In Itsuki Noda, Adam Jacoff, Ansgar Bredenfeld and Yasutake Takahashi, editors, *RoboCup 2005*, Springer Verlag, Berlin, Germany, 2006.

Towards Eliminating Manual Color Calibration at RoboCup

Mohan Sridharan[1] and Peter Stone[2]

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

[2] Department of Computer Sciences, The University of Texas at Austin
pstone@cs.utexas.edu, http://www.cs.utexas.edu/~pstone

Abstract. Color calibration is a time-consuming, and therefore costly requirement for most robot teams at RoboCup. This paper presents an approach for autonomous color learning on-board a mobile robot with limited computational and memory resources. It works without any labeled training data and trains autonomously from a color-coded map of its environment. The process is fully implemented, completely autonomous, and provides high degree of segmentation accuracy. Most importantly, it dramatically reduces the time needed to train a color map in a new environment.

1 Introduction

Upon arrival at RoboCup competitions, one of the first steps for most teams in any of the real robot leagues is *color calibration*: the process of mapping raw camera pixel values to color labels, such as green or orange. 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."

The time required for color calibration contributes in large part to the need for multiple days of setup time before each competition, a costly proposition both from the perspective of reserving the venue and from the perspective of individual travel expenses. In addition, both soccer-playing and rescue robots must eventually be able to operate in natural, changing lighting conditions. Rescue robots in particular must be operational as soon as possible after arriving at a disaster site.

Though events, to date, have all been held under constant, bright lighting conditions, it takes an hour or more to train the robot to recognize the desired colors in its environment. One way to dramatically reduce this time is to enable the robot to autonomously learn the desired colors from the environment using the inherent structure. Doing so may also enable them cope more easily with changing lighting conditions.

In the abstract, automatic color segmentation can be characterized by the following set of inputs, outputs and constraints:

1. *Inputs:*
   - A color-coded map of the robot's *world*. This contains a representation of the size, shape, position, and colors of all objects of interest.
   - A stream of limited-field-of-view images that present a view of the *structured* world with many useful objects, and 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 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.

This paper presents an approach for autonomous color learning on-board a mobile robot with limited computational and memory resources. It works without any labeled training data and trains autonomously from a color-coded map of its environment. The process is fully implemented, completely autonomous, and provides high degree of segmentation accuracy. Most importantly, it dramatically reduces the time needed to train a color map in a new environment.

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 are captured in the *YCbCr* format at $30Hz$ and an image resolution of 208×160 pixels. 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 modality, and images possess common defects such as noise and distortion. Figure 1 shows a picture of the robot and the $4.4m \times 2.9m$ playing field.

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

On the robot, visual processing typically occurs in two stages: color segmentation and object recognition ([3] presents our implementation). Color segmentation is a well-researched field in computer vision with several good algorithms, for example [2, 10]. But these involve computation that is infeasible to perform on autonomous robots given the computational and memory constraints. In the RoboCup domain too, several methods have been applied, from the baseline approach of creating mappings from the YCbCr values to the color labels [**?**], to the use of decision trees [11] and axis-parallel rectangles in the color space [12]. All of them involve an elaborate training process wherein the color map is generated by hand-labeling several ($\approx 20 - 30$) images over a period of at least an hour.

The color map is used to segment the image pixels to one of the desired colors and construct connected constant-colored blobs. The blobs are used to detect useful objects (e.g. markers and the ball). The robot uses the markers to localize itself on the field and coordinates with its team members 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 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 have rarely been successful. One such instance is [6]), wherein the author presents a method to learn the color map using three layers of color maps with increasing precision levels. But the generated map is reported to be not as accurate as the hand-labeled one and other domain specific constraints are introduced to disambiguate between object colors, during the object recognition phase. In [**?**], colors are estimated using a hierarchical Bayesian model with *Gaussian* priors and a joint posterior on robot position and environmental illumination.

This paper presents a novel approach that enables the robot to autonomously learn the entire color map, using the inherent structure of the environment and about seven images, in less than five minutes. It involves very little storage and the resultant segmentation accuracy is comparable to that obtained by the color map generated by the hand-labeling process.

3 Problem Specification

Here, we formally describe the problem of generating a color map for the robot.

To be able to recognize objects and operate in a color-coded world, a robot generally 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 [4] under consideration:

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

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

We represent colors using a Three-Dimensional (3D) Gaussian model (reasonably approximates actual distributions) with the assumption of mutually independent color channels. In practice, the independence assumption, which implies a lack of correlation among the three color channel values for any given color, does not hold for all colors. Nonetheless, it closely approximates reality and greatly simplifies the calculations — computationally expensive operations such as inverting a covariance matrix need not be performed.

Each color $\omega \in [0, N-1]$ can then represented by the density distribution:

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

where, $c_i \in [C_{i_{min}}, C_{i_{max}}]$ 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.

Under this model, the means and variances of the distributions are the only statistics that need to be collected and stored for each color that is to be learnt, making the learning process fast and feasible to execute on the robot. Next, we describe the learning setup and the actual process that the robot goes through to learn the color map.

4 Learning Setup

In this section we describe the algorithm (summarized in Algorithm 1) that the robot executes to autonomously learn the color distributions.

Algorithm 1 General Color Learning

Require: Starting Pose Known, Map of the robot's world.
Require: *Empty* color map.
Require: Array of poses for learning colors, $Pose[]$.
Require: Array of objects, described as shapes, from which the colors need to be learnt, $Objects[]$.
Require: Ability to move to a target pose.

```
1: i = 0, N = MaxColors
2: Time_st = CurrTime
3: while i < N and CurrTime − Time_st ≤ Time_max do
4:    Motion = RequiredMotion( Pose[i] )
5:    Perform Motion {Monitored using visual input}
6:    if LearnGaussParams( Colors[i] ) then
7:       Learn Mean and Variance of color from candidate image pixels
8:       UpdateColorMap()
9:       if !Valid( Colors[i] ) then
10:         RemoveFromMap( Colors[i] )
11:      end if
12:   end if
13:   i = i + 1
14: end while
15: Write out the color statistics and the color map.
```

The algorithm can be described as follows: The robot starts off at a known position in its map of its world. It has no initial color information, i.e. the means and variances of the colors to be learnt are initialized to zero. It also has three lists: the list of colors to be learnt (*Colors*), a list of corresponding positions that are appropriate to learn those colors (*Pose*), and a list of corresponding objects, defined as shapes, that can be used to learn the colors. Using a navigation function (RequiredMotion()), the robot determines the motion required, if any, to place it in a position corresponding to the first entry in *Pose*, and executes the motion command. The object shape definition – the corresponding entry in the *Objects* array – leads to a set of constraints (heuristic *candidacy tests*) that are used to select the candidate blob. The robot stops when either a suitable blob is found or it thinks it has reached its target position. Further details of the *candidacy tests* can be found in a technical report [3].

Once in position, the robot executes the function *LearnGaussParams()* to learn the color. If a suitable candidate blob of *unknown* color (*black* in our case) exists, each pixel of the blob is examined. If the pixel value is sufficiently distant from the *means* of the other known color distributions, it is considered to be a member of the color class under consideration. When the entire blob has been

analyzed, these pixels are used to arrive at a *mean* and a *variance* that then represent the *3D Gaussian density function* of the color being learnt.

The function *UpdateColorMap()* takes all the learned Gaussians as input and generates the complete mapping from pixel values to the color labels. This process of assigning color labels to each cell in the $128 \times 128 \times 128$ cube is the most intensive part of the learning process. Hence, it is performed only once very five seconds or so. Each cell is assigned a color label corresponding to the color whose density function (Equation 2) has the largest *probability* value. The updated color map is used to segment all subsequent images.

The segmented images are used for detecting objects, which are in turn used to validate the colors learnt (*Valid()*). The entire learning procedure is repeated until all desired colors are learnt and/or the predecided learning time ($Time_{max}$) has elapsed. A more detailed description can be found in [9].

5 Experimental Setup

A line drawing of the legged league field, with its color coded goals and markers, is shown in Figure 2. We present the results when the robot always starts off in *Position-1* and moves through a deterministic sequence of positions (the elements of the array *Pose[]*).

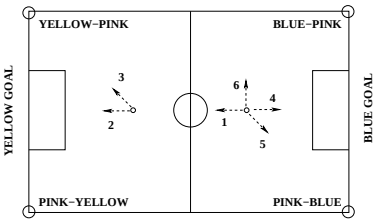

Fig. 2: The Learning positions.

The steps involved in the algorithm can be presented as an ordered list of positions, colors (to be learnt) and objects:

1. Step-1: Position1 with head tilted down, *white* and *green*, Field line and center circle.
2. Step-2: Position-2, *yellow*, Yellow goal.
3. Step-3: Position-3, *pink*, Yellow-pink marker.
4. Step-4: Position-4, *blue*, Blue goal.
5. Step-5: Position-5, *blue* (Disambiguate *green* and *blue*), Pink-blue marker.
6. Step-6: Position-6 with head tilted down, ball color (*orange*), Ball.
7. Step-7: Position-6 with head horizontal, opponent's uniform color, Opponent.

The robot then writes out the color map and the statistics to the memory stick. A few important points are to be noted with regard to the learning process. In *Position-1*, learning is performed based on the fact that a large portion of the image (in that position) consists of *green*. The algorithm is entirely dependent on inherent structure of the environment and not on the particular color that is being learnt. The positions for learning the ball and opponent colors are set so as to minimize the movement. Currently we only learn *red* for the opponent uniform color, though the process could be used to learn *darkblue* too. The video of the learning mechanism, as seen from the robot's camera, can be viewed online [1].

6 Experimental Results

We tested the accuracy of the color maps that were learned autonomously on the robots by comparing their segmentation accuracy with a color map generated by the prevalent approach of hand-segmenting a set of ≈ 25 images. We refer to

this color map as the *Hand Labeled* (*HLabel*) color map. This map corresponds to a fixed illumination condition. Here, an intermediate map (IM) of the same size as the overall color map is maintained for each color. Each cell of an IM stores a count of the number of times an image pixel that maps into that cell was labeled as the corresponding color. Each cell in the final color map is assigned the label corresponding to the color whose IM has the largest count in that cell.

Based on results [5, 7, 8] that the *LAB* color space could be reasonably robust to illumination variations, we trained a color map each in *LAB* and *YCbCr*. Since the colors of the ball and the opponent overlap with the marker colors, we performed the analysis in stages: first with just the fixed marker colors and then with all the colors included.

On a set of sample images of the markers (15) captured using the robot's camera, we first compared the performance of the three color maps with the color labeling provided interactively by a human observer, the *Ground Truth* (*GTruth*). We are interested only in the colors of the markers and other objects on the field and/or below the horizon. Also, the *correct* classification result is unknown (even with *HLabel*) for several background pixels in the image. Therefore, the observer only labels pixels suitable for analysis and these labels are compared with the classification provided by the three color maps. On average, $\approx 20\%$ of the pixels in the image get labeled by the observer. The average classification accuracies are 87.8 ± 3.18, 97.9 ± 0.76, and 98.8 ± 0.44 for the *YCbCr*, *LAB* and *HLabel* color maps respectively, as compared to *GTruth*.

The color labeling obtained by using the *HLabel* color map or the map generated in the *LAB* color space is almost perfect in comparison to the human color labeling. There is not much difference in the qualitative performance between these and the *YCbCr* map. Sample image results are available in [9, 1] – the robot is able to learn a reasonable color map in both color spaces when only the marker colors are considered.

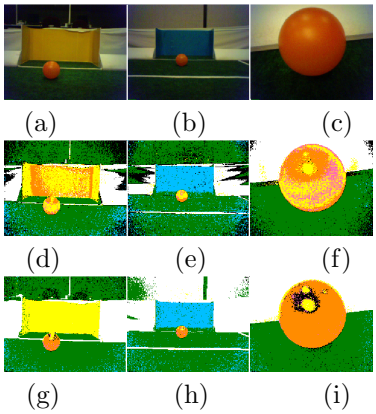

Fig. 3: Sample Images with Ball. (a)-(c) Original, (d)-(f) YCbCr, (g)-(i) LAB

Next, we let the robot learn the ball color (*orange*) in addition to the marker colors. The average classification accuracies are $74.8 \pm 9.2\%$, $94 \pm 5.6\%$ and $98.5 \pm 0.8\%$ for the *YCbCr*, *LAB* and *HLabel* color maps respectively, as compared to *GTruth*. Figure 3 show the segmentation results over a set of images.

In the *YCbCr* color space, the inclusion of *orange* causes the segmentation to degrade even over the colors (*pink* and *yellow*) that it could classify well before. This is not the case in *LAB* – the object recognition procedure is able to find the ball without any additional constraints (the ball is rarely found in *YCbCr*).

Therefore the color of the opponent's uniform (*red*) is learnt only in the LAB color space. Images illustrating this can be seen at [9, 1].

While operating in the LAB color space, we still do not want to transform each pixel in the test image from $YCbCr$ to LAB due to computational constraints. So, during the color map update, we assign the color label to each discrete cell in the $YCbCr$ color map by determining the label assigned to the corresponding pixel values in LAB. The pixel-level transformation increases in the training time. The learning process takes $\approx 2.5 minutes$ in $YCbCr$ while it takes $\approx 4.5 minutes$ in LAB, still much smaller than the time taken to generate *HLabel*, an hour or more.

When the illumination changes within a range of illuminations, the original color map does not perform well. But the robot is able to learn a new color map in a few minutes.

Finally, we tested the hypothesis that the algorithm is robust to color *remapping*. We changed the field setting by moving a goal to a carpet that has a non-uniform blue design and we placed a small piece of white paper on it instead of the field lines. The robot still learnt the carpet color as *green* and proceeded to learn other colors. Next, we started the learning process with the robot in *Position-2*, facing the blue goal (Figure 2). The robot ended up learning the color *blue* as *yellow* and vice versa. This confirms our hypothesis that the process is dependent only on shape and size and not on the particular color that is being learnt.

Sample image results for all experiments can be seen in [9] or on the team web-site [1].

7 Discussion and Conclusion

We have presented an approach to automating the color learning and segmentation process on-board a legged robot with limited computational and storage resources. In spite of the relatively low-resolution images with inherent noise and distortion, the algorithm enables the robot to autonomously generate its color map in a very short period of time. The corresponding segmentation accuracy is comparable to the that obtained by hand-labeling several images over a period of an hour or more. This could result in a substantial reduction in the setup time before the games can begin at RoboCup competitions.

Though we have tested our approach only in the legged league environment, it applies to the other leagues where the vision is done on-board a mobile robot in a known, color-coded environment. Color-calibration in the small-size league is currently more straightforward because vision is often done with a stationary overhead camera. However, as teams move towards on-board vision, they will face the same constraints as the other robot soccer leagues. To apply this method in the rescue league requires the generation of a test-environment with objects of relevant colors in known locations. One could imagine quickly collecting relevant training objects and placing them in fixed locations that can be communicated to the robot for training purposes. However it remains to be shown that doing so generalizes to the larger environment, and if so, that it enables a reduction in manual effort and training time. This is an important area for future research.

The algorithm depends only on the structure inherent in the environment and a *re-mapping* of the colors does not prevent the robot from learning them. Further, the color map can be learnt in several fixed illumination conditions between a minimum and maximum on the field. The learning can be easily repeated if a substantial variation in illumination is noticed.

Currently, the color map is learnt from a known fixed starting position without any prior knowledge of colors. An extension that we are currently working on is to learn from any given starting position on the field.

Acknowledgments

We would like to thank the members of the UT Austin Villa team for their efforts in developing the soccer-playing software mentioned in this paper. 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. The Utaustinvilla research website, 2004. `http://www.cs.utexas.edu/users/AustinVilla/?p=research/auto\_vis`.
2. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
3. P. Stone et al. UT Austin Villa 2004: Coming of Age, AI Technical Report 04-313. Technical report, Department of Computer Sciences, University of Texas at Austin, October 2004.
4. Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2002.
5. Jeff Hyams, Mark W. Powell, and Robin R. Murphy. Cooperative navigation of micro-rovers using color segmentation. *In Journal of Autonomous Robots*, 9(1):7–16, 2000.
6. Matthias Jungel. Using layered color precision for a self-calibrating vision system. In *The Eighth International RoboCup Symposium*, Lisbon, Portugal, 2004.
7. B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire. Low-order-complexity vision-based docking. *IEEE Transactions on Robotics and Automation*, 17(6):922–930, 2001.
8. M. Sridharan and P. Stone. Towards illumination invariance in the legged league. In *The Eighth International RoboCup Symposium*, Lisbon, Portugal, 2004.
9. M. Sridharan and P. Stone. Autonomous color learning on a mobile robot. In *The Twentieth National Conference on Artificial Intelligence (AAAI)*, 2005, To Appear.
10. 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.
11. The UNSW Robocup 2001 Sony Legged League Team. *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002.
12. The UPennalizers Robocup 2003 Sony Legged League Team. *RoboCup-2003: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2004.
13. William Uther, Scott Lenser, James Bruce, Martin Hock, and Manuela Veloso. Cm-pack'01: Fast legged robot walking, robust localization, and team behaviors. In *The Fifth International RoboCup Symposium*, Seattle, USA, 2001.