Real-Time, Adaptive Color-based Robot Vision[1]

Brett Browning and Manuela Veloso
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA, USA, 15213
{brettb,mmv}@cs.cmu.edu

***Abstract* – With the wide availability, high information content, and suitability for human environments of low-cost color cameras, machine vision is an appealing sensor for many robot platforms. For researchers interested in autonomous robot teams operating in highly dynamic environments performing complex tasks, such as robot soccer, fast color-based object recognition is very desirable. Indeed, there are a number of existing algorithms that have been developed within the community to achieve this goal. Many of these algorithms, however, do not adapt for variation in lighting intensity, thereby limiting their use to statically and uniformly lit indoor environments. In this paper, we present a new technique for color object recognition that can adapt to changes in illumination but remains computationally efficient. We present empirical results demonstrating the performance of our technique for both indoor and outdoor environments on a robot platform performing tasks drawn from the robot soccer domain. Additionally, we compare the computational speed of our new approach against CMVision, a fast open-source color segmentation library. Our performance results show that our technique is able to adapt to lighting variations without requiring significant additional CPU resources.**

Index Terms* – *Robot vision, adaptation, object recognition.

I. Introduction

For many robot domains, vision provides the ideal sensor for robots due to its low cost, wide availability, high data content and information rate. For highly dynamic adversarial tasks, such as robot soccer, being able to extract significant information about the world is crucial to operating effectively, making vision an appealing sensor. Additionally, the presence of adversaries mean that it is essential to process the sensory information in minimum time. Consequently, for vision-based autonomous robots in dynamic domains it is crucial that vision processing algorithms are fast in addition to being robust.

Color-based object recognition, where objects of interest are colored in a uniquely identifiable and known way, is a technique that has found wide use in the robotics community. Correspondingly, there are now a number of fast color segmentation vision libraries that are available such as CMVision [3,4]. However, to date, the available algorithms are unable to adapt to lighting variations. Clearly the ability to adapt is a key factor to improving robustness and is crucial in outdoor environments where lighting variation is unavoidable.

In our work with the Segway RMP platform (see figure 1 and [9]), we have been exploring techniques to enable a vision-centric robot to be able to play soccer in outdoor environments where illuminat is variable [5]. The Segway robots are able to reach speeds of 3.5m/s and can kick the ball at just under 4m/s. As the task is adversarial and highly dynamic, the cobmination of robot speed and ball speed means that it is essential that the vision algorithms be both robust, and extremely efficient. Indeed, only a fraction of the CPU resources can be devoted to vision processing as the remainder of the CPU resources must be used for cognition in order to get low-latency robot behaviors.

We have developed a new technique for fast color-object recognition that is suitable for use in robot platforms like the Segway. The key contribution of our new approach is that it is able to adapt its segmentation process to different lighting conditions, within reason. In addition to being adaptive, this new technique requires only moderate additional computational resources beyond existing fast color vision algorithms. In this paper, we present our new technique and provide empirical evidence of its performance using Segway RMP robot bases. We present these results as a step towards useful, but fast color-based robot vision techniques.

This paper is structured in the following way. In the ensuing section we motivate the problem and describe relevant algorithms that have been established in the literature. In section III, we present our new algorithm followed by its performance results drawn from experiments with our Segway RMP robot soccer platforms in section IV. Section V concludes the paper.

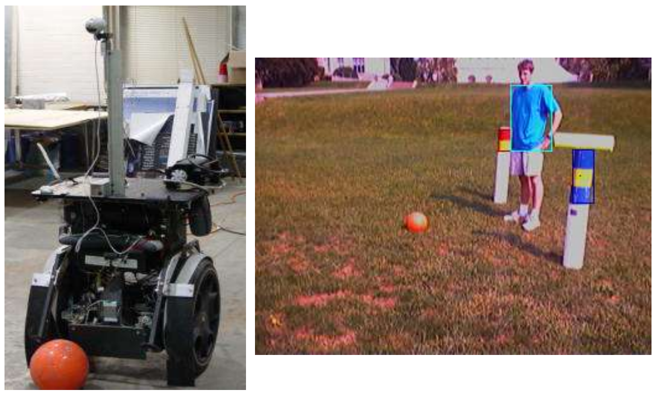
Figure 1. The segway robot, and a typical view from the camera located on a pan-tilt unit on top of the robot.

II. Problem Description

In this paper, we focus on the problem of robot vision for a robot operating in a highly dynamic world filled with a small set of fast moving, but visually distinctive objects. Each object is color coded from a finite set of distinctive colors in a unique way. Concretely, the objects are the ball,

[1] This work was supported by United States Army under Grant No. DABT63-99-1-0013. The content of the information in this publication does not necessarily reflect the position or policy of the Defense Advanced Research Projects Agency (DARPA), US Army, US Government, and no official endorsement should be inferred.

the field surface, field markers, two types of goal markers, teammates, opponent robots and opponent humans. The field is a mostly uniform colored surface of either grass, concrete, carpet or astro turf. The ball is orange, while the field markers are white, with a blue-yellow-blue band at the top. The goal markers are white with a yellow-red band at the top, where one pair has yellow on top while the other has red on top. The teammates and opponents wear colored shirts (usually cyan, red, green, or yellow), where the color is different for each team.

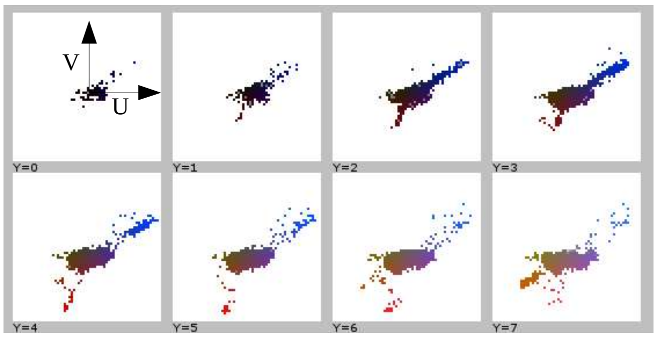

Figure 2. An example histogram from an outdoor image similar to figure 11. Each square is a slice of the histogram along the UV plane with constant Y value. The Y value increases across each row, and down the columns.

The goal for robot perception is, therefore, to be able to quickly and robustly identify each object using a monocular color camera and estimate its range and bearing. Given the dynamic nature of the task, and that the robot is fully autonomous, identification must happen as quickly as possible meaning all vision algorithms must be extremely parsimonious of computational resources.

While there has been considerable research into general purpose object recognition, such as [10,11,12], these algorithms are invariably not fast enough for use in this setting where only a fraction of the CPU resources are available for vision processing. In contrast, there are a number of techniques that have been developed that provide very fast color blob tracking from which object recognition algorithms for simple, pre-defined objects can be developed. Examples include CMVision [3,4], the Newton Labs Cognachrome [2], and various region growing techniques [1, 6]. Such techniques have found use in robot research, particularly in robot soccer, as well as commercial applications [5].

The key to the speed of these approaches is the use of lookup tables to classify each pixel according to which symbolic color class it appears to belong to. While these techniques are extremely fast, the use of a lookup table for classification is fragile to changes in illumination. For an indoor environment, this is less of a limitation than that it seems. For an outdoor environment, this is a severe limitation as variable lighting is a fact of life even in open fields. In addition to illumination variations, there is the additional problem of non-diffuse lighting causing specular reflections, shadows, and large variations in color from the sun-side to the shade-side.

Although outdoor environments are very challenging, we believe that for the constrained environment of an open soccer field with color coded objects vision is still a tractable problem. The motivation for our belief stems from the observation that although colors can vary, they do so in a constrained way given the restrictions above. In particular, as the general illumination level increases or decreases, a colored pixel prescribes a constrained trajectory in color space. Moreover, nearby colors move in a continuous fashion so that the color space does not fold at any point. In other words, the color red is always more 'red' than say any green surface.

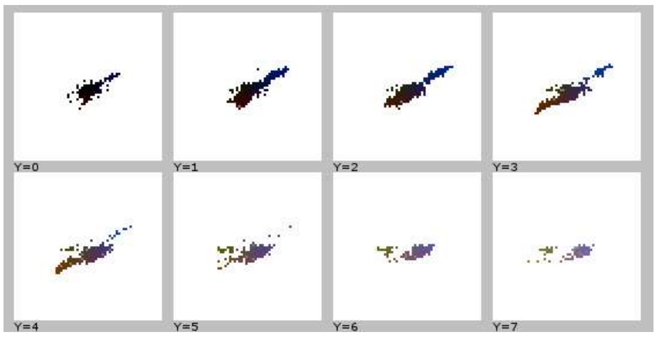

Figure 3. Another histogram drawn from a similar scene on a different day with lots of cloud cover. Note there is no red object in this image.

For this discussion, let us focus on the YUV color space. Figure 2 shows an example of part of the YUV histogram for an outdoor scene much like that shown in figure 6. The ball is a orange object, which contributes to the collection of dots in the bottom half of the UV plane, blue is up towards the right hand corner, gray colors are in the middle. Figure 3 shows a different histogram drawn from a similar scene but under differing lighting conditions. Note that the two histograms are different, however, much of the structure remains. The blue objects in the image occupy the upper right hand side in the same way. The red objects in the image (lower left hand streak) moves in towards the center. A hard labelling scheme such as a lookup table, fails to capture such changes, however. We propose that a soft-labeling scheme with an adaptive threshold is the key to overcoming such dynamic changes, as discussed in the next section.

III. An Adaptive Segmentation Approach

Our segmentation technique is motivated by the observation that for most of the domains of interest here changes in illumination lead to small changes in color value and that these changes are relatively uniform across all colors. In other words, with modern cameras with automatic shutters and gain control red pixels may vary in color but will stay in the same region of color space. Therefore, we propose that if pixel classificiation thresholds are able to adapt by small amounts, it should become possible to robustly classify pixel colors across moderate changes in illumination. Our goal is to achieve such a robust, adaptive system but without significantly increasing computational requirements.

```
Vision Algorithm(image):
    segmentImage(image)
    buildHistograms()
    adaptThresholds()
    findObjects()
```

Figure 4. The main algorithm.

Figure 4 shows the main components of the approach. The key idea is to use a soft labeling of pixel class, followed by a hard decision using an adaptive threshold.

The combination of soft-labeling and adaptive thresholding provides the plasticity for lighting variation. Following this, connected pixels can be conglomerated using a connected component analysis. Objects are detected and recognized by searching for nearby regions that match apriori models, with soft-comparisons to account for variations in shape, size, and missing features. We now describe each of these components in greater detail.

A. Pixel Classification

To label pixels according to which symbolic class they belong to, we use a soft-labeling scheme followed by a hard decision based on adaptive thresholds. Essentially, we examine each, *p*, pixel in the image and estimate the likelihood, $P(p \in C_j)$ of it belonging to a color class, *j*. The pixel is assigned to the highest priority color class for which its likelihood is above the threshold for that color class, θ_j. The priority for each color class, defined as $Pr(j)$ is defined a priori and is fixed. Thus, we have:

```
Segment:
    for each p in image
        for each class j
            if  $P(p \in C_j) > \theta_j$
                $S = S \cup \{j\}$
        $s = \max_{Pr(j)}(S)$
```

Figure 5. The segmentation algorithm.

Figure 6 shows a typical outdoor image, the raw labeling for 'red' and the resulting segmentation. There is the question of where the likelihoods for each color class comes from. In the work described in this paper, the likelihood mappings are defined apriori by generating a lookup table for the mapping from a hand-calibrated table using a GUI tool, as described in section IV.

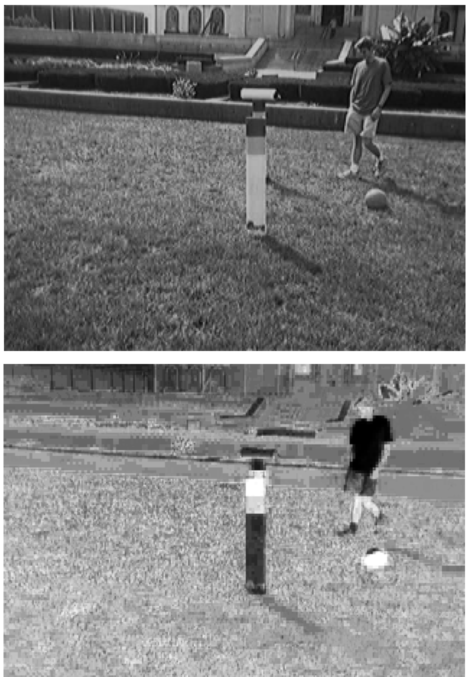
Figure 6. The top figure shows a raw image, while the bottom image shows the corresponding probability value for each pixel for the red color.

B. Threshold Adaptation

A histogram based approach is used to adapt the threshold from frame to frame. Following the use of histogram techniques for monochrome images [7], the key assumption here is that pixels in the image are drawn from two different underlying distributions: pixels that belong to the color class of interest and pixels that do not. Thus, the key assumption translates to a histogram of likelihood values consisting of two, clearly distinguishable Gaussian peaks centered around likehood values of 1, and 0, respectively.

While the presence of conflicting colors, and lighting variations, will introduce additional peaks and complicate the histogram, the basic assumption should hold. Therefore, our approach is as follows. First, the histogram is smoothed by convolution with a zero-mean Gaussian kernel operator, where the width of the kernel is chosen a priori and is a parameter to the system. With a suitably chosen kernel, this smoothing operation produces a histogram with a small number of clearly distinguishable peaks and valleys. It is a relatively straightfoward exercise to search for each stationary point in the smoothed histogram and label it as an inflection, peak, or trough. With a sufficiently large Gaussian kernel, the smoothing operation ensures that peaks and valleys are clearly distinguishable.

The peak with the highest likelihood value corresponds to the pixels of interest. Thus, the target threshold is selected corresponding to the trough that fully captures the peak and both its sides. If the target threshold is above a minimum value, the actual threshold is stepped part way towards this target threshold. The minimum value acts to rule out setting bad thresholds when there is no color of interest in the image. The partial step acts as a first-order filter to smooth out rapid changes in threshold value. Figure 7 shows the adaptation algorithm. This algorithm is executed once per frame for each color class. Figure 10 shows a typical histogram after smoothing.

```
Adapt Threshold:
    for each class j
        h'=conv(h,gauss_j)
        sp = stationaryPoints();
        dt = arg max (sp.trough)
        if (dt > minv)
            t' = alpha * (dt-t) + t;
```

Figure 7. The histogram-based threshold adaptation algorithm.

C. Region Extraction

Once the image has been segmented, regions of similarly labelled pixels are found using connected component analysis. For this, we make use of the publicly available Open Source software CMVision [2,3]. CMVision provides fast connected components using a combination of run length encoding and run conglomeration. Given this is a standard algorithm, we will not discuss it further other than to note that it produces a list of regions for each color class, sorted by number of pixels. Additionally, simple region statistics of centroid, and bounding box are calculated.

D. Object Recognition

Once an image has been segmented and colored regions extracted, high-level vision must attempt to detect and recognize relevant objects in the image if any are present.

Given that regions are the features available for object recognition, all objects will consist of one or more colored regions. Each of these regions is constrained by geometry and can be recognized to some degree by its size, shape, average color, and so on. The relative position between regions on an object are also constrianed by geometry. Therefore, our approach to object recognition is to exploit these constraints in an efficient manner.

Our algorithm for recognizing objects is drawn initially from [8]. The algorithm works by first filtering out unrecognized regions based on a probablistic template match over a set of features calculated from the region properties. Concretely, we define a set of features for each different region, where the features are functions of the region properties of area, bounding box width and height, density within the bounding box, and expected size and area given its position in the image or relation to neighbor regions. We model each feature with a Gaussian uncertainty model with hand-calibrated mean and variance. Lastly, the uncertainty or noise in each feature is assumed to be independent. Based on these assumptions, the likelihood of a region being one of interest is given by:

$$P(r \in object_k) = \prod_{\mu} \frac{1}{\sqrt{2\pi\sigma_{k\mu}}} e^{\frac{-1}{2}\|x - x_{k\mu}\|^2 / \sigma_{k\mu}^2}$$

Regions with too low a likelihood value are rejected. The threshold for the combination of these likelihoods is set a priori. Table 1 lists the features we have made use of.

Feature	***Description***
Shape	Ratio of width to height
Area	Number of pixels in region
Density	Ratio of pixels to bounding box area
Expected width/height	Comparison between observed width/height and expected width/height given position in image and camera geometry
PCA	Principle Components of pixels using SVD to determine long axis/short axis

Table 1. The features used to filter regions.

Once each region is filtered, the graph of regions that make up the object are matched against a set of templates describing the object in question, again using a probablistic matching technique. Here knowledge of the expected graph for an object is used to limit the search space for comparing regions. We use the same approach of assuming that features can be matched probabilistically, and have a Gaussian uncertainty distribution. We also assume that each feature is independent, allowing a multiplicative approach to work as above. The features used to evaluate the match are calculated from region statistics across two or more regions in the connected graph. The features describe either relative bounding box dimensions, relative locations of the regions compared to some expectation, and relative area of two regions. Lastly, a sanity check is applied based on size in image against expected size given the calculated location. Table 2 lists the features we have made use of.

Feature	***Description***
Histogram	Local histogram of classified colors to meet specified constraints
Distance	Distance between regions
Direction	Direction vector between regions
Dimensional	Comparison in area, width, height between regions

Table 2. Features for graph recognition. Note the histogram feature is boolean and acts as a gate on whether the regions are acceptable.

The location of an object relative to the robot, one of the major outputs for vision, is calculated using one of two approaches. If the object has a known height or width, a pin-hole camera model combined with the knowledge of the camera geometry relative to the ground surface is used. If such informaiton is unavailable, the distance is estimated by projecting a ray and intersecting it with a known fixed plane parallel to the ground surface at some offset *z*. The ray is projected through the known part of the object that intersects the *z*-plane using a pin-hole camera model and the known camera geometry.

IV. Implementation Details

For the task required, robot vision, having a good algorithm is not sufficient to produce a good working system. Robots operating in highly dynamic environments require minimal delay in responding to changes in the world. For vision processing, this translates into maintaining a high frame rate and ensuring that the time to process a single image is as rapid as possible. Hence, algorithms must be encoded efficiently. In this section, we present the implementation details for our specific system.

A. Vision Implementation

The pixel classification step is the most time intensive part of the algorithm. Even for a 320x240 image, 72,000 pixels must be processed just to classify the frame. Thus, the key step is to convert the probability distribution of each color belonging to a class into a lookup table. That is, to compute the probability assignment quickly a lookup table approximating $P(p \in C_j)$ is generated for each color class.

For the application discussed in this paper, images are available in YUV 4:2:0 planar format at 320x240 resolution. Each pixel consists of a vector $p_i = (y, u, v)^T,\ y, u, v \in [0, 255]$ where the vector elements can take on any value between 0 and 255. To limit the size of the lookup table, to prevent overloading the cache, each lookup table is stored as a 16KB, 3 dimensional array with 6 bits for each of U and V, and 4 bits for Y.

The histograms for each color class are recalculated while processing the image. To maintain speed, threshold adaptation is a pipelined process whereby thresholds from the current frame are selected for use in the the following frame. There is a key assumption here that sequential frames are similar in appearance and lighting levels. While not strictly true, it is a reasonable assumption. As described earlier, CMVision [3,4] is used for the connected component analysis to extract regions from the image.

Object detection is coded specifically for each known object in question. The general structure for each object is identical – identify a seed region, attempt to match the graph template from this seed. As described each step consists of calculating features from the region statistics or the relationship between different region statistics. Given the relatively small number of matching regions, these algorithms do not have the same emphasis on utilizing CPU resources efficiently as image classification.

B. Calibration

The last implementation detail concerns the parameters for the system. At the highest level, the graph templates used to identify each object are hand-coded as seperate algorithms. We are currently examining ways to extend this to a more general purpose system, but this remains as future work. Similarly, each of the features that are used to evaluate regions, or region neighbors are hand-coded. The expected value and variance for all of these features are obtained from parameter files, loaded at boot time, or calculated from other region parameters (e.g. expected size of two regions on the one object). Typically, these parameters once set are rarely modified again. However, in some cases these parameters provide a mechanism for tuning the algorithm performance. In practice, these parameters are relatively robust to small-scale change. Hence, the system has only moderate to low sensitivity to these parameters.

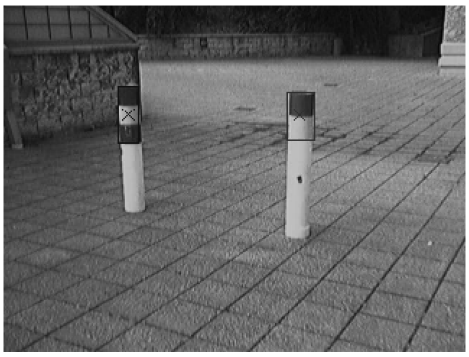

Figure 8 . A typical outdoor scene showing two markers. The recognized markers are shown with an X located at the centroid of the marker bands.

The color probability tables are generated using a GUI-based tool. The tool, called *colortrain*, allows a user to label parts of the color space as belonging with probability 1 to the color class in question. Additionally, the user can sample pixels from the image to help aid this process. Once, the user is satisfied with the labelling, he or she can generate the probability tables, whereby the table is smoothed through convolution with a zero-mean Gaussian kernel in YUV space. The kernel is modelled as three Gaussians, one in each color channel of YUV respectively. The variance for each channel is provided as a parameter to the system, although this is certainly a learnable value from pre-labelled images.

V. Performance Results and Discussion

In this section we examine the performance of the vision system under different lighting regimes.

A. Basic Results

Figure 8 shows an outdoor scene with the output of the vision algorithm is shown, while the final segmentation is shown in figure 9. The identification of both markers is shown by the X and bounding box for the color bands on the markers. For the red-yellow marker (the right one), the X should appear at the intersection of the two colored bands. For the blue-yellow-blue marker, the X should appear in the middle of the yellow marker. The distance to these markers is estimated from the vertical seperation between the two furthest colored regions.

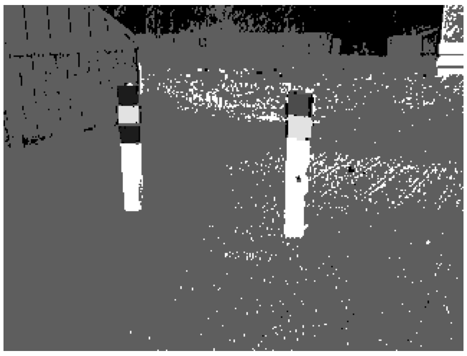

Figure 9. The segmentation output for figure 8.

Clearly, the algorithm is able to segment the image reliably if calibrated even though part of the left hand side of the image is in the shade. The speckles of white occur because the ground is not a uniform color, however these regions are all quite small and therefore are easily filtered. In outdoor scenes, such speckles are common. They impact upon the running time of the CMVision algorithm marginally, but not to any significant degree. We have experimented with morphological processes of eroding and dilating, although this also affects the desirable regions and is therefore of questionable value.

Figure 10 shows the histogram generated for the 'blue' color after smoothing. The various stationary points after smoothing are clear in the image. The peak on the right hand side corresponds to the group of pixels that we wish to be labelled as of that color class. This is the peak that the algorithm searches for by finding the trough to the left of this peak.

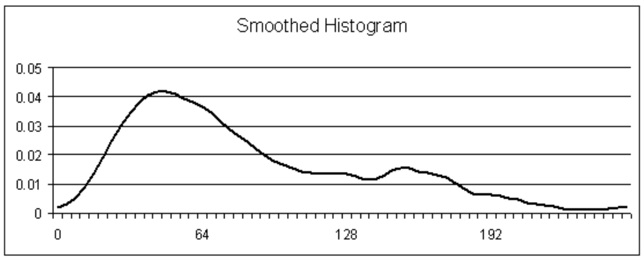

Figure 10. The blue histogram after smoothing. The first peak on the right-hand side which is difficult to see corresponds to the desired peak threshold.

B. Adaptation Results

To examine adaptation we use two approaches. First, we examine how well a calibration carries from one outdoor environment to another. Using the calibration for the above image sequences, we ran the vision system on an image sequence recorded on a different day, with different lighting. Rather than being a cloudy day, as in the first images, the image below is drawn from a sequence on a bright sunny day with small patches of clouds. Figure 11 and 12 show an image from the second test set. Note that the ground is grass instead of cement, hence it is a different color and matches poorly. In contrast, the red, yellow, white, and blue colors adapted successfully to the different lighting conditions without any recalibration required.

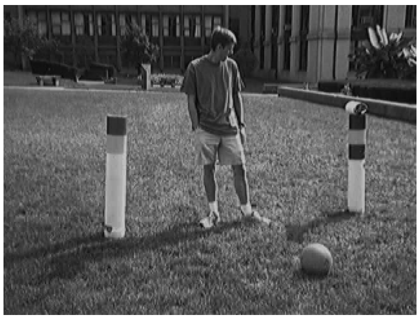

Figure 11. A second test-set movie sequence under outdoor lighting conditions on a different day. Note, the ground is a different surface.

Figure 13 shows a graph of the threshold values for this second sequence. Here the robot drove around in a circle with the focus of the circle on the human. As it was sunny, the thresholds adapted to the changes that occurred as the robot moved from the shadow side, to the sunny side. Additionally, the presence of a cloud caused significant changes in the camera gain settings towards the end of the sequence. The algorithm was able to adapt and continue to recognize the objects despite the lighting variation.

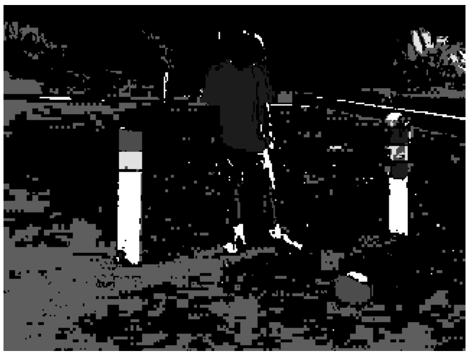

Figure 12. The classification performance corresponding to Figure 11, but with the calibrations developed for the image sequence shown in figure 8.

C. Benchmarking

Table 3 provides the performance benchmark comparison between the new algorithm with adaptation and segmentation with a CMVision lookup table. The results are shown for an outdoor movie sequence, with both systems calibrated, running on a 1.2GHz Pentium III processor laptop over approximately 30s. The images are 320x240 YUV 420 planar format and arrive at 30Hz. Clearly, the additional cost of using the soft-labelling technique is on the order of 2ms. This is certainly worth the cost at the benefit of adaptation across lighting changes.

Algorithm Step	***Adaptation mean (stdev.)***	***CMVision mean (stdev.)***
Segmentation	7.37 (0.91)	5.10 (0.60)
Threshold update	0.10 (0.01)	0.07 (0.09)
CMVision	0.97 (0.14)	1.21 (0.11)
High Level Vision	0.22 (0.02)	0.16 (0.08)
Total	**8.66 (0.87)**	**6.55 (0.62)**

Table 3. Shows the processor usage for the new algorithm and CMVision on a 1.2GHz Pentium III laptop machine for an outdoor scene (approximately 2 minutes of data or 3600 frames. All values are in milliseconds.

VII. Conclusions and Future Work

In this paper, we have presented a new technique for fast color vision algorithm for use in robot domains where lighting levels may vary. The key to our approach resides in the soft-labeling of pixels with an adaptive threshold technique. We have fully implemented this algorithm on a Segway RMP platform, and presented results showing its segmentation performance. Additionally, we have compared the computational requirements of the algorithm against CMVision as a benchmark and demonstrated that the additional computaitonal load for the soft-labeling process is sufficiently small to be negligible.

Our future work will focus on extending this approach to provide automated calibration of the probability tables, as well as further generalizing the high-level graph matching technique for use on a wide range of objects.

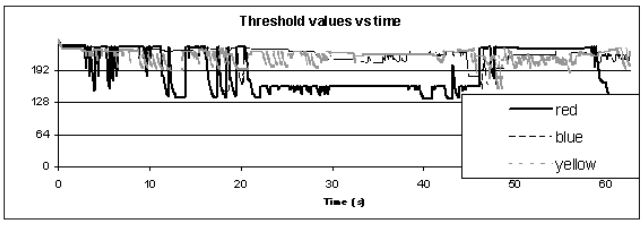

Figure 13. Threshold values varying over time for red, blue, and yellow. Note the red corresponds to a ball which has a rounded surface and therefore a greater spread of colors making it more sensitive to lighitng variation.

References

[1] Hundelshausen, F., Rojas, R. An omnidirectional Vision System that finds and tracks color edges and blobs. *RoboCup-01: Robot Soccer World Cup V*, Springer, 2001.

[2] Newton Labs Cognachrome *http://www.newtonlabs.com/cognachrome/*

[3] Bruce, J., Balch, T. and Veloso, M. Fast and Inexpensive Color Image Segmentation for Interactive Robots. In *Proceedings of IROS-2000*, Japan, October 2000.

[4] CMVision web page *http://www.cs.cmu.edu/~jbruce/cmvision*

[5] Browning, B., Rybski, P., Searock, J., and Veloso, M. Development of a soccer-playing dynamically-balancing mobile robot. In *Proceedings of International Conference on Robotics and Automation (ICRA'04)*, May 2004.

[6] Weigel1, T. and Nebel1, B. KiRo – An Autonomous Table Soccer Player. *RoboCup 2002: Robot Soccer World Cup VI*. Gal A. Kaminka, Pedro U. Lima, Raul Rojas (eds). Lecture Notes in Computer Science, vol. 2752 / 2003, Springer-Verlag Heidelberg, 2003.

[7] Fu, K., Gonzalez, R., Lee, C. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill, NY, 1987.

[8] Lenser, S., Bruce, J., and Veloso, M. CMPack: A Complete Software System for Autonomous Legged Soccer Robots. In *Proceedings of the Fifth International Conference on Autonomous Agents*, May 2001.

[9] Nguyen, H. G., Morrell, J., Mullens, K., Burmeister, A., Miles, S, Thomas, K., and Gage, D. W. Segway Robotic Mobility Platform, *SPIE Mobile Robots XVII*, Philadelphia, PA; 26-28 October, 2004.

[10] Schneiderman, H. Learning a restricted Bayesian network for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2004.

[11] Huber, D., Kapuria, A., Donamukkala, R. R., Hebert, M. Parts-based 3D object classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, June, 2004.

[12] Lowe, D. G. Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision (ICVPR'99)*, 1999.