

Robust Color Segmentation through Adaptive Color Distribution Transformation*

Luca Iocchi

Dipartimento di Informatica e Sistemistica
Università “La Sapienza”, Rome, Italy
Via Salaria 113 00198 Rome Italy
E-mail: iocchi@dis.uniroma1.it

Abstract

Color segmentation is typically the first step of vision processing for a robot operating in a color-coded environment, such as RoboCup soccer, and many object recognition modules rely on that.

Although many approaches to color segmentation have been proposed, in the official games of the RoboCup Four Legged League manual calibration is still preferred by most of the teams. In this paper we present a method for color segmentation that is based on an adaptive transformation of the color distribution of the image. In contrast with standard color space transformations that are applied constantly on all the images, in our approach the transformation is dynamically computed depending on the current image (i.e., it adapts to condition changes) and then it is used for color segmentation with static thresholds. The method requires the setting of only a few parameters and has been proved to be very robust to noise and light variations, allowing for setting parameters only once (e.g., when arriving at the competition site).

The approach has been implemented on AIBO robots, extensively tested in our laboratory, and successfully experimented in the some of the games of the Four Legged League in RoboCup 2005.

1 Introduction

RoboCup soccer is a color-coded environment, where colors are used to define principal objects needed for the robots to perform their robot tasks. Recognition and positioning of colored beacons and goals in the field are used for self-localization and reactive behaviors, while the recognition of the orange ball feeds behaviors and coordination tasks.

Consequently, color segmentation is typically the first step of the vision system of a robot playing RoboCup soccer. Since good color segmentation allows for easy implementation of object recognition and localization, most of the robot vision systems are based on fast and accurate implementation of such process. Conversely, it is also possible to recognize and locate objects from a rough segmentation (e.g., [6]), applying more sophisticated recognition techniques (e.g., region growing) at a later stage. However, this second approach may be less reliable or require more computational resources.

Many approaches to color segmentation have been proposed in the RoboCup soccer scenario. An analysis of these methods and comparisons with the approach presented in this paper are reported in Section 2. However, in the official games most of the teams still use manual calibration, since it provides for fast and accurate results, accepting the drawback of time consuming manual setting that is often repeated several time (e.g., just before each game) and for each robot (since color cameras have different response on different robots).

In this paper we present an approach to color segmentation that has been used for RoboCup soccer in the Four Legged League. The method performs an adaptive transformation of the color distribution of the image, that is dynamically computed during robot task and used for segmentation.

*This is an extended version of the RoboCup Symposium 2006 paper.

The approach integrates the following advantages: 1) it computes a dynamic transformation providing for robustness to noise and adaptivity to variable light conditions; 2) it uses static thresholds for fast segmentation; 3) it does not require time consuming manual calibration (only a few set of parameters must be set when arriving in a new location).

An effective implementation can be obtained by computing the transformation function periodically, for example every 25 frames (about 1 second), and when the robot is not in a critical phase of the game (e.g., about to approach the ball). In such steps results of color classification are stored in a lookup table (or color table), that is used for classification of the subsequent frames. In this way we reach the maximum performance of the image processing module (20-24 fps) for most of the frames (when such transformation is not computed), and periodically a lower performance (currently, slightly less than 100 ms) when this transformation is computed.

Experimental results show the effectiveness of the proposed method: a large data set of labeled images has been used to evaluate the approach in different locations and conditions. Furthermore, we have experimented the method during some of the games in RoboCup 2005. In that case we have set parameters once when we arrived at the competition site and then we used these parameters for some of the games and for the variable light challenge (with the very same code and settings), noticing no difference in the overall behavior of the team with respect to matches in which we have used manual calibration.

2 Related work

Color segmentation is an important task in many vision based applications and therefore the literature on this subject is very large. General approaches for tracking colored objects can be found for example in [2]. A large amount of work is also present in the RoboCup literature, where many authors have presented methods with the objective of specializing general techniques or providing new ones that are more suitable to this scenario.

In this section we briefly describe these related works, by classifying them in three groups: 1) coarse color segmentation; 2) off-line automatic calibration; 3) on-line color segmentation.

The first group includes works that use a coarse color segmentation that returns only a few correctly segmented pixels, and then start from these pixels an object recognition process that does not rely completely on a correct segmentation (see for example [6]). However, this kind of approaches usually demands more computational power for the object recognition module.

In the second group we classify methods that aim at generating a representation of the colors in the environment *off-line* with respect to the actual robot mission (e.g., before the game). This group can be further divided depending on the modality in which such representation is generated: a) by an external program from image sequences manually collected; b) by a pre-determined learning task executed by the robot. For example, the work in [8] describes the use of machine learning techniques to build a color table in a complete automatic way, by letting the robot execute a specific learning task, given only a model of the environment and its initial position. These approaches largely solve the problem of time consuming manual color calibration. However, once the representation is created, it is constantly used during the robot mission, thus being less robust to changing light conditions. Although in principle it is possible to detect significant illumination changes and re-run the automatic calibration process, in some scenarios (e.g., during a soccer game) this is very difficult to be achieved.

On the other hand, on-line color segmentation methods aim at dynamically adapting the representation used for color segmentation, thus being more robust to illumination changes. Many approaches for on-line color segmentation have been proposed in the RoboCup community. However, some of these approaches can be implemented in real-time only on robots with adequate computational resources (e.g., Middle-size robots). For example, [1] uses the Expectation-Maximization algorithm to adaptively track a model of the color classes over time. The computational performance of the approach are reported to be 5-8 fps on a 80x60 image with a 2.4 GHz CPU. In [4] color classes are learned with a region growing approach by using omni-directional vision that allows for partitioning the field in region with known colors. Time requirements for this method are quite high (from 0.5 to 2.9 seconds with a 933 MHz CPU). Another approach based on region growing is presented in [3], but implementation and testing on AIBO robots have not been reported there.

In the Four-Legged League, AIBO robots have very limited computational power (576 MHz CPU) and a low quality low resolution image acquisition device; moreover, real-time implementation of the vision process is extremely important for a correct execution of behaviors. An effective implementation of on-line color segmentation on AIBO robots has been reported in [5]. The paper presents an adaptive non-parametric method that computes color classes by representing them as cuboids in the YUV color space with different precision layers. Experimental results of this approach show good computational performance (about 5 fps). However, manual calibration is still preferred during the games [7].

The method presented in this paper is an on-line segmentation method that dynamically computes a transformation of the color distribution of the image. It is a parametric method, since it requires the setting of a few parameters that is usually performed manually from a small set of images. Moreover, in several cases only a few parameters must be changed when changing location. The presented approach has been actually implemented in such a way to obtain real-time performance on AIBOs, by performing dynamic computation only periodically. Moreover, it can be used for off-line generation of color tables that can be manually refined, speeding up manual calibration process.

3 Problem definition

Let us denote an image as \mathcal{I} , and each pixel of an image as $i \in \mathcal{I}$. A color space \mathcal{C} is a color representation of an image as captured by a color camera. Color spaces are typically represented with 3 components (e.g., RGB, YUV, HSV, etc.), different color spaces have different properties for color segmentation. We will use the notation $c(i)$ to represent the color of pixel i in a given color space. In case of a 3-D color space $c(i)$ is a vector with 3 components.

Color segmentation can be defined as the task of assigning for each pixel in the image one class within a predefined set of color classes. Color classes can be seen as a partition of the entire color space \mathcal{C} , $\mathcal{CC} = \{\mathcal{C}_1, \dots, \mathcal{C}_n\}$, such that $\mathcal{C}_k \subset \mathcal{C}$ and $k_1 \neq k_2 \Rightarrow \mathcal{C}_{k_1} \cap \mathcal{C}_{k_2} = \emptyset$. It is convenient to consider a complete coverage for the color classes, i.e. $\mathcal{C} = \cup_{k=1..n} \mathcal{C}_k$, possibly defining a special color class including undefined (or uninteresting) colors.

Therefore, color classification (or color segmentation) can be obtained by evaluating the function $f_{cc} : \mathcal{I} \rightarrow \{1, \dots, n\}$

$$f_{cc}(i) = k \text{ such that } c(i) \in \mathcal{C}_k$$

In this paper we focus on color classification in environment in which the number and the color classes to find are known in advance. This is the case for RoboCup soccer. In this context two possible kinds of approaches have been studied, that can be classified as static or dynamic.

Static segmentation. Approaches based on static segmentation, statically define the set of color classes before the robot starts its task and this is not changed during robot operation. Static definition of \mathcal{CC} can be obtained either by specifying a set of thresholds in the color space, a set of classification rules, an explicit representation of \mathcal{CC} , or an explicit representation of the function f_{cc} (also known as lookup table or color table).

Static approaches allow for a very fast implementation of the segmentation step, however they suffer of some well known limitations: they are sensitive to light variations, explicit representations of color tables are time consuming to be obtained (unless automatic learning procedures are used, e.g. [8]), static thresholds may be not sufficient to correctly partition the color space.

Dynamic segmentation. Dynamic segmentation is instead obtained by dynamically computing the color classes \mathcal{CC}_t at time t for each image \mathcal{I}_t , possibly considering previous color classes \mathcal{CC}_{t-1} .

It is thus adaptive since it is related to current image and is performed during robot task execution. Therefore it is more robust to noise and light changes and does not require intensive calibration. However, dynamic approaches may be computationally expensive and may not be applied to robots with low computation power (like AIBOs).

3.1 Analysis of color spaces

Dynamic segmentation is often obtained by an analysis of color spaces. To this end histograms of the images are considered. The histogram (or color distribution) of an image \mathcal{I} in the color space \mathcal{C} can be expressed by

$$D_{\mathcal{I}}(\gamma) = |\{i \in \mathcal{I} \mid c(i) = \gamma\}|, \gamma \in \mathcal{C}$$

A first method can be obtained by extracting dynamic thresholds as the local minima in the distribution of $D_{\mathcal{I}}$. The mean shift approach for analyzing feature spaces can also be very effective [2]. However, these methods typically cannot be implemented in real-time on the AIBOs. Moreover, they do not exploit the knowledge about the color classes that in RoboCup soccer are known in advance.

3.2 Dynamic Segmentation for AIBO soccer robots

AIBO robots have many limitations that must be considered in the development of their vision system: low computational power, low quality of the color image. Therefore, many of the state-of-the-art segmentation techniques in the literature cannot be implemented in real-time and do not give optimal results.

We provide an example analyzing a mono-dimensional color space: the H (hue) component of HSV. The reason for this choice is that H is a mono-dimensional color space that allows for discriminating among most of the colors of interest in RoboCup soccer (red, orange, yellow, green, blue). Since H is not defined when colors have not sufficient saturation, we use a fixed threshold on saturation to select pixels. In fact, pixels with low saturation are used to determine other colors (for example, white) and static thresholds on saturation can be used to distinguish color with same hue: red vs. pink and sky blue vs. blue.

The choice of using only H has been proved to be efficient, since analysis is performed on a mono-dimensional space, as well as effective in recognizing the colors of interest in RoboCup, as shown by experimental results. However, the same concepts expressed in this paper can be applied to other color spaces (not necessarily mono-dimensional ones).

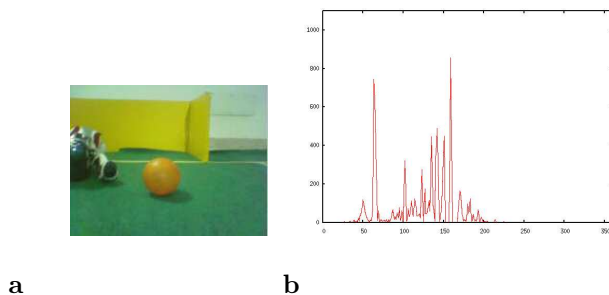


Figure 1: An example of color distribution in the H color space.

Figure 1b) shows a typical color distribution of the H component from the displayed image. The distribution is not smooth (denoting high frequency noise in the color response of the camera) and thus finding dynamic thresholds there is not easy.

4 Color segmentation though color distribution transformation

The approach to color segmentation proposed in this paper is based on an adaptive transformation of the color distribution of an image and subsequent static thresholding. The aim of this approach is to integrate the robustness to noise and light changes typical of dynamic methods with efficiency of static ones.

More formally, we define a transformation function $\tau : \mathcal{C} \rightarrow \mathcal{C}$, and we call τ -distribution the new color distribution obtained by applying the function τ to the color distribution of an image, and τ -transformation the operation of computing the τ -distribution of an image. Having τ , color classification is obtained by

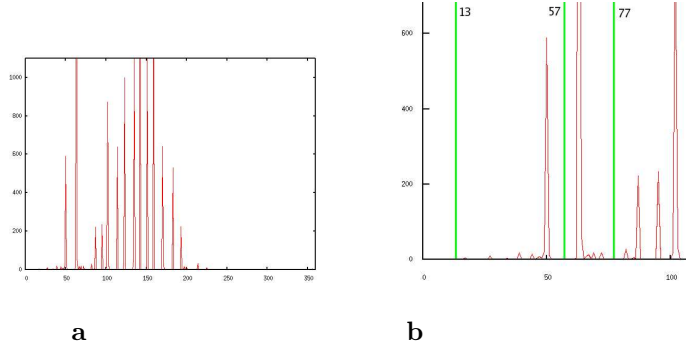


Figure 2: τ -distribution of the image in Fig. 1.

$$f_{\tau,CC}(i) = k \text{ such that } \tau(c(i)) \in \mathcal{C}_k \quad (1)$$

The objective of this transformation is to modify the color distribution of an image in such a way that the subsequent use of fixed thresholds can be effective for color segmentation, i.e., both robust to noise and light variations and efficient. In particular, we would like that the new color distribution do not vary too much in presence of light variations, specially in proximity of the thresholds. To obtain an adaptive method, τ is periodically computed from the current images the robot acquires.

Figure 2a) shows the τ -distribution of the image in the previous Figure 1. The function τ is computed by the algorithm described in the next section of this paper. The τ -distribution is roughly formed by a set of peaks: by interpreting the values of the H component in the X axis of the diagram, it is easy to recognize the left-most peak as corresponding to orange color (hue is about 50), the second from the left corresponding to yellow (hue is about 60), the others corresponding to green (hue > 100). Figure 2b) shows a detail of Figure 2a) with lines denoting static thresholds (respectively red-orange, orange-yellow and yellow-green separation values). Notice that the peaks of the transformed color distribution are all within the thresholds thus actually resulting in no misclassification. Moreover, the large distance between the peaks allows for obtaining good results without requiring fine tuning for the thresholds, making also the method not sensible to small variations (in the order of 10 deg for the H component of HSV space) in the color distribution due to illumination changes.

It is important to notice here that thresholds for a good segmentation of this image can be found as well by analyzing the original color distribution (Figure 1b)). However, these values may not be adequate for subsequent classification in presence of illumination changes.

A second example is shown in Figure 3. Figure 3b) shows the original distribution (lighter color), the τ -distribution (darker color) and three values for the orange-yellow threshold (vertical lines), respectively 51, 57, and 64. The images in Figure 3c) are the results of segmentation with such different values for the orange-yellow threshold, by using τ -distribution (left side) and original distribution (right side). It is possible to see that the segmentation obtained with τ -distribution is more robust since it essentially returns the same correct segmentation for all the values of the threshold, while the segmentation obtained with the original distribution produces some bad classification (orange is classified as yellow when the threshold is lower, yellow is classified as orange when the threshold is higher). This example shows that segmentation on the τ -distribution is more robust. In fact, any value for the orange-yellow threshold between 51 and 64 would have been good.

4.1 Computation of $\tau(\cdot)$

In this section we present the algorithm for computing the transformation $\tau(\cdot)$. As already mentioned, in this paper we refer to the color space H, that we represent as degrees discretized to 360 integer values (i.e. $H = [0, 360)$).

The transformation $\tau(\cdot)$ is computed by Algorithm 1, by using an explicit representation of the τ function (i.e., a vector $\tau[0 : 359]$)

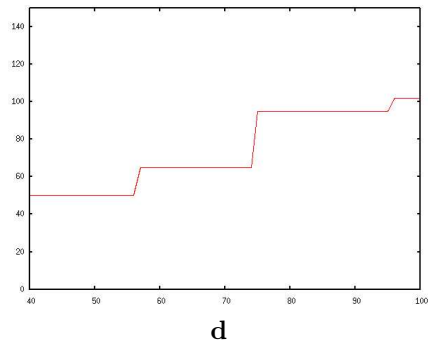
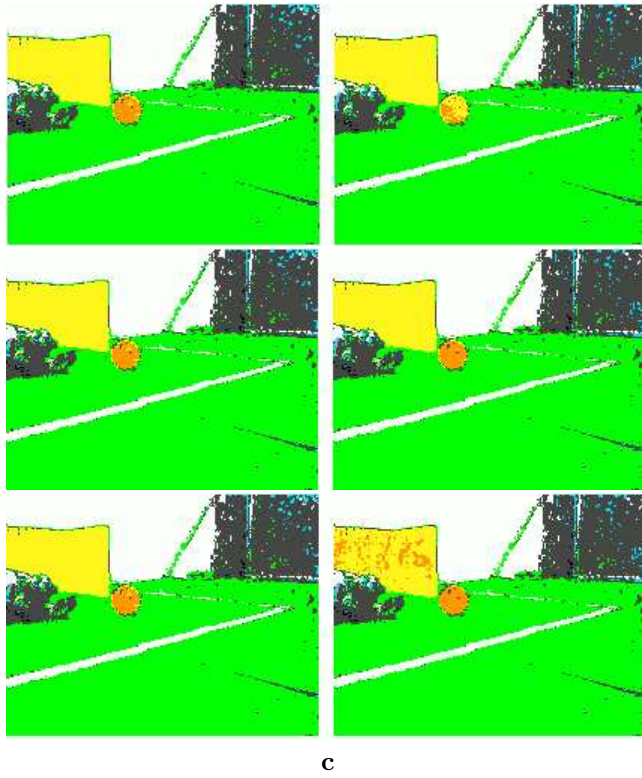
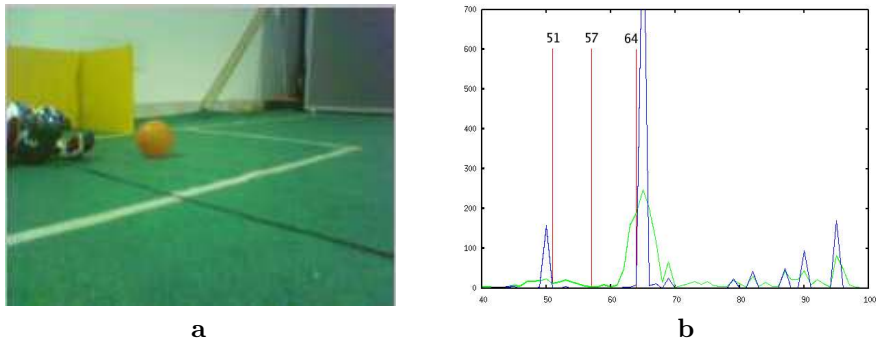


Figure 3: a) Original image; b) Color distributions; c) Color segmentation; d) A portion of $\tau(\cdot)$ function for image a).

Algorithm 1 Color distribution transformation

Input: Image \mathcal{I} , window size δ

Output: Function $\tau(h)$, $h \in H$

(explicit representation though a vector $\tau[0 : 359]$)

```
for each  $\gamma \in [0 : 359]$  set  $\tau(\gamma) = \gamma$ 
compute color distribution  $\mathcal{D}_{\mathcal{I}}(\gamma)$ 
for  $\gamma = \delta/2$  to  $360 - \delta/2$  do
   $\tau[\gamma] = \mathbf{argmax}_{\lambda \in [\gamma - \delta/2, \gamma + \delta/2]} \{\mathcal{D}_{\mathcal{I}}(\lambda)\}$ 
end for
while  $\tau[\gamma] \neq \gamma \wedge \tau[\gamma] \neq \tau[\tau[\gamma]]$  do
   $\tau[\gamma] = \tau[\tau[\gamma]]$ 
end while
```

The function τ is first initialized to the identity function, i.e. $\tau(\gamma) = \gamma$, then two steps are performed: the first step assigns to $\tau[\gamma]$ the value $\gamma^* \in [\gamma - \delta/2, \gamma + \delta/2]$ for which $\mathcal{D}_{\mathcal{I}}(\gamma^*)$ is maximum in such interval; the second step performs a transitive closure of the τ function. The value δ is used to limit the interval in which the maximum value is searched. This value is set in such a way to process a significant interval of values in the color space: smaller values of δ may fail to accumulate colors over a single component (thus producing more peaks), larger values may collapse different colors in a single peak. By empirical tests, we have determined that a value around 10 provides good results in our implementation.

An example of τ is given in Figure 3 d). It is the result of the presented algorithm for the image in Figure 3 a). As shown in this figure the transformation tends to accumulate all the colors within an interval towards a nominal color that represents them. The nominal color is dynamically computed depending on the current image, but it usually remains within a fixed interval of values that allows for correctly using static thresholding. For this image the accumulation values for h are around the values 50, 65, 95 (corresponding to the peaks in the Figure 3b)), that are widely within the static thresholds for orange-yellow (57) and yellow-green (77) used in the segmentation illustrated in Figure 3.

The approach presented in this section can be extended to automatically generate a color table from a set of images, for example by selecting for each value in the color space the color class that has been chosen more often during the sequence. This is useful for off-line automatic calibration or for generation of a first color table to be manually refined.

4.2 Implementation issues

To effectively implement the approach a few implementation issues must be considered. In this section we first analyze the parameters that must be set and then discuss how to obtain a real-time execution.

Color distribution on H component is computed only for pixels in the image that have sufficient saturation and intensity. We defined two thresholds for each of these other two components: s_1, s_2 with $s_1 < s_2$ for the saturation and v_1, v_2 with $v_1 < v_2$ for the intensity. Typical values are $s_1 = 30, s_2 = 60, v_1 = 15, v_2 = 30$, they must be changed in case of special light conditions. For example, if the environment is not well illuminated then values v_1, v_2 should be reduced. Given these thresholds, in the computation of color distribution $\mathcal{D}_{\mathcal{I}}(\gamma)$ we count 1 for pixels whose saturation and intensity are above the thresholds s_2 and v_2 , 0.5 for pixels whose saturation and intensity are between s_1, v_1 and s_2, v_2 , 0 for all the other pixels.

After τ is computed, color classification is based on static thresholds in the H color space. We define five thresholds to separate five principal colors: red, orange, yellow, green, blue, respectively denoted as $h_{ro}, h_{oy}, h_{yg}, h_{gb}, h_{br}$. Note that H is a circular color space wrapped around the red color, thus red color class is given by $[0, h_{ro}) \cup [h_{br}, 360)$.

Besides the thresholds on H, we found useful to add lower limits to saturation and intensity for each color class. Thus adding other ten parameters. These values should be set depending on the actual colors used for the objects and on the actual light conditions. Therefore, they typically require a manual setting when arriving at a competition site.

Computational efficiency of color segmentation and image processing is critical for AIBO soccer robots. Manual calibration is very efficient since it computes a static lookup table (or color table) that is used for color classification with a single memory access for each pixel. Instead the method described in this paper requires a scan of the image for building the color distribution, the application of Algorithm 1 on a mono-dimensional color space H , the check for the static thresholds defined above. This process cannot be implemented at full frame rate on AIBOs.

This problem is solved in two ways: first, computation of τ is not performed for every image; second, color distribution is not computed for every pixel in the image.

Fortunately, it is not necessary to compute τ for every image. In fact, variable light conditions affect the color distribution but not the transformation that allows for a correct classification through static thresholding. Once τ is computed, it can be used to generate a color table (representing the classification function $f_{\tau,CC}$). This color table can be used for subsequent frames and in general as long as light conditions are stable enough. Several strategies can be used to determine when it is convenient to re-compute τ and thus the color table. Our simple choice is to re-compute it every 25 frames (about 1 second) unless the robot is involved in a critical phase of the game (e.g., about to approach the ball). In this way for most of the frames we have full frame rate (20-24 fps) since we use a static color table, with the advantage that the color table is dynamically generated considering recent light conditions.

When we decide to re-compute τ , we compute color distribution by analyzing only a subset of the points in the image. To select them we consider only the pixels around and below the horizon line and sub-sample them (e.g., taking one every two pixels). Since the robot acquires images in YCrCb format conversion to HSV is obtained through a lookup table in which we encode H with 8 bits (thus scaling from 360 values to 255 values), S and V with 4 bits each. We also discard the two least significant bits of YCrCb, thus this lookup table is formed by $64 \times 64 \times 64$ components of 2 bytes (i.e., 512 Kb). After τ is computed and static thresholds are applied we fill a $64 \times 64 \times 64$ color table assigning to each color in the YCrCb space the color class computed by the classification function $f_{\tau,CC}$, using the current τ . The overall process of computing τ and generating the color table takes slightly less than 100 ms.

5 Experiments

In order to evaluate the approach presented here we have performed several experiments. In this section we first discuss about performance metrics of color segmentation algorithms, and then present the results of our experiments.

5.1 Performance metrics

Defining performance metrics for color segmentation can be relatively easy. For example, we may manually label a set of images by correctly classifying each pixel and then compute classification rate of a segmentation method as the number of correctly classified pixels. Manual labeling can be performed only on a subset of pixels relative to important objects in the scene (see for example [8]). This is a good metric for evaluating color segmentation methods, but it requires a lot of time to generate the ground truth (since several pixels on each image must be manually classified). Moreover, typically object recognition processes that follow the segmentation phase are somewhat robust to small errors in classification, making this measure perhaps too much demanding. In fact, to our knowledge it does not exist a large data set of images labeled in this way, that can be used as a standard benchmark for segmentation algorithms (the one reported in [8] is limited to only a few dozens of images).

In this paper we propose an alternative metric for evaluation of a segmentation algorithm. This metric can be considered as an approximation of the above one, with the advantage of being much easier to be used to produce ground truth, thus allowing for easily generating larger data sets to be used for comparing different methods.

Instead of labeling each pixel on an image, we define a 4-sided polygon around any object that must be recognized in the environment. For the RoboCup soccer environment we chose to consider ball, goals and beacons. When objects are partially occluded or in case of non-polygonal shapes (e.g., a goal seen closely), we anyway define the best fitting 4-sided polygon around the object. The ball is instead modeled with an ellipse.

	Location	Frames	Balls	Goals	Beacons
1	Lab 1	395	84	110	163
2	Lab 2	593	286	107	196
3	Osaka	193	70	63	76

Table 1: Data sets used in the evaluation.

From a labeled image and the result of color segmentation on that image, we can measure the percentage of correctly classified pixels within such polygons. Let us denote with $\eta_o(\mathcal{I}_t)$ the percentage of correctly classified pixels of object o at frame \mathcal{I}_t . We want to measure the performance of the method over all the frames, i.e. over the entire data set $\{\mathcal{I}_t\}$. A simple choice would be to compute average and standard deviation of this value over t . However, by plotting typical responses of $\eta_o(\mathcal{I}_t)$ over t we see that the outcome is usually multi-modal, thus average and standard deviation seem not to be a good measure for the overall sequence. We found, for example, that in presence of motion blurring the percentage $\eta_o(\mathcal{I}_t)$ for the beacons may be very low. In order to summarize the behavior of the system over time, we propose to compute a distribution over η_o , counting how many times the pixels within the object o are correctly classified with a percentage rate of at least η_o . More specifically, we want to define $V_o(\lambda)$ as the percentage of times in which at least a percentage λ of pixels within the object o have been correctly classified. For example, $V_{ball}(0.5)$ expresses the percentage of frames in which at least half of the pixels of the ball have been correctly classified. $V_o(\lambda)$ ($\lambda \in [0, 1]$) is thus defined as

$$V_o(\lambda) = \frac{|\{\mathcal{I}_t | \eta_o(\mathcal{I}_t) \geq \lambda\}|}{|\{\mathcal{I}_t | \mathcal{I}_t \text{ contains object } o\}|}$$

This value can be directly related to the capabilities of an object recognition module. For instance, we have empirically determined that current implementation of our object recognition module allows for correct recognition of objects if at least about 60 % of the pixels are correctly classified.

5.2 Experimental data sets

To perform evaluation reported in the next section we have collected and manually labeled a set of image sequences as described above¹. Each image sequence has been recorded during normal behavior of a robot in a RoboCup Four-Legged field. Therefore, all the difficulties of image processing are included in the data sets: different light conditions, motion blurring, partial occlusions of the objects, etc. We have labeled many data sets from different locations: Osaka 2005, German Open 2005, and several data sets from two different locations of our lab. Each data set includes several sequences taken at different times (possibly from different robots) for a total of over 2,000 frames at 208x160 pixels resolution.

5.3 Experimental evaluation

Evaluation of the proposed method has been performed by computing the functions $V_{ball}(\lambda)$, $V_{goal}(\lambda)$, and $V_{beacon}(\lambda)$ over a number of different data sets, comparing the results of the approach presented in this paper and segmentation obtained by manual definition of a color table. The first two data sets used (see Table 1) are taken from two different locations of our laboratory, using the same elements for the field and the same kind of illumination devices, but in different external light conditions. The third data set has been taken during RoboCup 2005 in Osaka.

For the first data set we have manually created a color table for static segmentation and tuned the parameters of the method proposed in this paper. We then use the same configuration for all the three data sets. For each data set, we have thus evaluated the functions $V_o(\lambda)$ for each object (ball,goal,beacon) obtained with the method presented here and with segmentation through static color table. To evaluate the rate of false positives we have computed the ratio between the number of pixels classified with colors belonging to the objects (i.e., orange, yellow, sky blue and pink) that are outside of the polygons of the objects provided by the ground truth and the total number of pixels in the image.

¹The data sets are available from <http://www.dis.uniroma1.it/~spqr>

The results are reported in Figures 4, 5, 6, where red (darker) plot indicates our method, while the green (lighter) plot is the result of static segmentation and in Table 2.

As expected, static segmentation outperforms the dynamic method presented here in the first data sets, where it has been calibrated. However, the graphs also show that our method is very competitive. For example, in the worst case (the ball) values at $\lambda = 0.5, 0.7, 0.9$ are 0.87, 0.76, 0.50 for our method and 0.89, 0.82, 0.57 with manual calibration. Also the rate of false positives are similar.

In the second data set, that is similar to the first one but in different external conditions, we still have comparable results. However, while there is not a clear distinction in the functions $V_o(\lambda)$, the false positive errors are much larger when using static segmentation.

In the third data set, that is quite different from the first one, instead there is a clear difference in the performance. Our method provides for higher robustness to different lighting conditions, and in this case also to different environment set up.

These experiments show that the method proposed here is competitive with static segmentation obtained by manual calibration and robust to different light conditions and different environments.

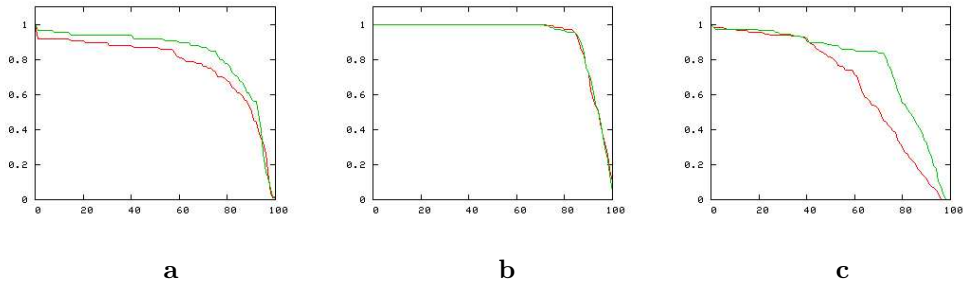


Figure 4: First data set: a) $V_{ball}(\lambda)$, b) $V_{goal}(\lambda)$, c) $V_{beacon}(\lambda)$

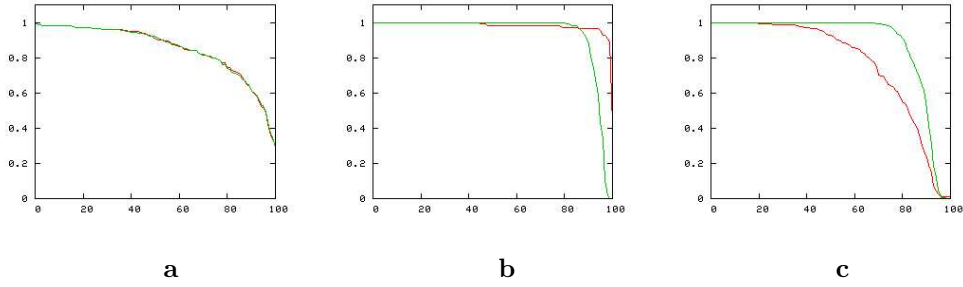


Figure 5: Second data set: a) $V_{ball}(\lambda)$, b) $V_{goal}(\lambda)$, c) $V_{beacon}(\lambda)$

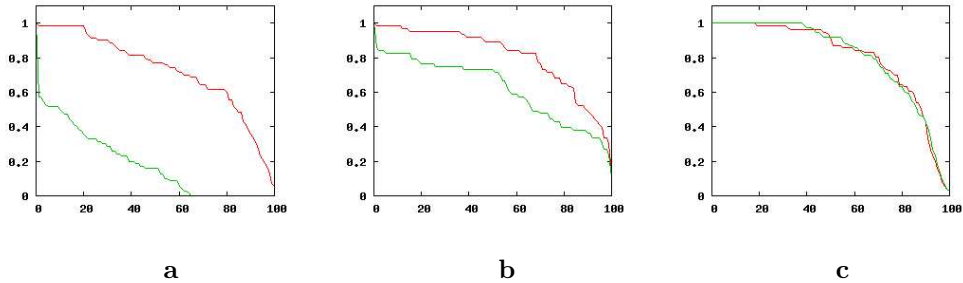


Figure 6: Osaka data set: a) $V_{ball}(\lambda)$, b) $V_{goal}(\lambda)$, c) $V_{beacon}(\lambda)$

Data Set	Our method	Static segmentation
	avg/max	avg/max
1	0.006 / 0.044	0.010 / 0.057
2	0.013 / 0.062	0.033 / 0.094
3	0.070 / 0.242	0.065 / 0.225

Table 2: False positives average and maximum rates.

6 Conclusions

In this paper we presented a dynamic color segmentation approach based on adaptive transformation of the color distribution of an image, that is suitable for implementation on robots with low computational resources and effective in presence of noise and illumination changes. The approach has been successfully implemented on AIBOs and extensively experimented in laboratory as well as during official RoboCup games. A new evaluation approach has been proposed and a large data set of labeled images have been created to evaluate and compare different methods.

The present method requires setting only a few parameters usually once arrived at a new location, and then it is robust to different light conditions over a typical competition period. As future work we intend to exploit machine learning techniques for learning the few parameters the method require for realizing a complete non-parametric method.

References

- [1] F. Anzani, D. Bosisio, M. Matteucci, and D.G. Sorrenti. On-line color calibration in non-stationary environments. In *RoboCup 2005: Robot Soccer World Cup IX*, 2005.
- [2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.
- [3] A. De Cabrol, P. Bonnin, T. Costis, V. Hugel, P. Blazevic, and K. Boucheфра. A new video rate region color segmentation and classification for sony legged robocup application. In *RoboCup 2005: Robot Soccer World Cup IX*, 2005.
- [4] K. Gunnarsson, Wiesel F., and R. Rojas. The color and the shape: Automatic on-line color calibration for autonomous robots. In *RoboCup 2005: Robot Soccer World Cup IX*, 2005.
- [5] M. Jüngel. Using layered color precision for a self-calibrating vision system. In *RoboCup 2004: Robot Soccer World Cup VIII*, 2004.
- [6] N. Lovell. Illumination independent object recognition. In *RoboCup 2005: Robot Soccer World Cup IX*, 2005.
- [7] W. Nisticò and T. Röfer. Improving percept reliability in the sony four legged league. In *RoboCup 2005: Robot Soccer World Cup IX*, 2005.
- [8] M. Sridharan and P. Stone. Autonomous color learning on a mobile robot. In *Proc. of AAAI*, 2005.