Automatic On-Line Color Calibration Using Class-Relative Color Spaces*

Pablo Guerrero, Javier Ruiz-del-Solar, Josué Fredes, and Rodrigo Palma-Amestoy

Department of Electrical Engineering, Universidad de Chile
{pguerrer,jruizd,jfredes,ropalma}@ing.uchile.cl

Abstract. In this article we present an automatic on-line color calibration system that makes extensive use of the spatial relationships between color classes in the color space. First, we introduce the definition of class-relative color spaces, where classes are represented in terms of their spatial relation to a base color class. Then, using class-relative color spaces, the system is able to remap classes from the already trained ones, which gives a starting point for training the remaining classes. The color-calibrating system also uses a feedback from the detected objects using the remapped (or partially trained) classes. As a result, the system is able to generate a complete color look-up table from scratch, and to adapt quickly to severe lighting condition changes. A particularity of our system is that it does not need to solve the natural ambiguity in color classes' intersections, but it is able to keep and use it during color segmentation using the concept of soft-colors.

1 Introduction

In the RoboCup Four Legged League, as in most of the RoboCup soccer leagues, objects are specifically colored to allow robots to recognize them easily. Most of the employed vision systems use color segmentation to take advantage of the color information. There several approaches for segmenting colors in real time [5], a common one is based in the use of a 3D look-up table (LUT). When the robot operates in a controlled environment having fixed lighting conditions, a fixed LUT performs well, and thus it can be trained off-line. However, this approach has two main flaws: a lot of time is needed for a human to calibrate the LUT, and the operation of the robot is strictly limited to artificial environments with highly controlled illumination. In this context, the development of automatic or adaptive color calibration systems has been intensely treated by the RoboCup community in the last years. The motivation is very clear: RoboCup is supposed to increasingly move to more realistic game conditions, which of course include natural lighting.

The presented work proposes an automatic on-line color calibration system, which allows the robot to build a LUT on-line, and to adapt it quickly to severe lighting condition changes. To our knowledge, the paper is innovative in three aspects: (i) spatial relationships between color classes are used in a very general fashion, which

* This research was partially supported by FONDECYT (Chile) under Project Number 1061158.

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 246–253, 2008.

allows the system to be easily adapted to any other application with different objects and/or color classes, (ii) the intersections of the color classes (also called *soft-colors* in the literature) are also automatically trained and stored in the LUT, and (iii) non isotropic illumination is considered, and an automatic training procedure is proposed.

2 Related Work

A main stage in any automatic color calibration system is the extraction of pixels of colored objects to train the LUT or the classes' statistics. To obtain these pixels, some of the published approaches relay on the knowledge of the objects' shape and on the pose of the robot, and thus, on the relative positions of the fixed objects with respect to the robot [6][2]. Some other systems use scan lines and predefined transition rules based on simple spatial relations between color classes in the color space (for example: "cyan has a higher U component than green") [1]. Other systems use a priori membership distribution to track the classes' statistics by means of the EM algorithm [3]. Some of the proposed approaches make use of incremental layers or estimations of the color classes [1], where coarse layers are used to extract pixels that are used to train more precise layers. Regarding color information representation, there are several ways to represent color classes. The following are examples of proposed class representations, sorted by complexity and flexibility: cuboids [1], non-rotated ellipsoids (mean and uncorrelated variances of color components) [6], union of rotated ellipsoids (Gaussian Mixture) [3], and hybrids that bounds in different ways the different color coordinates [4]. Most of them do not allow intersection between classes so ambiguity must be solved before filling the LUT, attempting to minimize the expected classification error. Additionally, color constancy approaches (e.g. [8]) propose the existence of transformations or mappings, in a determined color space, that describe what happen to colors of an image when the lighting changes. If one applies such an approach literally to the color segmentation problem, one could preprocess each image, and get a transformed one that should be easily segmented with a previous LUT. But, this transformation should be applied to each pixel, which is a prohibitive task in real-time robotics. However, we are using the color space mapping idea to propose the remapping applied to color classes instead of pixels, which is a task that can be performed in real time.

3 Proposed Approach

The proposed automatic color calibration system starts its operation with the extraction of pixels corresponding to color classes that can be trained with a total lack of a priori knowledge. These color classes are green and white in our application (in the RoboCup soccer environment the field's lines and carpet can be detected without using color information), but from a more general point of view, they can be colors of any objects that can be detected without use of color information. The extracted classes' statistics are used in combination with a priori knowledge of the spatial relationships among the color classes to remap the rest of the classes. This remapped color classes are then used for the on-line detection of objects. Then, the system takes

feedback from the detected objects to extract pixels of the respective classes, and makes a smooth transition from the *remapped estimations* of the color classes to *trained estimations* of the classes.

3.1 Basic Definitions: Colors, Color Classes and Color Classes Representation

The system works in the YUV color space because the AIBO camera takes images in this format. A point in the YUV color space will be named a *color*. A *color class* is a set of colors that can be observed in pixels corresponding to an object or an object part having a given human-defined color. For example the class "yellow" is the one that contains pixels belonging to a yellow goal or a yellow part of a beacon. The set of color classes Ω is defined by the application. As discussed in [6] (even when they chose a simpler representation), we have found that a correlated 3D Gaussian is enough to represent a color class. Thus, we have chosen to represent each class $\mathbf{K} \in \Omega$ by a mean and a covariance in the YUV space, $(\boldsymbol{\mu}_\mathbf{K}, \boldsymbol{\Sigma}_\mathbf{K})$. An *innovation threshold* $\lambda_\mathbf{K}$ is used to determine when a color belongs to any class $\mathbf{K}$:

$$\mathbf{K} = \left\{ \mathbf{c} \in [0,255]^3 \,/\, (\mathbf{c}-\boldsymbol{\mu}_\mathbf{K})^T \, \boldsymbol{\Sigma}_\mathbf{K}^{-1} (\mathbf{c}-\boldsymbol{\mu}_\mathbf{K}) < \lambda_\mathbf{K} \right\} \tag{1}$$

Note that the size of a class is determined by its covariance matrix $\boldsymbol{\Sigma}_\mathbf{K}$ and its innovation threshold $\lambda_\mathbf{K}$. This class representation corresponds to an ellipsoid in the YUV space with possibly rotated axes and different radiuses. A value of $\lambda_\mathbf{K} = 10$ is found to be optimal when the class statistics are reasonably well estimated.

When lighting conditions change, color classes change their position and size in the color space. This makes a fixed color's LUT inapplicable in those situations. However, even when the lighting condition change drastically, and the color classes suffer severe modifications, some spatial relationships between them in the color space remain unaltered. Thus, given a color class $\mathbf{K}$, we can define a ***K**-relative color space* as one centered in $\boldsymbol{\mu}_\mathbf{K}$ and with a metric linearly transformed by a function of $\boldsymbol{\Sigma}_\mathbf{K}$. Any color $\mathbf{c}$ can be transformed to the **K**-relative color space:

$$\mathbf{c}^\mathbf{K} = \sqrt{\Sigma_\mathbf{K}^{-1}} \, (\mathbf{c} - \boldsymbol{\mu}_\mathbf{K}) \tag{2}$$

Where the square root of a matrix $\mathbf{A}$ is defined as a lower triangular matrix that satisfies: $\mathbf{A} = \sqrt{\mathbf{A}} \sqrt{\mathbf{A}}^T$. The square root is implemented using the Cholesky factorization [9]. We call $\mathbf{c}^\mathbf{K}$ the ***K**-relative representation* of $\mathbf{c}$. Analogously, given any color class $\mathbf{D}$, with mean and covariance $(\boldsymbol{\mu}_\mathbf{D}, \boldsymbol{\Sigma}_\mathbf{D})$, it can have its **K**-relative representation defined as $(\boldsymbol{\mu}_\mathbf{D}^\mathbf{K}, \boldsymbol{\Sigma}_\mathbf{D}^\mathbf{K})$, where,

$$\boldsymbol{\mu}_\mathbf{D}^\mathbf{K} = \sqrt{\Sigma_\mathbf{K}^{-1}} \, (\boldsymbol{\mu}_\mathbf{D} - \boldsymbol{\mu}_\mathbf{K}) \,; \; \Sigma_\mathbf{D}^\mathbf{K} = \sqrt{\Sigma_\mathbf{K}^{-1}} \, \Sigma_\mathbf{D} \sqrt{\Sigma_\mathbf{K}^{-1}}^T \tag{3}$$

Note that in particular, $(\boldsymbol{\mu}_\mathbf{K}^\mathbf{K}, \boldsymbol{\Sigma}_\mathbf{K}^\mathbf{K}) = (0, \mathbf{I})$, with $\mathbf{I}$ the 3x3 identity matrix.

3.2 Off-Line Training

In the proposed system color classes are trained manually using a procedure as the one described in [7]. The statistics $\left(\boldsymbol{\mu}_{\mathbf{D}}^{\mathbf{K}}, \boldsymbol{\Sigma}_{\mathbf{D}}^{\mathbf{K}}\right)$ are calculated and stored for every pair of classes $(\mathbf{K}, \mathbf{D}) \in \Omega^2$. This procedure is intended so that the system learns the spatial relationships between classes, and it is needed to be carried out it only once for a determined set of color classes. This is why we associate this procedure to the one when a human learns the colors for the first time. We have found that the system is robust enough to small changes in the actual colors of the objects (for example, the color of the carpet).

3.3 On-Line Operation

Our system maintains two estimations of the color classes, a *remapped estimation* and *trained estimation* (see explanation in the next sections). These two estimations are combined to obtain the resulting estimation that is used to fill a LUT. This resulting LUT is the output of the system. Fig. 1 shows the system's components and the information flow. In the next sections the different modules will be explained.

Pixels Extraction and Statistics Calculation: In this stage, acquired images are used to extract pixels from detected objects. A fixed maximum number of pixels colors are stored for each color class. The number of stored colors is selected to ensure that enough images are considered (approximately 10 images, with a mean number of extracted pixels per image of ~200 green pixels and ~40 for the rest of the classes). When necessary, oldest colors are rewritten by the newest ones. Green and white are extracted using scan lines. Scan lines are perpendicular to the horizon line and the scan is performed similarly as described in [1], but following upwards direction. For the sake of brevity, we will not describe in detail this procedure since it is not the focus of the paper. When using this procedure, it is not necessary to have a priori knowledge about the lighting conditions to extract green and white pixels because the visual sonar is based on Y channel transitions, thus we call green and white *self-sufficient classes*. This is why the visual sonar is the starting point of the system. Yellow, cyan, pink, orange, red and blue (or other colors in the case of applications different than RoboCup soccer) are extracted from detected objects having the corresponding color class. Of course, to detect these objects it is necessary to have a previous estimation of those color classes, which is not possible when the system starts or when the lighting conditions change. That is why we call these classes *dependent*.

Pixels selected to train a class are filtered (using (1), but considering a higher innovation threshold) according to their innovation with respect to the resulting estimation, to prevent outliers from damaging classes' statistics. Trained classes' statistics are recalculated after each image is processed using the stored colors. This recalculation is implemented in an efficient incremental fashion.

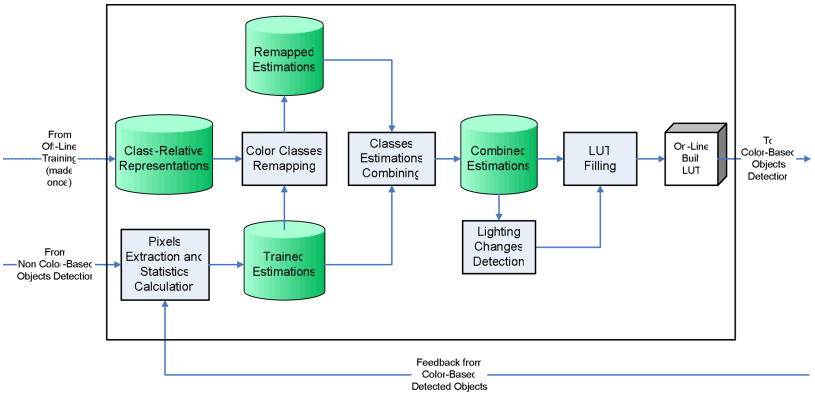

Fig. 1. Block diagram of the system. Trained and remapped color classes estimations are combined to get a resulting estimation which is used to fill the resulting LUT. The system is able to completely train and adapt the resulting LUT having prior knowledge of the spatial relationships between colors.

Color Classes Remapping: As discussed in [8], a linear mapping is not enough to cope with the possible color space transformations that may appear when lighting changes, but, one could locally approximate such a mapping with a linear transformation. We present a statistic method for remapping the non-trained color classes. To overcome the lack of extracted pixels for the dependent classes, we make use of the class-relative color spaces to create a first approximation of them. If any class **K** is already trained, a remapped estimation of any other class **D** can be obtained from its **K**-relative representation and the **K** trained estimation:

$$\boldsymbol{\mu}_{\mathbf{D},r}^{\mathbf{K}} = \boldsymbol{\mu}_{\mathbf{K},t} + \sqrt{\Sigma_{\mathbf{K},t}}\,\boldsymbol{\mu}_{\mathbf{D}}^{\mathbf{K}}\ ;\ \Sigma_{\mathbf{D},r}^{\mathbf{K}} = \sqrt{\Sigma_{\mathbf{K},t}}\,\Sigma_{\mathbf{D}}^{\mathbf{K}}\sqrt{\Sigma_{\mathbf{K},t}}^{T} \tag{4}$$

When there is more than one trained class, the system uses *remapping weights* $\beta_{\mathbf{D}}^{\mathbf{K}}$ to determine how relatively important is the **K**-relative color space to remap the class **D**. Remapping weights are calculated as:

$$\beta_{\mathbf{D}}^{\mathbf{K}} = \frac{\alpha_{\mathbf{K}} \left\| \boldsymbol{\mu}_{\mathbf{D}}^{\mathbf{K}} \right\|^{-1}}{\sum_{\mathbf{K}' \in \Omega/\{\mathbf{D}\}} \alpha_{\mathbf{K}'} \left\| \boldsymbol{\mu}_{\mathbf{D}}^{\mathbf{K}'} \right\|^{-1}}\ ;\ \alpha_{\mathbf{K}} = \frac{N_{\mathbf{K}}}{T_{\mathbf{K}}} \tag{5}$$

Where $T_{\mathbf{K}}$ is the maximum number of extracted pixels for the class **K**. Then, the remapped estimation of **D** is calculated as:

$$\boldsymbol{\mu}_{\mathbf{D},r} = \sum_{\mathbf{K}' \in \Omega/\{\mathbf{D}\}} \beta_{\mathbf{D}}^{\mathbf{K}'} \boldsymbol{\mu}_{\mathbf{D},r}^{\mathbf{K}'}\ ;\ \boldsymbol{\Sigma}_{\mathbf{D},r} = \sum_{\mathbf{K}' \in \Omega/\{\mathbf{D}\}} \beta_{\mathbf{D}}^{\mathbf{K}'} \boldsymbol{\Sigma}_{\mathbf{D},r}^{\mathbf{K}'} \tag{6}$$

Every class is remapped in function of the existing trained colors. This remapped estimation is stored to be combined with the trained estimation, if it already exists.

Classes Estimations Combining: A linear combination of remapped and trained estimations is used to get the resulting color class estimation:

$$\boldsymbol{\mu}_{\mathbf{D}} = \alpha_{\mathbf{D}}\boldsymbol{\mu}_{\mathbf{D},t} + (1-\alpha_{\mathbf{D}})\boldsymbol{\mu}_{\mathbf{D},r}\,;\ \boldsymbol{\Sigma}_{\mathbf{D}} = \alpha\boldsymbol{\Sigma}_{\mathbf{D},t} + (1-\alpha)\boldsymbol{\Sigma}_{\mathbf{D},r} \tag{7}$$

The use of $\alpha_{\mathbf{D}}$ allows a smooth transition from the use of the remapped estimation of the class (when no pixel has yet been trained) to the use of the trained estimation (when the maximum number of pixels has already been trained). This smooth transition has the objective of avoiding mistakes in the association of training pixels to partially trained classes.

LUT Filling: The LUT is filled when any of the classes' resulting estimation moves enough, from the used to build the current LUT, to make it obsolete. The LUT filling is efficiently implemented: For each pair (Y,U), the two solutions V_1 and V_2 of the quadratic equation $(\mathbf{c}-\boldsymbol{\mu}_{\mathbf{K}})^T\,\boldsymbol{\Sigma}_{\mathbf{K}}^{-1}(\mathbf{c}-\boldsymbol{\mu}_{\mathbf{K}}) = \lambda_{\mathbf{K}}$ are calculated, with $\mathbf{c} = (Y,U,V)$. If $V_1, V_2 \in \mathbb{R}$, the LUT is filled in the (Y,U) row, from V_1 to V_2 with class $\mathbf{K}$.

4 Results

We have tested our autonomous calibration system in real AIBO image sequences, with both the robot and its camera moving and partially controlled lighting conditions. To illustrate how the system creates a new LUT from scratch, figure 2 shows important events in the color calibration process, and how the segmentation improves as new images are processed. The whole sequence corresponds to a half turn of the robot around itself (~2 sec). From testing the system in several image sequences as the shown in fig. 2, we have concluded that the system is able to completely train a LUT from scratch.

We compare the performance of the proposed method with Adaptive Color Distribution Transformation (ACDT) [10]. Fig. 3 shows the evolution of the *correctly classified pixel rate* (CCPR) over an image sequence[1]. The CCPR corresponds to the rate of pixels correctly classified inside the regions of the image occupied by actual objects. As can be seen from the curves, the system performs very similar to ACDT (CCPR≈40%) when the off line stage was trained in a different environment (UChile Lab). When the offline stage is performed in the same environment, the performance of the system is noticeably superior (CCPR≈55%).

Processing time is a very relevant issue in mobile robotics systems, and even more when having limited processing power. Thus, we limit the frequency in which each of the operations is executed. This limitation is flexible and it is possible to balance the reactivity of the system versus the demanded processing time.

[1] The image sequence and its correspondent ground truth information was downloaded from http://www.dis.uniroma1.it/~spqr/cms/

Fig. 2. Example pictures from a video sequence obtained while the robot is making calibration from scratch (above), and the correspondent segmented images using the LUT obtained up to that moment (bellow). Some relevant events are the first detections of: the blue goal (left), the ball (center), and the pink yellow beacon (right).

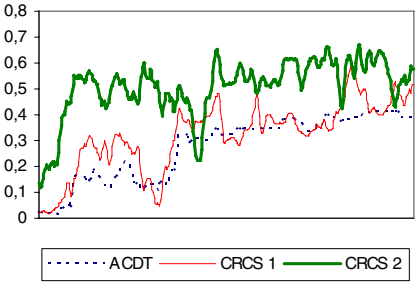

Fig. 3. Correctly Classified Pixel Rate evolution over a 394 image sequence for: Adaptive Color Distribution Transformation (ACDT), Class-Relative Color Spaces with an off-line stage in a different environment, with different illumination (CRCS1), and Class-Relative Color Spaces with an off-line stage in the same environment, with the same illumination (CRCS2)

Table 1. Processing times for each stage of the system

Stage	Frequency	Mean processing time (AIBO)
Training Classes	10Hz	6ms
Remapping Classes	0.5Hz	1.6ms
Combining Estimations	0.5Hz	0.5ms
Filling the LUT	~0.1Hz	26ms
Total time	1Hz	70ms

To convince the reader that the system is able to work on-line, table 1 shows the processing time consumed by each of the stages of the process and the frequency in which each of them is performed. The total time is presented with a frequency of 1Hz because the operations are not performed at the same time, so the presented total time is a mean over 1 second period. The presented processing times are measured in an AIBO CPU (64bit RISC, 576 MHz, Aperios). It is important to note that the system can be executed in real time over an AIBO CPU because, if necessary, some frequencies could

be further reduced without a noticeable impact on the performance of the system, assuming that the lighting conditions will not change too often. With no frequency limitations, the entire process takes approximately 35ms, which is not good enough to play soccer but allows the robot to get a good LUT as quick as possible.

5 Conclusions

We have presented a novel approach for automatic calibration of a color segmentation system. Although the system is applied for a specific RoboCup soccer league, the presented framework is general enough to be used in other soccer leagues, and in other applications having any reasonable set of color labels. As is shown in the results section, the system is able to work online and to completely train a LUT from scratch. However, there are several efficiency improvements that may be achieved as, for example, to perform the LUT filling only for the needed classes. Also, we are planning to make our software architecture disconnect the automatic color calibration when time demanding tasks, as pursuing the ball, are being performed.

Acknowledgments. We would like to thank Luca Iocchi for kindly providing us the image sequence and the code for testing it with his method.

References

1. Jüngel, M.: Using Layered Color Precision for a Self-Calibrating Vision System. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 209–220. Springer, Heidelberg (2005)
2. Heinemann, P., Sehnke, F., Streichert, F., Zell, A.: Towards a Calibration-Free Robot: The ACT Algorithm for Automatic Online Color Training. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434. Springer, Heidelberg (2007)
3. Anzani, F., Bosisio, D., Matteucci, M., Sorrenti, D.: On-Line Color Calibration in Non-Stationary Environments. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 396–407. Springer, Heidelberg (2006)
4. Gönner, C., Rous, M., Kraiss, K.: Real-Time Adaptive Colour Segmentation for the RoboCup Middle Size League. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 402–409. Springer, Heidelberg (2005)
5. Bruce, J., Balch, T., Veloso, M.: Fast and Inexpensive Color Image Segmentation for Interactive Robots. In: Proceedings of the 2000 IEEE/RSJ International Conference on Inteligent Robots and Systems (IROS 2000), vol. 3, pp. 116–122 (2000)
6. Sridharan, M., Stone, P.: Towards Eliminating Manual Color Calibration at RoboCup. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 673–681. Springer, Heidelberg (2006)
7. Palma, R., Guerrero, P., Ruiz del Solar, J.: Context-Dependent Color Segmentation for AIBO Robots. In: 3rd IEEE Latin American Robotics Symposium – LARS 2006 (CD Proceedings), Santiago, Chile, October 26 - 27 (2006)
8. Tieu, K., Miller, E.: Unsupervised Color Constancy. Advances in Neural Information Processing Systems (2002)
9. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C, 2nd edn. Cambridge University Press (1992)
10. Iocchi, L.: Robust color segmentation through adaptive color distribution transformation. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 287–295. Springer, Heidelberg (2007)