure is a close approximation to the Chernoff Bound.

B. BootStrapping

The process of *bootstrapping* [30] is an established statistical procedure for bias removal and statistical distribution fit analysis. We use it for our *goodness-of-fit* tests:

- The samples of any color's distribution are obtained by selecting suitable image regions and building a histogram — *Observed*. Assume that there are M samples.
- Using the estimated *Gaussian* for that color, the same number of samples M are randomly drawn — *Actual*.
- For both histograms (observed and actual), values in the 3D bins are lined up to form the 1D distributions. RA-KLD is determined between these distributions — R_{obs}.
- *Observed* and *Actual* are randomly mixed together.
- From the jumbled up set of samples, two sets of M samples are randomly drawn (with replacement) and binned to determine the RA-KLD between them. This is repeated several hundred times and the distribution of distances is observed — $R_i\ i \in [1,200]$.
- Using the distribution of RA-KLD values and the original one (R_{obs}), a test (Z) statistic is determined. This in turn provides a p-value, which can be used to test the null hypothesis (H_0), also stated as: *the estimated Gaussian is a good fit for the sample points.*

The results of this process are tabulated in Table VIII. The table shows R_{obs} in the column *Orig* and also shows the mean and standard deviation of the estimated distribution of JS distances. The significance is decided on the basis of the p-values in the last column.

Color	RA-KLdists			Z_{stat}	p-val
	Orig	Mean	Stdev		
Orange	0.003808	0.004016	$3.5\cdot10^{-4}$	0.596	0.56
Green	$4.6\cdot10^{-4}$	$4.73\cdot10^{-4}$	$6.82\cdot10^{-5}$	0.18	0.86
Yellow	0.00126	0.00129	$1.98\cdot10^{-4}$	0.155	0.88
Blue	0.0017	0.0015	$2.34\cdot10^{-4}$	0.90	0.38
White	0.006	0.0057	$5.7\cdot10^{-4}$	0.81	0.42

TABLE VIII: Quality of fit based on RA-KLdists

The probability value (p-value) of a statistical hypothesis test is the smallest level of significance that would lead to the rejection of the null hypothesis H_0 with the given data [55], i.e. it is the significance level of the test for which the null hypothesis would be just rejected. The smaller the p-value, the more convincing is the rejection of the null hypothesis. Stated differently, if the level of significance α is greater than the p-value, H_0 can be rejected. Based on values in the table above (high p-values), we clearly fail to reject H_0.

Acknowledgment

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

References

[1] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *RAS Special Issue on Socially Interactive Robots*, 2003.

[2] B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire, "Low-order-complexity vision-based docking," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 922–930, 2001.

[3] S. Thrun, "Stanley: The Robot that Won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[4] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Journal of Artificial Intelligence*, 2001.

[5] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[6] 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.

[7] A. L. N. Fred and A. K. Jain, "Robust data clustering," in *The International Conference of Computer Vision and Pattern Recognition*, June 2003, pp. 128–136.

[8] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[9] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *Pattern Analysis and Machine Intelligence*, April 2002.

[10] A. Torralba, K. P. Murphy, and W. T. Freeman, “Sharing visual features for multiclass and multiview object detection,” in *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Washington D.C., 2004.
[11] G. Finlayson, S. Hordley, and P. Hubel, “Color by correlation: A simple, unifying framework for color constancy,” *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, November 2001.
[12] D. Forsyth, “A novel algorithm for color constancy,” *In International Journal of Computer Vision*, vol. 5, no. 1, pp. 5–36, 1990.
[13] C. Rosenberg, M. Hebert, and S. Thrun, “Color constancy using kl-divergence,” in *In IEEE International Conference on Computer Vision*, 2001.
[14] F. Farshidi, S. Sirouspour, and T. Kirubarajan, “Active multi-camera object recognition in presence of occlusion,” in *The IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2005.
[15] “The Sony Aibo robots,” 2004, http://www.sonystyle.com.
[16] H. Kitano, M. Asada, I. Noda, and H. Matsubara, “Robot world cup,” *Robotics and Automation*, vol. 5, no. 3, pp. 30–36, 1998.
[17] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.
[18] C. Pantofaru and M. Hebert, “A comparison of image segmentation algorithms, cmu-ri-tr-05-40,” Robotics Institute, Carnegie Mellon University, Tech. Rep., September 2005.
[19] 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*. Berlin: Springer Verlag, 2002.
[20] D. Cohen, Y. H. Ooi, P. Vernaza, and D. D. Lee, *RoboCup-2003: The Seventh RoboCup Competitions and Conferences*. Berlin: Springer Verlag, 2004.
[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.
[22] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2002.
[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (Second Edition)*. MIT Press, September, 2001.
[24] P. Stone, K. Dresner, P. Fidelman, N. K. Jong, N. Kohl, G. Kuhlmann, E. Lin, M. Sridharan, and D. Stronger, “UT Austin Villa 2004: Coming of Age, TR 04-313,” Department of Computer Sciences, UT-Austin, Tech. Rep., October 2004.
[25] T. Rofer and M. Jungel, “Vision-based fast and reactive monte-carlo localization,” in *The IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003, pp. 856–861.
[26] “Least Squares Line Fitting,” http://mathworld.wolfram.com/LeastSquaresFitting.html.
[27] 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.
[28] M. Sridharan and P. Stone, “Real-time vision on a mobile robot platform,” in *The IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2005.
[29] ——, “Autonomous color learning on a mobile robot,” in *The Twentieth National Conference on Artificial Intelligence (AAAI)*, 2005.
[30] B. Efron and R. J. Tibshirani, *An Introduction to Bootstrap*. Chapman and Hall Publishers, 1993.
[31] M. Swain and D. H. Ballard, “Color indexing,” *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.
[32] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley Publishing House, 1991.
[33] M. Sridharan and P. Stone, “Autonomous planned color learning on a mobile robot without labeled data,” in *The Ninth IEEE International Conference on Control, Automation, Robotics and Vision (ICARCV)*, December 2006.
[34] ——, “Towards illumination invariance in the legged league,” in *The RoboCup Symposium*, 2004.
[35] 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*, vol. 3826, June 1999, pp. 86–100.
[36] T. Gevers and A. W. M. Smeulders, “Color based object recognition,” *In Pattern Recognition*, vol. 32, no. 3, pp. 453–464, 1999.
[37] Y. B. Lauziere, D. Gingras, and F. P. Ferrie, “Color camera characterization with an application to detection under daylight,” in *Vision Interface (VI)*, May 1999, pp. 280–287.
[38] D. Cameron and N. Barnes, “Knowledge-based autonomous dynamic color calibration,” in *The Seventh International RoboCup Symposium*, 2003.
[39] Y. Rubner, C. Tomasi, and L. J. Guibas, “The earth mover’s distance as a metric for image retrieval,” *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.
[40] M. Jungel, “Using layered color precision for a self-calibrating vision system,” in *The Eighth International RoboCup Symposium*, Lisbon, Portugal, 2004.
[41] E. H. Land, “The retinex theory of color constancy,” *Scientific American*, pp. 108–129, 1977.
[42] G. Buchsbaum, “A Spatial Processor Model for Object Color Perception,” *Journal of Franklin Institute*, vol. 310, pp. 1–26, 1980.
[43] D. H. Brainard and B. A. Wandell, “Analysis of the retinex theory of color vision,” *Journal of Optical Soceity of America A*, vol. 3, no. 10, pp. 1651–1661, 1986.
[44] G. Finlayson, “Color in perspective,” *In IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 1034–1038, July 1996.
[45] G. Finlayson and S. Hordley, “Improving gamut mapping color constancy,” *In IEEE Transactions on Image Processing*, vol. 9, no. 10, October 2000.
[46] D. H. Brainard and W. T. Freeman, “Bayesian color constancy,” *Journal of Optical Soceity of America A*, vol. 14, no. 7, pp. 1393–1411, 1997.
[47] Y. Tsin, R. T. Collins, V. Ramesh, and T. Kanade, “Bayesian color constancy for outdoor object recognition,” *In IEEE Pattern Recognition and Computer Vision*, December 2001.
[48] 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.
[49] S. Lenser and M. Veloso, “Automatic detection and response to environmental change,” in *The International Conference of Robotics and Automation (ICRA)*, May 2003.
[50] F. Anzani, D. Bosisio, M. Matteucci, and D. G. Sorrenti, “On-line color calibration in non-stationary environments,” in *The Ninth International RoboCup Symposium*, Osaka, Japan, July 18-19 2005.
[51] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley Publishers, 2000.
[52] J. Hyams, M. W. Powell, and R. R. Murphy, “Cooperative navigation of micro-rovers using color segmentation,” *In Journal of Autonomous Robots*, vol. 9, no. 1, pp. 7–16, 2000.
[53] D. H. Johnson, C. M. Gruner, K. Baggerly, and C. Seshagiri, “Information-theoretic analysis of neural coding,” *In Journal of Computational Neuroscience*, vol. 10, pp. 47–69, 2001.
[54] R. E. Trichevsky and V. K. Trofimov, “The performance of universal coding,” *In IEEE Transactions of Information Theory*, vol. 27, pp. 199–207, 1981.
[55] D. C. Montgomery and G. C. Runger, *Applies Statistics and Probability for Engineers*, 3rd ed. Wiley Publishers, 2002.