k-평균 클러스터링과 L*a*b* 칼라 모델에 의한 칼라코드 분류

Classifying Color Codes Via *k*-Mean Clustering and L*a*b* Color Model

유현중
상명대학교 정보통신공학과

Hyeon-Joong Yoo(yoohj@smu.ac.kr)

요약

칼라 식별에 대한 칼라 왜곡 영향을 줄이려면 각 칼라 영역에서 가능한 한 많은 화소를 통계적으로 처리하는 게 바람직하다. 여기에는 영역 분할이 필요하며, 따라서 일반적으로 에지 검출이 필요하다. 그러나, 칼라 코드의 에지들은 암전류, 색 간섭, 지퍼 효과, 반사, 그늘 등의 수많은 왜곡에 의해 끊기기 때문에 흔히 영역 분할이 불완전하게 되며, 그에 대한 에지 연결 작업도 쉽지가 않다. 이 논문에서는 에지 검출로 영역 분할을 할 수 없는 영상들에 대해 k-평균 클러스터링을 수행한다. 서로 다른 카메라로 서로 다른 환경에서 촬영된 311개의 영상에 대해 실험을 수행하였다. 일차 및 이차 칼라들 중에서 랜덤하게 선택해서 각 칼라 코드 영역에 사용하였다. 두 가지 에지 검출기들에 의한 영역 분할률은 89.4%였으며, 제안된 방법은 이를 99.4%로 증가시켰다. 칼라 인식은 hue, a*, b*의 세 성분들에 기반해서 수행되었으며, 성공적 영역 분할 경우들에 대해 100%의 정확도를 보였다.

■ 중심어 : | 칼라 영역 분할 | k-평균 클러스터링 | CIE L*a*b* |

Abstract

To reduce the effect of color distortions on reading colors, it is more desirable to statistically process as many pixels in the individual color region as possible. This process may require segmentation, which usually requires edge detection. However, edges in color codes can be disconnected due to various distortions such as dark current, color cross, zipper effect, shade and reflection, to name a few. Edge linking is also a difficult process. In this paper, k-means clustering was performed on the images where edge detectors failed segmentation. Experiments were conducted on 311 images taken in different environments with different cameras. The primary and secondary colors were randomly selected for each color code region. While segmentation rate by edge detectors was 89.4%, the proposed method increased it to 99.4%. Color recognition was performed based on hue, a*, and b* components, with the accuracy of 100% for the successfully segmented cases.

■ keyword : | Color Segmentation | k-Means Clustering | CIE L*a*b* |

I. Introduction

1. Uses of color codes

While color codes and RFID, which is receiving growing attention with the increasing interest in ubiquitous computing, share some applications, color codes have additional applications that cannot be addressed with RFID. Color codes are also more

접수번호 : #061128-002
접수일자 : 2006년 11월 28일
심사완료일 : 2007년 01월 19일
교신저자 : 유현중, e-mail : yoohj@smu.ac.kr

economical than RFID, and, furthermore, need no special device to capture images; they can be captured by standard camera-equipped cellular phones or PC cameras. Furthermore, the RFID is susceptible to fraud because RF signals are accessible to everyone and are invisible. [Table 1] summarizes a comparison between the RFID and color codes.

Table 1. Comparison of the RFID and color codes

	passive/active RFID	color code
Simultaneous multiple asset scanning	O/O	O
Tag cost of less than 5 cents		O
Long distance tag reading	O	O
Pin-point indication of non-tagged and non-read assets		O
Unaffected by RF interference		O
Works with metals / moisture present	O	O
No line-of-sight requirement	O/O	

[Fig. 1] shows the typical color codes, and [Fig. 2] shows some examples of their applications. In the application shown in [Fig. 2(a)], pictures from an event are uploaded to a website. Visitors to the event are able to access their pictures and give orders for having them printed. [Fig. 2(b)] shows another application in which color codes are marked on, e.g., news papers, magazines, and business cards. The images can be taken by using digital cameras, which can be directly linked to a related database, or other services.

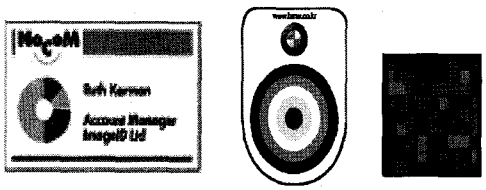
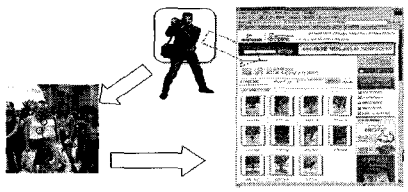


Fig. 1. Typical color codes



(a) Application 1

(b) Application 2

Fig. 2. Examples of color code applications

2. Need for Research of Color Code Recognition

It is quite common to find a distortion of RGB component values from color images exceeding 50% of the whole scale. This could be viewed as the main reason why bar codes have not yet been replaced by color codes, and further, explain why research on the subject of color code recognition for which the precise color information is required is rare [1], while extensive research exists on using rough information of colors to find certain objects like face [2-4].

[Fig. 3] shows color code tags obscured by reflection, demonstrating that the traditional error checking or error correction techniques may not work for color codes. There is also an instrumental distortion. Single-sensor digital cameras spatially sample the incoming image using a color filter array (CFA) [5]; consequently, each pixel only contains a single color value. In order to reconstruct the original full-color image, a demosaicking step must be performed which interpolates the missing color of each pixel. Demosaicking algorithms usually do well for color fidelity, but there is often a tradeoff between a sharp image and the so-called "zipper-effect" or jagged edge look. This, together with the above-mentioned, suggests that segmentation techniques resorting to other than edge detection are

required.

Section II explains experiments and results. Section III analyzes and discusses the findings of the experiment. Section IV offers some conclusions and makes suggestions for further research.

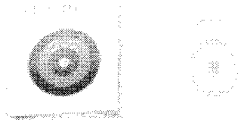


Fig. 3. Examples of color codes obscured by reflection

II. Detection, Segmentation and Recognition of Color Codes

The circularly symmetric design of [Fig. 1(b)], which consists of four concentric rings and one central circular plate, was used in experiments. The color for each ring was randomly chosen from R, Y, G, C, B, and M. These colors theoretically are equally spaced by 60° in hue domain and have values of 1.0 in saturation domain. The rings had the same thickness, and the diameter of central circular plate was 1.5 times the thickness to take into account color interference from neighboring color regions.

Experiments were conducted on three sets of outdoor images of the sizes of 4M (2464x1632), 4.3M (2544x1696), or 7.1M (3264x2176) pixels taken in different environments with Canon and Nikon cameras. The image sets consist of 144, 140, and 27 images, respectively. Each color ring had at least 7 pixels of thickness. In the images, the size of color code regions was 90x90 pixels on average (σ = 10.9), which resulted in around 9.5 pixels of thickness per color ring; the minimum size was 72x73, in which case each color ring had the thickness of 7.6 pixels on average. Color codes were printed on high quality commercial photo papers using an enterprise Fuji photoprinter.

1. Detecting Code Area

1.1 Dichotomizing Pixels in RGB Space

Code area detection was performed in RGB image and hue- and saturation-component images, using the features of the primary and secondary colors in RGB and HSI color models. In RGB space, R-, G-, and B-component values of each pixel were compared to determine if its color belonged to the set of the primary and secondary colors. The three component values were sorted first, and then the differences between the median value and the maximum and minimum values were computed. If the absolute difference between the two differences was high enough, the pixel was classified into the set of primary and secondary colors. Otherwise, pixels were classified into the set of grays. Thus the resulting image was binary. The algorithm is as follows:

begin initialize $x \leftarrow 0$, $y \leftarrow 0$, α, β, Δ
do $x \leftarrow x + 1$, $y \leftarrow y + 1$
if $\max(r, g, b) > \alpha$ **AND** $\min(r, g, b) < \beta$ **AND**
$|\max(r, g, b) - 2 * \mathrm{med}(r, g, b) + \min(r, g, b)|$
$> \Delta$
then $f(x, y) \in$ primary/secondary colors class
else $f(x, y) \in$ grays class
until $x = m$, $y = n$
end

Algorithm 1. The dichotomization in RGB space

The primary and secondary colors have the maximum value 255 for either one or two components; and either two or one components have the minimum value 0. Hence the line 3 in algorithm 1 detects pixels of the six colors. The criterion in line 4 uses the facts that the difference between the maximum- and median-component values must be large while the difference between median- and minimum-component values must be small for the primary colors. The similar criterion is also useful for

detecting the secondary colors.

1.2 Thresholding Hue- and Saturation-Component Images

Since the hue values of the six colors are multiples of 60, pixel values in hue-component image were converted using Eq. (1) in order to emphasize them.

$$hue(x,y) = |30 - hue(x,y) \ \% \ 60| \quad (1)$$

Then the threshold values for the transformed hue-component image and saturation-component image were computed using their average values.

1.3 Screening Candidate Areas

The three resulting binary images obtained above were logically AND-ed. The resulting binary image obtained through the initial detection phase above usually contained a number of candidate areas. To remove unlikely candidate areas, morphological operations and labeling [6] were performed on the binary image. The labeling must be significantly faster after morphological operations.

After labeling, unlikely objects were removed based on such criteria as size, shape, and area so that only one candidate remained. Specifically, for each object, the ratioe of its size to image size (Eq. (2)), its height (h_c) to width (w_c) (Eq. (3)), its area (S_c) to the area of its bounding rectangle (Eq. (4)), and its area to the corresponding circle's area (Eq. (5)) and the mean hue (Eq. (6)) and saturation (Eq. (7)) values were computed.

$$\alpha_1 < \min(h_c, w_c)/\min(h, w) \le \alpha_2 \ (<< 1) \quad (2)$$

$$\rho_1 < \min(h_c, w_c)/\max(h_c, w_c) \ (\le 1) \quad (3)$$

$$\rho_2 < S_c/(h_c * w_c) < \rho_3 \ \text{ for } \rho_3 < 1 \quad (4)$$

$$\rho_4 < \min(S_c, \pi * h_c * w_c/4)/\max(S_c, \pi * h_c * w_c/4) \le \rho_5 \ (< 1) \quad (5)$$

$$\overline{hue} = \frac{1}{N}\sum_{(x,y)\in R} hue(x,y) \quad (6)$$

$$\overline{sat} = \frac{1}{N}\sum_{(x,y)\in R} sat(x,y) \quad (7)$$

In Eqs. (6) and (7), R stands for the region of an object, and N the number of pixels in R. The criterion of Eq. (2) removes unreasonably large or small candidate areas. Eq. (3) removes candidates if their bounding rectangles were not close to square. The criteria of Eqs. (4) and (5) roughly check circularity. The values from Eqs. (3) and (5) - (7) can be used as memberships to construct the following objective function, which is useful when there exist more than one final candidate areas.

$$g = \alpha_1 * \min(h_c, w_c)/\max(h_c, w_c) + \alpha_2 * \min(S_c, \pi * h_c * w_c/4)/\max(S_c, \pi * h_c * w_c/4) + \alpha_3 * \overline{hue} + \alpha_4 * \overline{sat} \quad (8)$$

In this equation, the fist two terms use the features of shape, and the other two terms use the features of the colors. The coefficients are determined empirically. [Fig. 4] summarizes the prepocessing.

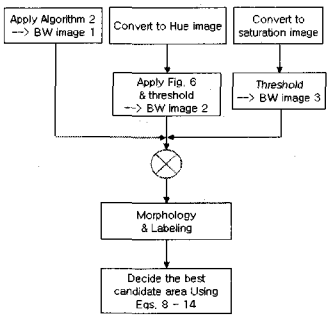


Fig. 4. The summary of the preprocessing

In this way we could successfully extract code regions from 310 images out of 311 images. [Fig. 5] shows the case in which the extractions failed, where severe reflection existed, and the result. There were also quite a few cases where the extractions were inaccurate. [Fig. 6] shows some of them (bottom row)

with their original tag images (upper row). For the first image of [Fig. 6] shadow caused unnecessary part to be detected together. For the rest, out-of-focus or reflection problems caused some part to be lost.

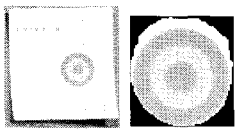


Fig. 5. The case of extraction failure

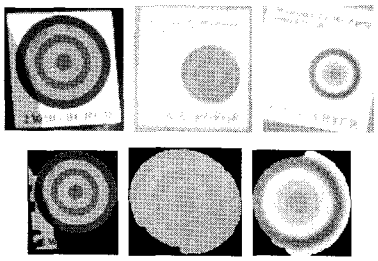
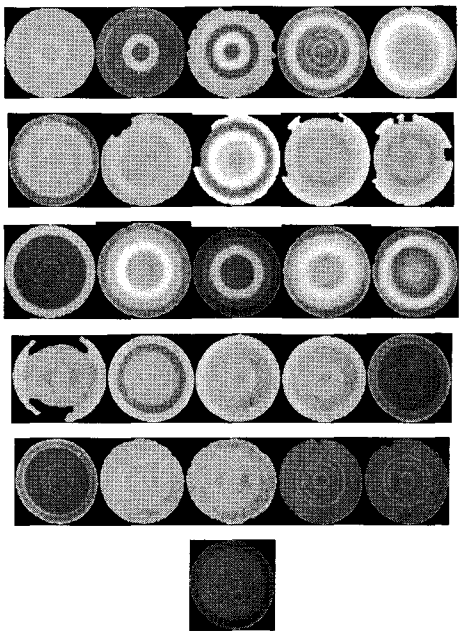


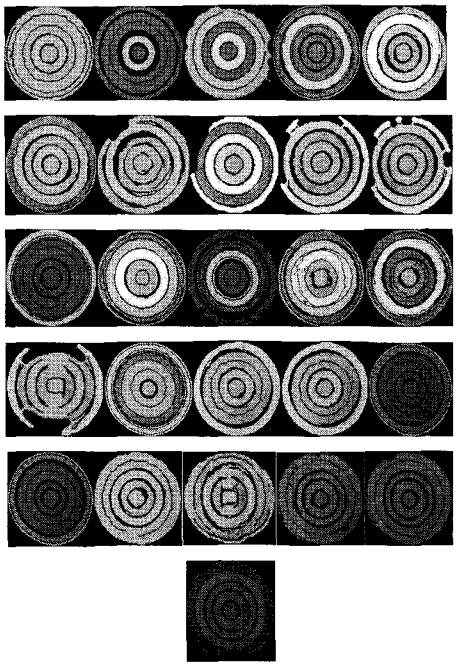
Fig. 6. Examples of inaccurate extraction

2. Segmenting Code Region Via Edge Detection or *k*-Means Clustering

For the final candidate code region, edges were detected using canny and sobel edge detectors [7]. Then the resulting edge images were combined. If the edge detection failed to segment color regions, *k*-means clustering would be used. For *k*-means clustering, five samples were randomly chosen from code region to serve as initial code vectors. [Fig. 7] shows 26 code images that could not be successfully segmented by the edge detection, but that had to be segmented by *k*-means clustering algorithm; the images in [Fig. 7(b)] show code images with disconnected edges.



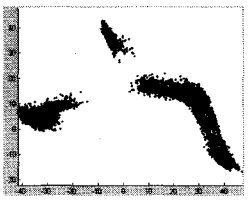
(a) Extracted code region



(b) Detected edges

Fig. 7. Cases segmented by *k*-means clustering

[Fig. 8(a)] shows the a*b* distribution of pixels in the 2nd code image of [Fig. 7]. The five code vectors trained by k-means clustering algorithm are indicated on the distribution as red circles. The true code values were GRMYR, and the loci indicate that each of R, M, and Y colors were represented by one code vector and G color by two code vectors. [Fig. 8(b)] shows segments corresponding to the five code vectors.

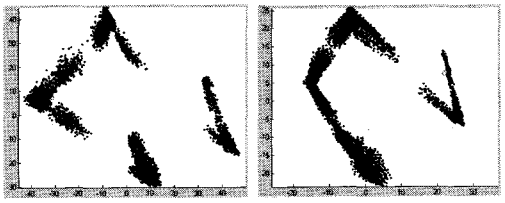


(a) a*b* distribution

(b) Separated segments

Fig. 8. Segmentation result with k-means clustering algorithm. (a) Color distribution in a*b* plane and trained code vectors. (b) Segments corresponding to the code vectors

In [Fig. 7] there are some code images that were over-segmented by edge detection: that is, in some of which exist double edges, and in some of which is a ring separated. k-means algorithm was also successful for such cases. There was only one case in which k-means clustering did not work either -- 13th code in [Fig. 7]. For the code, one code vector was assigned to R and M colors as shown in [Fig. 9(a)], while B was assigned two code vectors. That might be caused by the facts that the number of R pixels was much smaller than that of B color (even together with M pixels), initial code vectors were inappropriate, or the distribution was disconnected. [Fig. 9(b)] shows the distributions from a code image with the same code (12th code image in [Fig. 7]), but k-means clustering worked for it by assigning two code vectors to R-M region. The mean shift algorithm [8] may help resolve this problem. Consequently, by employing k-means clustering algorithm we could increase the segmentation rate from 89.4% (= 278/311) to 99.4% (= 309/311).



(a) (b)

Fig. 9. The color distributions and trained code vectors for two code images with the same value (13th and 12th in [Fig. 7]), for which k-means clustering (a) failed and (b) succeeded, respectively

3. Feature Vector and Decision Boundaries

[Fig. 10(a)] shows the distribution of peak hue values, h_p, and the differences among R-, G-, and B-component means, d_m, from 1390 color regions. h_p and d_m are defined as follows:

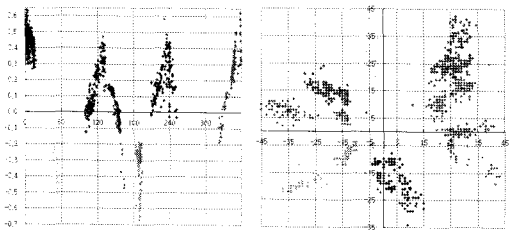
$$h_p = \{hue \mid \arg\max_{hue} f(hue)\} \quad (9)$$

where f stands for hue histogram, and

$$d_m = \max(r',g',b') - 2 * \text{median}(r',g',b') + \min(r',g',b') \quad (10)$$

where r', g', and b' stand for the mean values of R, G, and B components in a color region, respectively. Hence, d_m is supposed to be high positive for the primary colors and low negative for the secondary colors. We can see that R and M were overlapped. We found that peak h_p values were more

useful than mean values.



(a) (b)

Fig. 10. Color distributions in (a) d_m–h_p and (b) a*b* planes obtained from 1390 color regions

[Fig. 10(b)] shows the a*b* distribution of colors obtained from 1390 color regions. Hence, in a*b* plane, we could obtain error-free decision boundaries. However, for convenience, colors are first dichotomized into R or M and the other colors based on the d_m–h_p distribution. [Fig. 11] shows the decision tree. Then R and M were discriminated in a*b* plane while the other colors were discriminated based on peak hue.

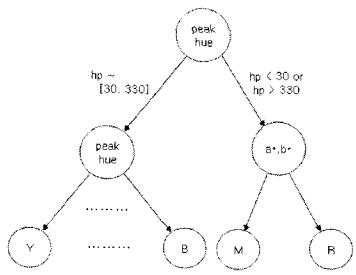


Fig. 11. Decision tree using h_p and (a*, b*) coordinates for color classification.

4. Recognizing Colors

Before analyzing colors, noise (thin or small) segments were excluded using the equation below.

$$S_i > \pi * \delta * (2\ r - \delta) \qquad (11)$$

where

$$r = (h_i + w_i)/2$$

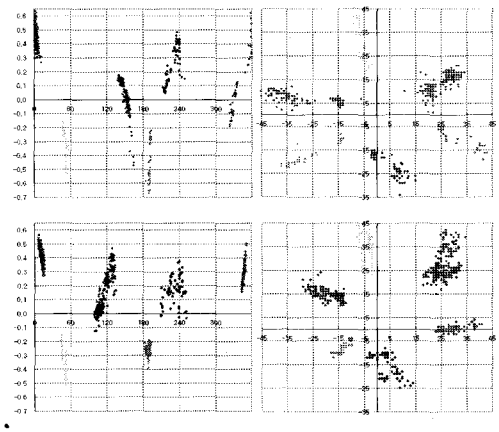
where h_i and w_i stand for the height and width of i^{th} object, respectively, and S_i the number of pixels in i^{th} object. δ specifies minimum thickness criterion for rings. Then the decision tree in [Fig. 11] was used to classify colors. For the 309 segmented code images either by edge detection or k-means clustering, color codes were successfully recognized for all cases, which amounts to the total accuracy of 99.4% (= 309/311).

III. Discussion

Even though we might skip edge detection, it played important role in removing noise pixels which otherwise would have caused decision boundaries to be sophisticated or non-error-free.

Hue component image was not as much useful as the saturation-component image and RGB image in detecting code regions. That is because hue histogram did coincide with the transformation function in Eq. (1).

From [Fig. 10(b)] we can imagine that with more image sets there may be overlap between R and M even in a*b* plane. [Fig. 12] show d_m–h_p and a*b* distributions from two of the three different image sets. Unfortunately, obtained values for the six colors were quite different from those of intended colors, and they changed with environments. Fortunately, however, since the six colors did not overlap for each image set, if EXIF (exchangeable image file format) data were added to the feature vector, a robust color code recognition could be achieved.



(a) (b)

Fig. 12. d_m–h_p and a^*b^* distributions of 2 image sets

IV. Conclusion

We employed the *k*-means clustering algorithm in segmenting color codes for which edge detectors failed. The *k*-means clustering algorithm increased the segmentation rate from 89.4% to 99.4%. Color recognition was performed on each segmented color region using 3-D feature vectors consisting of peak hue, peak a*, and peak b*. For the successfully segmented images, color code recognition accuracy was 100%.

A more robust color code classifier could be constructed if the EXIF information were added in the feature vector, and the mean shift algorithm may help improve the performance.

Reference

[1] E. Kabelka, W. Yang, and D. Francis, "Improved tomato fruit color within an inbred backcross line derived from lycopersicon esculentum and L. hirsutum involves the interaction of loci," J. Amer. Soc. Hort. Sci., Vol.129, No.2, pp.250-257, 2004.

[2] C. Garcia and G. Tziritas, "Face detection using quantized skin color regions merging and wavelet packet analysis," IEEE Transactions on Multimedia, Vol.1, No.3, pp.264-269, 1999.

[3] M. Jones and J. Rehg, "Statistical color models with application to skin detection," Cambridge Research Laboratory Technical Report Series, 1998.

[4] K. Sobottka and I. Pitas, "Extraction of facial regions and features using color and shape information," IEEE Int. Conf. on Pattern Recognition, Vol.3, pp.25-30, Aug. 1996.

[5] W. Lu and Y. Tan, "Color filter array demosaicking: New method and performance measures," IEEE Trans. Image Processing, Vol.12, No.10, pp.1194-1210, 2003.

[6] R. Gonzalez, R. Woods, and S. Eddins, Digital Image Processing, Prentice-Hall, 2004.

[7] J. Parker, Algorithms for Image Processing and Computer Vision, Wiley, 1997.

[8] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," IEEE Trans. PAMI, Vol.24, pp.603-619, 2002.

저 자 소 개

유 현 중(Hyeon-Joong Yoo) 정회원

- 1982년 : 서강대학교 전자공학과 (공학사)
- 1991년 : 미조리대학교 전기및컴퓨터공학과 (공학석사)
- 1996년 : 미조리대학교 전기및컴퓨터공학과 (공학박사)
- 1996년 ~ 현재 : 상명대학교 정보통신공학과 교수

<관심분야> : 패턴인식, 영상처리, 인공신경망