On-Line Color Calibration in Non-Stationary Environments *

Federico Anzani[1], Daniele Bosisio[1]
Matteo Matteucci[1], and Domenico G. Sorrenti[2]

[1] Politecnico di Milano, matteucci@elet.polimi.it
[2] Università di Milano - Bicocca, sorrenti@disco.unimib.it

Abstract. In this paper we propose an approach to color classification and image segmentation in non-stationary environments. Our goal is to cope with changing illumination condition by on-line adapting both the parametric color model and its structure/complexity. Other authors used parametric statistics to model color distribution in segmentation and tracking problems, but with a fixed complexity model. Our approach is able to on-line adapt also the complexity of the model, to cope with large variations in the scene illumination and color temperature.

1 Introduction

Color is a strong clue for object recognition, and for this reason it is used in many industrial and research applications. The real-robot leagues of Robocup, one of which is the mid-size league, are challenging applications conceived to use information about color. Here we have color codes for each relevant object in the robot world, altogether to changing lightning conditions in a dynamic scene with many objects to recognize. These varying conditions represent a non-stationarity in the environment and they are the main issues that prevents a reliable object recognition in natural light conditions, as clearly stated in [1].

The literature in this field is huge, we just mention here some works from Robocup literature. Cameron and Barnes [2] approach the problem relying on the a priori known scene geometry for identifying some regions of the image as corresponding to some features of the environment; this is done without the use of color-classification, but on domain-specific knowledge instead. Then the color information of such regions is used for building the color classifier(s); in order to track the dynamics of the lightning they associate the would-be-current classifier(s) with the previous, so to keep them tied to the symbolic label(s).

In Jüngel et al. [3] colors are defined by simple partitions parallel to the color space axes; in order to attain adaptation the colors are defined relative to a reference color which, in the Robocup legged league, the example application of the paper, is the green. No explanation is given about this choice. Our opinion is

* This work has been partially supported by the project P.R.I.N. 2003 "Software tecnologies and knowledge models for design, implementation and testing of multi-agent robotic system in real environments", funded by M.I.U.R.

that the reason is the (domain-specific) observation of its nearly-granted presence in the image (as mentioned e.g., by [4]). The setup for the classification of each color is executed off-line, and then a domain-specific heuristic is used to track the reference color, while other colors are classified by displacing their region in the color space by the same amount the reference color has moved. This is risky, as noticed by [1], because distances in the color space can be stretched by changes in the illumination.

Color constancy [5] approaches have been also used to reconstruct the incident light and adjust perceived colors for automatic color calibration [6]. However these approaches have unpractical computational requirements and thus cannot be applied on-line for real-time tracking of color changes. Different issues prevent the effective use of histogram based approaches [7]; they require a large number of data-points (pixels) and a coarsely quantized color space. In the absence of an accurate model for the apparent color, which changes over time, these models for density estimation cannot be obtained and a different approach is required. The subsequent sections briefly presents the parametric approach to color modeling which is at the basis of our contribution and the algorithm used for adaptation; experimental results on real images and conclusions are presented in the final sections.

2 Mixture of Gaussians for Classification and Modeling

In order to attain adaptivity, we use a generative approach, which implements a probabilistic color modeling and classification scheme based on mixture models. This approach has been originally proposed to implement object tracking under non-stationary illuminating conditions [8, 9]. In this context, the color of each pixel is supposed being determined according to a given probability distribution, which depends on the colors in the observed object and is characterized by some parameters.

Let us begin by temporary forgetting the color labels and limiting to model the probability distribution of color values in the color space. This distribution will be multi-modal not only because we expect more than one color, but also because more than one tone will be present for some color; this can be due, e.g., to different lightning conditions, in different places, for the same object. This probability distribution allows a probabilistic classification of the image pixels and in our case is approximated by the use of a parametric model, i.e., a mixture probability distribution.

In the scheme introduced by McKenna in [9], we have pixel values $\mathbf{x}$ in a color space of D dimensions (i.e., 2 dimensions in McKenna's word), an object O, and $j = 1, ..., m$ components, which live in the color space too. The model is trained on the pixels known to belong to the object; the whole object is represented with a mixture of Gaussians, i.e., components. Our problem requires to deviate a little from this object-based scheme, because our aim is color modeling and not direct object recognition. We still have pixel values in the color space (whos dimentions are also called color components, but we shall not use this term component with

this semantic any more, to avoid confusion with mixture model components), but we have not a single object to recognize. We have many, i.e. much more than one, color labels which we would like to recognize. Each color label could require more than one component in the color space, for an adequate modeling. This trained model will be used for classification of online data.

To gain a better understanding of the parametric model involved by such a mixture distribution, suppose we have J different colors in the scene and let $\mathbf{x}$ be the vector of numbers representing the pixel values in the color space (e.g., the RGB or HSV coordinates for that pixel). The probability of observing this vector depends on the probability of the vector of features given the real color label of the object in the scene C_j. However, suppose we do not know in advance the real color of the observed object, but only the probability of the observed object being of a certain color $p(C_j)$, by the total probability theorem, we can write the probability of each pixel observed as

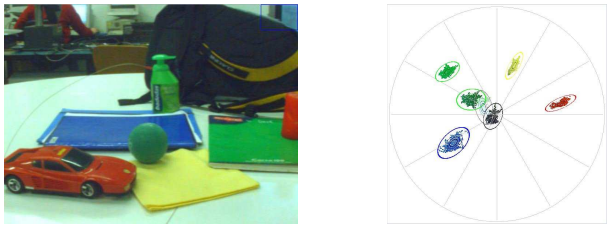
$$p(\mathbf{x}|\Theta) = \sum_{j=1}^{J} p(\mathbf{x}|\theta_j, C_j)p(C_j) \tag{1}$$

where $\sum_{j=1}^{J} p(C_j) = 1$, $p(C_j)$ is the so called *mixing probability* and corresponds to the prior probability that $\mathbf{x}$ was generated by the j^{th} component; $p(\mathbf{x}|\theta_j, C_j)$ is the conditional density for the pixel color features given that this pixel belongs to color component C_j (i.e., our generative model). The term Θ, in Equation 1, represents the set of all parameters describing the pixel probability distribution including conditional probability parameters θ_js and color label probabilities $p(C_j)$s. We can extend this mixture model to have more components than the J colors in the scene since we can associate two or more components to the same color label. Moreover, if we suppose to have M components and the conditional probability of the feature color vector being a Gaussian density, the probability of a pixel feature vector becomes a Gaussian mixture model [10, 9]

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{M} \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}[\mathbf{x}-\mu_k]^T \Sigma_k^{-1}[\mathbf{x}-\mu_k]} \cdot p(C_k) \tag{2}$$

In this model, each mixture component is modeled by a Gaussian distribution with mean μ_k and covariance matrix Σ_k. This generative model is somehow independent from the logical color label associated to each component and its parameters can be obtained by a maximum likelihood estimation of mixing probabilities $p(C_j)$s and Gaussians parameters θ_js. The association between each mixture component and the appropriate color label can be obtained by a (human) supervised labeling of components. Actually, in our method, this is implemented in a different way by using a (human) labeled dataset to initialize mixture parameters (i.e., initial number of components, mixing probabilities, and Gaussian parameters) and associate from the very beginning the appropriate labels to the components of the mixture.

Modeling the color distribution with a mixture of Gaussians can be easily explained: in Figure 1(b) patches of colors from image in Figure 1(a) are modeled



(a) (b)

Fig. 1. An example of color density model obtained using a mixture of Gaussians, on the left the original image, on the right the probability distribution of image pixels; the lines represent the curve where the cumulative probability reaches the 0.9 value

by a mixture of 2D Gaussian in the Hue-Saturation (HS) color space. Black and white pixels are also projected on the $V = 255$ plane just for easing the visualization; please notice how the white pixel of the image have a greenish hue as well as the presence of two different green components due to spatial variation of color.

At classification time, each pixel is classified according to the conditional probability of being generated by a certain color component to which is associated a label. This is implemented by the maximum a-posteriori criterion for the selection of the color component, and label, by attributing to the pixel the color component, and label, of the component which is the most likely:

$$C(\mathbf{x}) = \arg\max_{C_j} p(C_j|\mathbf{x}, \Theta) = \arg\max_{C_j} p(\mathbf{x}|\theta_j, C_j)p(C_j). \tag{3}$$

In really dynamic environments, the mixing probability might change almost at any frame due to occlusions and changes in the field of view. This would interfere with the maximum a-posteriori criterion since the prior probability for each class $p(C_j)$ would change with the observed scene. For instance, suppose we miss the red ball for a few frames because of occlusions, using a null mixing probability as prior will prevent to detect it as it will come back in the field of view. This can be faced by adopting, at classification time, an improper uniform prior for color labels (i.e., $p(C_j) = 1/J$) turning the maximum a-posteriori approach into maximum likelihood classification:

$$C(\mathbf{x}) = \arg\max_{C_j} p(C_j|\mathbf{x}, \Theta) = \arg\max_{C_j} p(\mathbf{x}|\theta_j, C_j). \tag{4}$$

3 On-Line Adaptation with Expectation-Maximization

We can now come to the main issue of the work, i.e. adaptation of the model to changing light conditions. To overcome issues due to non stationarity, we use *Expectation-Maximization* (EM) algorithm [11] to adapt model parameters over

time. EM is an iterative algorithm that computes maximum likelihood estimation, in a very efficient and fast way, by optimally coping with missing data (i.e., we do not know mixing probabilities in future frames). Let N be the number of data-points in the data-set. The Estimation step of EM computes the "expected" classes at time t for all data points $\mathbf{x}_n$ using the current Gaussian parameters:

$$p(C_j|\mathbf{x}_n, \theta_j^{(t)}) = \frac{p(\mathbf{x}_n|C_j, \mu_j^{(t)}, \Sigma_j^{(t)})p(C_j)^{(t)}}{\sum_{i=1}^{M} p(\mathbf{x}_n|C_i, \mu_i^{(t)}, \Sigma_i^{(t)})p(C_i)^{(t)}} \tag{5}$$

The Maximization step of EM computes the maximum likelihood estimates of the mixture distribution given the data class membership distributions [12]:

$$\mu_j^{(t+1)} = \frac{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})\mathbf{x}_n}{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})} \tag{6}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})[\mathbf{x}_n - \mu_j^{(t+1)}][\mathbf{x}_n - \mu_j^{(t+1)}]^T}{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})} \tag{7}$$

$$p(C_j)^{(t+1)} = \frac{\sum_n p(C_j|\mathbf{x}_n, \theta_j^{(t)})}{N} \tag{8}$$

EM is proved to reach a local maximum, and the optimality of the extremum depends on the initial estimate of mixture parameters, especially when these parameters are randomly selected. In our application, the components means and variances are initialized by using hand-classified sets of pixels from the first frame; the mixing probability is taken proportional to the density of the corresponding subset of data used in the initialization and the number of components is initializated to the number of classified sets.

From each subsequent frame, a new set of pixels is sampled and inserted in the data-set without a label. One step of EM is computed on this augmented data to provide model adaptation. A soft and stable adaptation is granted by weighting data points, increasingly from the older to the newer, by using a forgetting factor. To avoid excessive computations with reduced marginal gains, data points older than a fixed threshold are removed from the data-set.

To understand how EM is able to track non-stationary distributions and thus adapt the color model to illumination changes, refer to Figure 2. In this case we present a (simulated) data-set where a cloud of points, referring to the same color, moves in the color space during time because of non-stationary light conditions. From the image sequence it can be noticed how the algorithm performs an unsupervised probabilistic clustering of the data while adapting to the distribution changes.

4 Adaptation of Model Complexity

In modeling with mixture of Gaussians, the main problem is to choose the right structure for the model, i.e, the *right* number of components of the Gaussian mix-

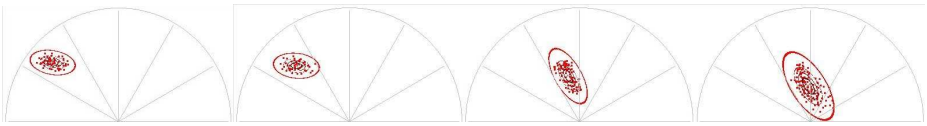


Fig. 2. Distribution tracking using EM (upper part of HS diagram reported)

ture. A common approach in stationary situation (e.g., data mining) is to try every possible structure (up to a given threshold) and select the best one. However, this method cannot be used in real-time dynamic contexts because of both non-stationarity and excessive computational cost. In the following subsections we extend the mixture of Gaussian model previously described by introducing an automated procedures to on-line adapt the model complexity.

4.1 Reducing Mixture Components

Since data distribution varies over time, the number of Gaussian mixtures could become higher than the optimal value. This strongly affects the performances of the algorithm and could lead to overfitting the noise in the data. This unrequired complexity of the model can be reduced by merging clusters that could be represented by only one Gaussian without losing precision in the model and preserving the correct classification with respect to color labels. This situation happens quite often when the scene become darker (e.g., clouds in natural light context or light bulbs switched off in offices) and the color distribution, as represented in the HSV space, concentrates on the bottom of the cone.

To perform cluster merging we use the greedy approach; at every frame, before computing EM, the two nearest components, for every label, are selected as candidate for the merging. In our current implementation we use the Euclidean distance although better and more sound results could be obtained using the Mahalanobis distance, given that our components are Gaussian. To decide if they have to be merged or not, a new Gaussian component is created with mean and variance computed using all the points belonging to the two candidate-to-the-merge components.

In order to estimate which of the two models (i.e. the merged single component or the original two separate components) is the best model, the Bayesian Information Criterion (BIC) [13] is used. Assuming we are given the data $\mathcal{D}$ and a family of alternative models M_j, this criterion uses the posterior probabilities $p(M_j|\mathcal{D})$ to give a score to the models, using the following formula, from Kass and Wasserman [14]:

$$BIC(M_j) = \hat{\mathcal{L}}_j(\mathcal{D}) - \frac{P_j}{2}\log R \tag{9}$$

where $\hat{\mathcal{L}}_j(\mathcal{D})$ is the log-likelihood of the data according to the j-th model, taken at its maximum-likelihood point, P_j is the number of free parameters in M_j and

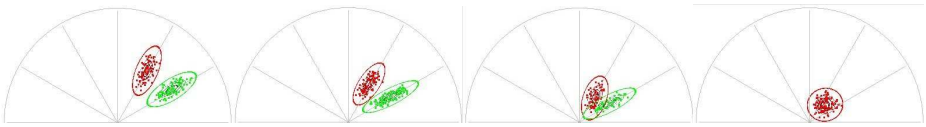


Fig. 3. An example of merging of components, in the HS plane. The components can merge only if they have the same label. This is a synthetic example, built after real cases, which, for an easier understanding, presents components referring to just one label. This behaviour usually shows up when the intensity of the light on the scene decreases.

R is the number of data used to fit the model. In order to increase the speed only two components are examined at a time. The algorithm iterates until no more components can be merged, according to the BIC scoring.

4.2 Increasing Mixture Components

In this section we present our proposal for overriding the opposite problem of merging components, i.e. when one cluster of pixel values in the color space divides in two. If this situation goes undetected the model will loose one of the two, by classifying it as noise; or, alternatively, the two clusters of pixel values will be modelled as a single component with large variance. This situation usually occurs when the observed scene becomes brighter and the different tones of the same color begin to separate in the color space.

We propose a very simple, but effective, approach in order to detect potential splitting cases based on a measure of data-points density. Let $N_{(p)}$ denote the number of data-points contained in a given confidence interval $V(\overline{p})$, i.e., the hyper-ellipsoid centered on the Gaussian mean and containing in its volume a probability mass $\overline{p}$. The average density $D(\overline{p})$, as a function of the confidence $\overline{p}$, can be computed as: $D(\overline{p}) = N_{(\overline{p})}/V(\overline{p})$.

The approach is based on the observation that when we would like to split, we have a much lower density around the mean of the component than the average values across the component. $D(0.95)$ could be considered a reasonable approximation of the average density for the whole component while $D(0.25)$ can be taken as a reasonable approximation of the density about the mean. Using a K coefficient, empirically we found that a good value for K is between 1.5 and 1.8, if $D(0.95) > K \cdot D(0.25)$ we decide the current model does not fit the data distribution well enough. We then split the cluster along the direction of maximum variance. The two resulting components will be initialized with mean at $\frac{1}{4}\sigma$ and $\frac{3}{4}\sigma$ respectively, and a diagonal covariance matrix with diagonal elements set to $\frac{1}{4}\sigma^2$. The subsequent EM steps will fine tune the parameters of the components, Figure 4.

It is quite easy to find counter-examples, e.g. in data-mining, where this heuristic does not work properly, but we found in our experiments that in the

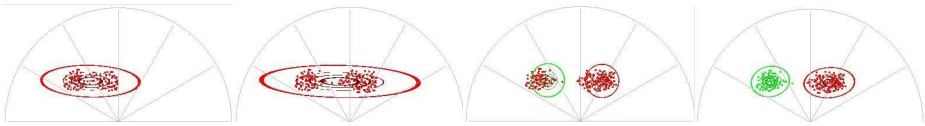


Fig. 4. An example of splitting of components, in the HS plane (left to right). Note in the third frame that the two new components are initialized to symmetrical Gaussians; in the next frame the EM algorithm adjusts their parameters.

color domain, it is highly reliable. Moreover, it involves a small amount of computation and does not slow-down the algorithm significantly. Another solution could be to split all the clusters, calculate BIC and then decide which model is the best as we do for merging. The different approaches used in increasing and reducing model complexity are due to the cost of computing the parameters of the Gaussians. In a merging operation calculating the parameters of the new cluster is easy and fast; in the opposite operation, finding the two new centroids is more difficult and expensive because some EM steps are needed.

4.3 Stopping the Adaptation

Although adapting the model structure and its parameters seems to be the definitive solution to color model tracking, using an adaptive color model could cause different problems, first of all due to the lack of ground-truth. Citing from [8]: "any color-based tracker can loose the object it is tracking, due, for example, to occlusion or object disappearing from the field of view".

As an object disappears its color component(s) are removed from the color space. The color model, i.e. its component(s), will try thus to fit image regions which do not correspond to the object. A method to detect such data associations problem and then to stop the adaptation of that component is therefore needed. This idea of stopping adaptation is adapted from the selective adaptation of McKenna et al. [8] (the pun is intentional). Moreover, it will not suffice to stop adaptation at all, but it will be necessary to detect this situation, and eventually to stop adaptation, selectively for each component.

Our proposal to circumvent this problem is based on the observed log-likelihood measurements. The EM algorithm maximizes the log-likelihood of the color data over time. Let $\mathcal{L}_i$ be the normalized log-likelihood for the i^{th} component, and $X^{(t)}$ be the data-set at frame t we have:

$$\mathcal{L}_i^{(t)} = \frac{1}{N^{(t)}} \sum_{\mathbf{x} \in X^{(t)}} \log p(\mathbf{x}|\theta_i) \tag{10}$$

At each frame t, $\mathcal{L}_i^{(t)}$ is evaluated, and this is done for each component. If the tracker looses the object(s) connected to the i^{th}-component, a large drop in the $\mathcal{L}_i^{(t)}$ will occur. The component adaptation is then suspended until the

model gains again a sufficient support from the data. In practice we have a threshold $T_i^{(t)}$ and adaptation is performed only if $\mathcal{L}_i^{(t)} > T_i^{(t)}$. The mean, $\nu_i^{(t)}$, and standard deviation, $\sigma_i^{(t)}$, of $\mathcal{L}_i^{(t)}$ are computed, in our experiments, for the n most recent above-threshold frames. The threshold is $T_i^{(t)} = \nu_i^{(t)} - k\sigma_i^{(t)}$, where k is a constant whose value has been experimentally set at about 1.5. The whole set of components, under online adaptation or not, are used for classification purposes instead.

5 Experiments on Real Data-sets

In the following we present some experiments in order to validate our proposal. These are on real data-sets, one source is the set of images[3] provided by the authors of [1], and the other source are video sequences grabbed in our laboratory. In some cases we use 2D plots to explain the results, but notice that the adaptation always took place in the 3D HSV color space. In this color space, data-points can translate or rotate. Translations are usually due to changes of the intensity of light, while rotations are due to changes in the color temperature. In all the experiments, the algorithm has been initialized by selecing a certain number of patches from the first frame in the sequence; the initial number of compents has been set to the number of patches selected and the initial mixture parameters has been estimated from the pixel patches.

The first example describes how the algorithm works with respect to changes of light conditions. Normally this is due to day/night cycle. Assuming slowly

[3] Available at `http://smart.informatik.uni-ulm.de/DataSets/RoboCup/natLight/`

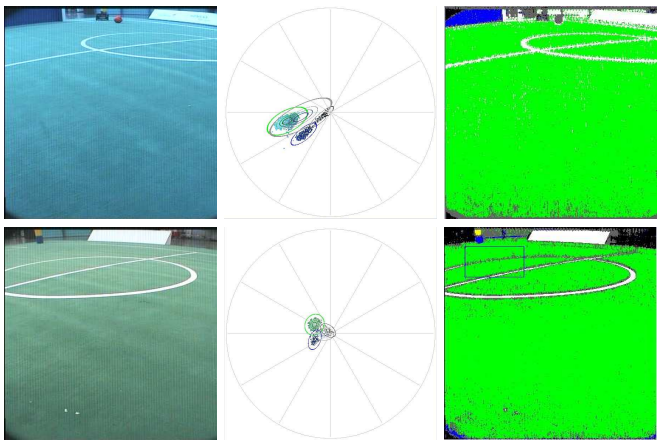


Fig. 5. The different color distribution obtained by adapting to the two images on the left, treated like they were consecutive. The upper image has been grabbed during natural day lights, the one underneath has been taken during evening with neon lights.

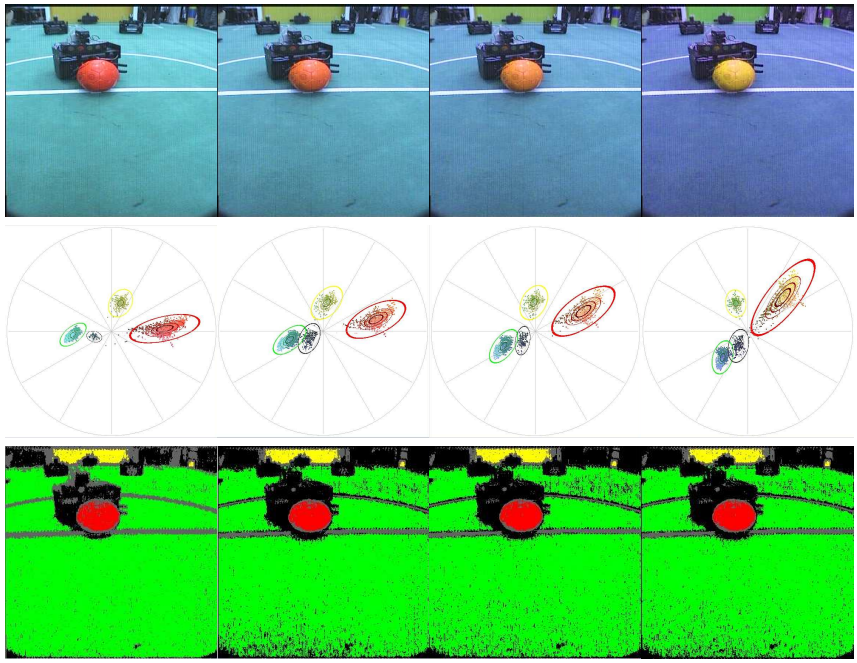


Fig. 6. The algorithm adapts the components to Hue variations, see in the central row how the color distribution rotates in HS plane, and how the image classification remains good. The images have been concatenated as if they were a sequence.

varying conditions, the model adapts and the variance of the components grows if the scene becomes brighter and decreases if becomes darker. Components with different labels will not be merged, so the real data distribution is well described. Hereafter we present the results obtained in a really extreme situation: an abrupt change of the light on the scene, from natural day light in one image, to artificial (neon) light at night in the other; the two were put in sequence and the results show the capability of the algorithm to adapt, see Figure 5. We think that such an abrupt change is a realistic estimate of what can be perceived by a generic indoor mobile robot. Therefore this confirms the capabilities of our system to make really more robust the perception system.

The second experiments validates the adaptation of the algorithm with respect to changes of the color temperature. We hand-manipulated some images, taken from the web-site mentioned before, changing the hue channel, see Figure 6, in order to mimic situations of natural light in outdoor environments. In such situations the light is not always white and changes with time; for example, in the morning it usually tends to move toward the blue, while in the evening tends to move toward the red. In such situations adaptation on the Hue component is really relevant. As it can be checked in the pictures, the algorithm allows

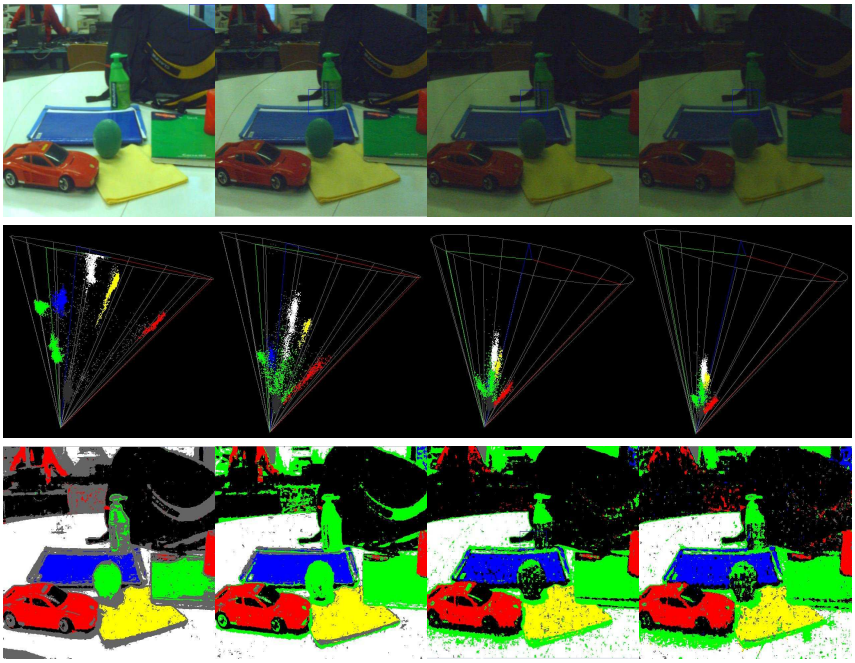


Fig. 7. From top to bottom: original images, HSV space, classified image. Images are taken from a 45 frames sequence grabbed in our lab.

a correct and robust image classification over time by adjusting the components to the real data distribution, rotating in the HS plane.

The last experiment presents a bright scene that becomes dark. The model adapts over time diminishing the number of cluster during the darkening phase, when the data-points concentrate in the bottom of the HSV cone. The experiment demonstrate a very good performance and only very few errors are due to mis-classification of noise, moreover this happened in very difficult, i.e. dark, conditions (Figure 7). When the scene will be brigh again the split algorithm will divide the components to keep-up with the number of clusters.

The algorithm for on-line color calibration presented has been developed to work in parallel with a complete 15fps color tracking system. The color look-up table used for classification is updated by the current (unoptimized) algorithm at 5-8 Hz being the only thread on a P4 at 2.4Ghz using a subsample (80×60) of the original image.

6 Conclusions

Soccer robots must be able to play under natural illumination that can change over time both in intensity and color temperature. To obtain good color clas-

sifications, two approaches are available: to adapt the color model or to use algorithms enforcing color constancy. The first approach is today the only one allowing performances compatible with an on-line real-time use. We started with a known approach to color modeling, based on EM maximum-likelihood iterative estimation. This known algorithm includes the possibility of stopping the EM-based adaptation; we adapted this algorithm to the multi-target domain required by Robocup real-robots league, especially for what concerns the stopping adaptation functionality; more important is the addition of on-line model order adaptation. This is a relevant issue in color modeling, especially for enabling its use in dynamical and difficult domains like the real-robots Robocup as well as indoor robotics. Our proposal produced quite good results, and results in quite extreme experiments validate such claim.

References

1. Mayer, G., Utz, H., Kraetzschmar, G.K.: Playing robot soccer under natural light: A case study. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 238 – 249
2. Cameron, D., Barnes, N.: Knowledge-based autonomous dynamic colour calibration. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 226 – 237
3. Jüngel, M., Hoffmann, J., Lötzsch, M.: A real-time auto-adjusting vision system for robotic soccer. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 214 – 225
4. Jamzad, M., Lamjiri, A.K.: An efficient need-based vision system in variable illumination environment of middle-size robocup. In Polani, D., Browning, B., Bonarini, A., eds.: RoboCup 2003 - Robot Soccer World Cup VII, Berlin, Springer-Verlag (2004) pp. 654 – 661
5. Forsyth, D.A.: A novel algorithm for color constancy. International Journal of Computer Vision **5** (1990) 5 – 36
6. Mayer, G., Utz, H., Kraetzschmar, G.K.: Towards autonomous vision self-calibration for soccer robots. In: Intl. Conference on Intelligent Robots and Systems (IROS02), Lausanne, Switzerland (October 2002) pp. 214 – 219
7. Swain, M.J., Ballard, D.H.: Color indexing. International Journal of Computer Vision **7** (1991) 11–32
8. McKenna, S.J., Raja, Y., Gong, S.: Tracking colour objects using adaptive mixture models. Image and Vision Computing **17** (1999) 225 – 231
9. Raja, Y., McKenna, S., Gong, S.: Tracking and segmenting people in varying lighting conditions using colour. In: Proceedings of FG'98, Nara,Japan (1998)
10. Wren, C., Azarbayejani, A., Darrel, T., Pentland, A.: Pfinder: Real-time tracking of the human body. IEEE Transaction on Pattern Anaysis and Machine Intelligence **9** (1997) 780–785
11. Hartley, H.: Maximum likelihood estimation from incomplete data. Biometrics **14** (1958) 174–194
12. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via em algorithm. Journal of the Royal Statistical Society, Series B **39** (1977) 1–38
13. Schwarz, G.: Estimating the dimension of a model. Ann. of Stat. **6** (1978) 461–464
14. Kass, R., Raftery, A.: Bayes factors. Journal of American. Stat. Ass. **90** (1995) 773–795