Autonomous Planned Color Learning on a Legged Robot

Mohan Sridharan[1] and Peter Stone[2]

[1] Electrical and Computer Engineering
smohan@ece.utexas.edu
[2] Department of Computer Sciences,
The University of Texas at Austin
pstone@cs.utexas.edu

Abstract. Our research focuses on automating the color-learning process on-board a legged robot with limited computational and memory resources. A key defining feature of our approach is that instead of using explicitly labeled training data it trains autonomously and incrementally, thereby making it robust to re-colorings in the environment. Prior results demonstrated the ability of the robot to learn a color map when given an executable motion sequence designed to present it with good color-learning opportunities based on the known structure of its environment. This paper extends these results by demonstrating that the robot can plan its own such motion sequence and perform just as well at color-learning. The knowledge acquired at each stage of the learning process is used as a bootstrap mechanism to aid the robot in planning its motion during subsequent stages.

Keywords: Robot Vision, Color Learning.

1 Introduction

The first step for most teams, upon arrival at RoboCup, in any of the real robot leagues, is *color calibration*: mapping raw camera pixels to color labels such as *white* or *pink*. Due to differences in lighting conditions and object colors between the teams' labs and the competition venue, pre-trained vision modules are unlikely to work "out of the box." Also, the time required for color calibration (more than an hour in the legged league) leads to multiple days of setup time before each competition, a costly proposition from the perspective of reserving the venue. But both soccer-playing and rescue robots must eventually be able to operate in natural, changing lighting conditions, as soon as possible after arriving on site. One way to dramatically reduce this time is to enable the robot to autonomously learn the desired colors from the environment.

The most common approach to color calibration is manual labeling of a small subset of the color space, which is used to label the values of nearby pixels and produce the color map. Instead, we specify the properties of objects in the robot's environment (locations, color labels, and sizes), but no information on the pixel values corresponding to the colors is given. The known locations and *structure* (color labels and sizes) of objects are used to seed the color-learning process and

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 270–278, 2007.

plan the corresponding motion sequence. When illumination conditions change, assigning pixel-label biases could require human supervision each time, or fail altogether. Our method does not suffer from this problem since no information is needed regarding the pixel values that correspond to each color.

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

Building on our previous work [9], where the robot learnt colors by moving through a pre-defined motion sequence that was generated manually, here we enable the robot to autonomously plan its motion sequence for any given configuration of objects, based on environmental knowledge and heuristic constraints on its motion sequence. Further, it simultaneously learns colors and localizes, and incrementally performs better at both these tasks.

2 Background Information

The SONY Aibo, *ERS-7*, is a four legged robot whose primary sensor is a CMOS camera with a field-of-view of 56.9^o (hor) and 45.2^o (ver), providing the robot with a limited view of its environment. The images have a resolution of 208×160 pixels and are captured in the *YCbCr* format at $30Hz$. The robot has 20 degrees-of-freedom (dof). It also has noisy touch sensors, IR sensors, and a wireless LAN card for inter-robot communication. The camera jerks around a lot due to the legged locomotion, and images possess common defects such as noise and distortion. Figure 1 shows the robot and the $4.4m \times 2.9m$ playing field.

On the robot, visual processing typically occurs in two stages: color segmentation and object recognition (see [6]). Color segmentation is a well-researched field in computer vision with several good algorithms [4,10]. But these involve computation that is infeasible to perform on autonomous robots with computational and memory constraints. In the RoboCup domain, the methods applied range from the baseline approach of creating mappings from the YCbCr values to the color labels [2], to the use of decision trees [11] and axis-parallel rectangles in the color space [3]. All of them involve an elaborate training process where the color map is generated by hand-labeling several (≈ 25) images over a period of at least an hour.

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

The color map is used to segment the image and construct connected constant-colored regions, which are used to detect useful objects (e.g. markers). The robot uses the markers to localize itself on the field and coordinates with its teammates to score goals on the opponent. All processing, for vision, localization, locomotion, and action-selection, is performed on board the robots, using a 576MHz processor. Though games are currently played under constant and reasonably uniform lighting conditions, a change in illumination over several days often forces teams to re-calibrate the vision system. Also, the overall goal of eventually playing against humans in natural lighting puts added emphasis on the ability to learn the color map in a very short period of time.

Attempts to automatically learn the color map on the Aibos have rarely been successful. In one approach, edges are detected and closed figures are constructed to find image regions corresponding to known environmental features [1]; color information from these regions is used to build the color classifiers. This is time consuming even with the use of offline processing and requires human supervision. In [7], a color map is learnt using three layers of color maps, with increasing precision levels. This is still not as accurate as the hand-labeled one and additional constraints are required to disambiguate the colors. Schulz and Fox [8] present another example where colors are estimated using a hierarchical Bayesian model with *Gaussian* priors.

Our approach does not need color priors. It enables the robot to *autonomously plan* its motion to *learn* the color map, using the knowledge of location and *structure* of the objects, in less than *five minutes*. It involves very little storage and the resultant color map is comparable in segmentation accuracy to the hand-labeled one that take more than an hour of human effort. Note that we provide a world model instead of a color map and/or the motion component. This removes the manual-intensive component and enables the robot to function in different environmental settings.

3 Problem Specification

As described in [9], to recognize objects and operate in a color-coded world, a robot typically needs to recognize a certain discrete number (N) of colors ($\omega \in [0, N-1]$). A complete mapping identifies a color label for each possible point in the color space:

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

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

We represent each color by a three-dimensional (3D) Gaussian model with mutually independent color channels, i.e. no correlation among the values along the three color channels. Though more expressive color representations, such as histograms, have been used extensively in the literature, and the independence assumption does not hold perfectly in practice, we determined, using empirical data and the statistical technique of bootstrapping [5], that a 3D Gaussian model

with independent channels closely approximates reality. In addition to simplifying calculations the Gaussian has the advantage that the mean and variance are the only statistics that need to be stored for each color. This makes the learning process feasible to execute on mobile robots with constrained processing power.

Under the three-dimensional Gaussian model with independent channels, the *apriori* probability density functions (color $\omega \in [0, N-1]$) are given by:

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

where, $c_i \in [C_{i_{min}} = 0, C_{i_{max}} = 255]$ represents the value at a pixel along a color channel C_i while μ_{C_i} and σ_{C_i} represent the corresponding means and variances.

Assuming equal priors, the *aposteriori* probabilities for each color are:

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

For each pixel, the color label corresponds to the color that has the maximum *aposteriori* probability.

4 Autonomous Color Learning

Our learning algorithm is summarized in Algorithm 1 and specific details are described below. The basic color learning component (lines $9-14$) was described in [9] while the rest of the algorithm deals with the motion sequence planning.

The robot starts off at a known position in its world model and the locations of various color coded objects are known. The robot has no initial color information (means and variances of all colors are zero) but it has the list of colors to be learnt ($Colors[]$). It also has an array of structures ($Regions[][]$) — a list for each color. Each structure corresponds to an object of a particular color and stores a set of properties for that region, such as its size (length and width) and its three-dimensional location (x,y,z) in the world model. Both the starting pose of the robot and the object locations can both be varied between trials, which causes the robot to also modify the list of candidate regions for each color.

Given the robot's limited field of view, it is essential to adjust its pose to focus on objects with the colors of interest. This can be extremely challenging in the initial stages due to the inherent inaccuracy of the motion model (due to slippage) and the initial lack of visual information. Geometric constraints on the position of the objects are essential to resolve conflicts. These heuristic constraints depend on the robot and the problem domain. In our case, they are:

- No two objects should occupy the same position in the world model — there should be a minimum distance (600mm) between two objects.
- No two objects of the same dimensions can be within 90^o of each other (with respect to the corresponding robot position) if they each consist of only one unknown color.

Algorithm 1. Planned Autonomous Color Learning

Require: Known initial pose (but can be varied across trials).
Require: Color-coded model of the robot's world - objects at known positions that can change between trials.
Require: Empty Color Map; List of colors to be learnt - $Colors[]$.
Require: Arrays of colored *regions*, rectangular shapes in 3D space; $Regions[][]$. A list for each color, consisting of the properties (size, shape) of the regions of that color.
Require: Ability to navigate (approximately) 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: LearnGaussParams($Color$)
10: *Learn* Mean *and* Variance *of color from candidate image pixels*
11: UpdateColorMap()
12: **if** !Valid($Color$) **then**
13: RemoveFromMap($Color$)
14: **end if**
15: **else**
16: Rotate at target position.
17: **end if**
18: **if** $CurrTime - Time_{st} \geq Time[Color]$ or $RotationAngle \geq Ang_{th}$ **then**
19: $i = i + 1$
20: $Time_{st} = CurrTime$
21: **end if**
22: **end while**
23: Write out the color statistics and the Color Map.

The order in which the colors are to be learnt is computed dynamically and greedily (*BestColorToLearn()* — line 4); it chooses the best color one at a time without actually planning ahead for where it will be after learning that color. This is based on:

1. The amount of motion (distance) that is required to place the robot in a location suitable to learn the color.
2. The existence of a region that can be used to learn that color without requiring the knowledge of any other (as of yet) unknown color.

The goal is to learn colors with minimal motion, so as to increase the chances of being well-localized. Once a color order is chosen, for the first color in the list, ($Color$), the robot determines the *best candidate region* to learn that color from.

Once the candidate is determined, the robot calculates the pose that would be best suited to recognize this candidate region – *BestTargetPose()* (line 5).

The robot then determines (*RequiredMotion()* — line 6) and executes the motion sequence to place it in the target position. The motion to the target position is monitored (*visual feedback*) using the current knowledge of colors to recognize objects and localize to the correct location. Once it gets close to the target location, the robot searches for candidate regions that satisfy the heuristic constraints of size and shape for the region that it is looking for. The actual world-model definitions in the structure *Regions[Color][best-candidate-region]* are dynamically modified by the robot, based on its pose and standard geometric principles, to arrive at suitable constraints.

The robot stops when either the candidate region is found or the target position is reached. If the candidate region is not found (*TargetRegionFound()*, line 8, is *false*), it is attributed to slippage and the robot turns in place, searching for the candidate region. The world model and heuristic constraints resolve any conflicts that arise. Once such a region is found, the robot stops, with the region at the center of its visual field. Then the robot proceeds to learns the color (*LearnGaussParams()* - line 9). Each pixel in candidate region is accepted as a member of the color class being learnt if it is sufficiently distant from the means of the other known color classes. The *mean* and *variance* of the accepted pixels define the color's 3D Gaussian. The learnt Gaussians are used to generate the $128 \times 128 \times 128$ color map (*UpdateColorMap()* - line 11) around once every five seconds. The updated color map, in addition to being used to segment subsequent images and validate the color parameters currently learnt (lines 12-14), helps the robot *localize* itself and move to suitable locations to learn the other colors. The learning algorithm *bootstraps*, with the knowledge available at any given instant being exploited to plan and execute the subsequent tasks efficiently.

If the robot has rotated in place for more than a threshold angle (Ang_{th}) and/or it has spent more than a threshold amount of time learning a particular color ($Time[Color]$), the robot transitions to the next color in the list. The process continues until the robot has attempted to learn all the colors.

Note that instead of providing a color map and/or the motion sequence each time the environment or the illumination conditions change, we just provide the positions of various objects in the robot's world and have it plan its motion sequence autonomously. This significantly reduces the amount of manual input required in our color learning approach [9]) while still learning colors much faster than the baseline approach of hand-labeling several images.

5 Experimental Results

Our previous work [9] demonstrated the ability of the robot to learn the colors when provided with an appropriate action sequence. Here, we show that the robot can succeed at this task while planning its owb action sequence.

To localize, the robot has to learn five colors - *white, green, yellow, blue, pink*, and we measure both its segmentation accuracy and localization accuracy (the robot uses this color map to move to a few positions on the field).

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

Table 1. Successful Planning and Localization Accuracy

Config	Success (%)	Localization Error		
		X (cm)	Y (cm)	θ (deg)
Worst	70	17	20	20
Best	100	3	5	0
avg	90.0 ± 10.7	8.6 ± 3.7	13.1 ± 5.3	9.0 ± 7.7

In Table 1 we tabulate the performance of the robot in its planning task over these configurations. It also shows the localization accuracy of the robot using the learnt color map. The results in the table indicate the performance of the robot over 15 configurations, with 10 trials for each configuration. The robot is able to plan its color learning task and execute it successfully in most of the configurations (that were designed to be adversarial) and the localization accuracy is comparable to that obtained with the hand-labeled color map ($\approx 6cm, 8cm, 4deg$ in X, Y, and θ).

One configuration where the robot performs worst is shown in Figure 2. Here, the robot is forced to move a large distance to obtain its first color-learning opportunity (from position 1 to position 2). This motion sometimes leads the robot into positions that are quite far away from its target location (position 2) and it is then unable to find any candidate image region that satisfies the constraints for the yellow goal. Currently, failure in this initial stage strands the robot without any method for recovery: a suitable recovery mechanism using additional geometric constraints is an important area for future work. Note that the 30% failure rate in this case is entirely due to the unreliability of the robot's motion model: the color-learning plan generated by the robot is quite reasonable.

To test the segmentation accuracy of the learnt color map, we generated a color map by hand-labeling images [6]. We refer to this color map as *HLabel*. We compared the labeling provided by the two color maps (*HLabel* and *Learnt*) with the that provided by a human observer, the *Ground Truth* (*GTruth*). Only the colors of the objects on the field and/or below the horizon matter because other regions are automatically rejected in the object recognition phase. Also, the *correct* classification result is unknown for several background pixels in the image. So, the observer only labeled pixels that appear on or around the field and they were compared with the classification

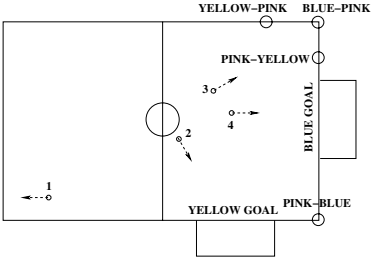

Fig. 2. Sample Configuration where robot performs worst

provided by the two color maps. On average, $\approx$ 6000 of the 33280 pixels in each image get labeled by the observer. The average classification accuracies for *HLabel* and *Learnt*, when compared with *GTruth*, are 99% and 96.7% respectively. We then tested the algorithm under different illumination conditions in addition to testing the algorithm's independence to color labels (labeling all *pink* objects as *blue* and vice versa does not pose any problems). This confirms our hypothesis that a *repainting* of the environment in any way, from just changing color shades, to scrambling colors entirely, does not disrupt our approach. Sample results for these experiments are available on-line: *www.cs.utexas.edu/users/AustinVilla/?p=research/auto_vis*.

6 Conclusions and Future Work

We have presented an approach that automatically plans a motion sequence to learn the desired colors on-board a legged robot with limited computational and storage resources. The corresponding segmentation and localization accuracies comparable to that obtained by the previous approach of having the robot learn the color map by executing a prespecified motion sequence [9]. The robot is able to *plan* its motion sequence dynamically in different world configurations based on heuristic constraints. The planned color learning can be repeated under different illumination conditions and object configurations, exploiting the inherent *structure* in the environment.

Our approach may apply to much more general environments, such as robots in homes or industrial settings. All that's needed is an environmental model, with the locations of distinctive features labeled. A major premise of this research is that generating such a model is significantly easier for a human than labeling pixels or generating a good motion path for color learning. This is reasonable, for example, whenever the configuration of objects in the world changes less frequently than the lighting conditions.

Currently, the color map is learnt from a known starting position without any prior knowledge of colors. We are working on learning colors from an *unknown* starting position on the field. Ultimately, we aim to develop efficient algorithms for a mobile robot to function autonomously under completely uncontrolled natural lighting conditions.

Acknowledgments

We would like to thank the members of the UT Austin Villa team.This work was supported in part by NSF CAREER award IIS-0237699, ONR YIP award N00014-04-1-0545, and DARPA grant HR0011-04-1-0035.

References

1. Cameron, D., Barnes, N.: Knowledge-based autonomous dynamic color calibration. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, Springer, Heidelberg (2004)

2. Sony Legged League Team CM-Pack: In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, Springer, Heidelberg (2002)
3. Cohen, D., Ooi, Y.H., Vernaza, P., Lee, D.D.: RoboCup-2003: The Seventh RoboCup Competitions and Conferences. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, Springer, Heidelberg (2004)
4. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. PAMI 24(5), 603–619 (2002)
5. Efron, B., Tibshirani, R.J.: An Introduction to Bootstrap. Chapman and Hall Publishers, Sydney (1993)
6. Stone, P., et al.: UT Austin Villa 2004: Coming of Age, AI TR 04-313. Technical report, Department of Computer Sciences, UT-Austin (October 2004)
7. Jungel, M.: Using layered color precision for a self-calibrating vision system. In: The Eighth International RoboCup Symposium, Lisbon, Portugal (2004)
8. Schulz, D., Fox, D.: Bayesian color estimation for adaptive vision-based robot localization. In: IROS (2004)
9. Sridharan, M., Stone, P.: Autonomous color learning on a mobile robot. In: The Twentieth National Conference on Artificial Intelligence (AAAI) (2005)
10. Sumengen, B., Manjunath, B.S., Kenney, C.: Image segmentation using multi-region stability and edge strength. In: ICIP (2003)
11. Sony Legged League Team UNSW: In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, Springer, Heidelberg (2002)