Detecting Boundaries between Different Color Regions in Color Codes

B. H. Kwon* H.-J. Yoo* and T. W. Kim**
* Maj. of I&T Engr., Sangmyung University, Chonan, Korea
Tel : +82-41-550-5349 Fax : +82-41-550-5355 E-mail: yoohj@smu.ac.kr
** Dept. of I&T, Hanyang Cyber University, Seoul, Korea
E-mail: twkim2@hycu.ac.kr

Abstract: Compared to the bar code which is being widely used for commercial products management, color code is advantageous in both the outlook and the number of combinations. And the color code has application areas complement to the RFID's. However, due to the severe distortion of the color component values, which is easily over 50 % of the scale, color codes have difficulty in finding applications in the industry. To improve the accuracy of recognition of color codes, it'd better to statistically process an entire color region and then determine its color than to process some samples selected from the region. For this purpose, we suggest a technique to detect edges between color regions in this paper, which is indispensable for an accurate segmentation of color regions. We first transformed RGB color image to HSI and YIQ color models, and then extracted I- and Y- components from them, respectively. Then we performed Canny edge detection on each component image. Each edge image usually had some edges missing. However, since the resulting edge images were complementary, we could obtain an optimal edge image by combining them.

Key words: color code, Canny edge detection, I component, Y component, HSI, YIQ, RGB

1. INTRODUCTION

Although the bar code is being widely used in managing commercial products, it has restrictions in further expanding areas of its applications particularly because of cosmetic reason. Compared to the bar code, the color code is more advantageous not only in appearance but also in the number of combinations. And compared to the RFID, which is recently emerging with the ubiquitous computing, though they have some overlapped applications, the color code still finds some applications in which the RFIF cannot be used. Moreover, the color code is more economical than the RFID.

Among the companies that have commercialized color codes are Colorzip media, Imedia, and Vividot. Vividot and Imedia have their color codes' applications in events as shown Fig. 1. The photographers of the companies upload pictures from an event. Then visitors to the event visit the sites to find their pictures and have them printed by the companies.

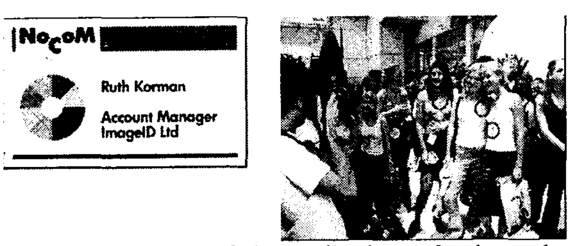


Fig. 1 An example of the applications of color code of Vividot (www.vividot.com): (a) Color code sticker (b) Color code stickers put on. The red circles indicate the color codes.

The color code of Colorzip Media is marked on news papers, magazines, business cards, etc., and its image is taken by such cameras as PC camera to be directly linked to the related DB or services. It is attractive in that it links off-line to on-line. The function that recognizes color codes on off-line products and then links customers to the site that provides related information is expected to encounter the increase of demands with the development of various services using color codes.

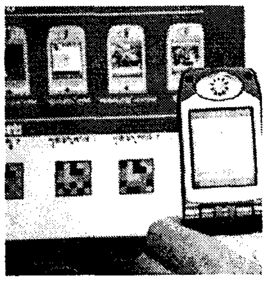


Fig. 2 An example of the applications of color code of Colorzip Media (www.colorzip.com).

However, there are some obstacles that should be overcome to make color codes more practical: that is, there is severe difference between the color used and the values read. It is not rare to find the differences well over 50 % of the scale. By drawing the Hue histogram of each color region of a color code, we could observe that the distribution of a color may overlap with those of other color(s) or even move to the range of other color(s). These problems may indicate the difficulty in the recognition of color codes.

Fig. 3 (a) and (b) show an example of designs of color codes and the hue histograms of each color region, respectively. In Fig. 3(a) the characters beside the color code represent the colors of each color regions. The red rectangle, which can be drawn either interactively or automatically via some detection procedures, around the concentric circular color code represents the area containing color code and to be processed for recognition. Fig. 3(b) shows the histograms of the region excluding the white background: The upper-most histogram is from the entire circular code area. The second is of the outer-most ring, and so on. Note that each graph has different vertical scales.

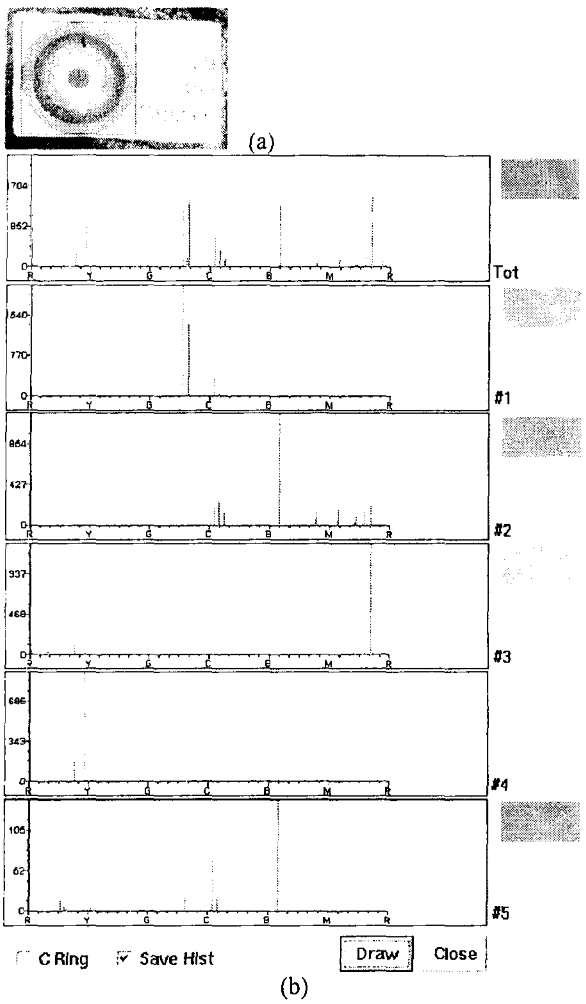


Fig. 3 A color code and its histograms. (a) A design of a color code. (b) Its histogram. (Note: Each graph has different scales.)

Notice the widely spread histogram of blue-color ring and the shift of the band of the histogram of the magenta-color ring over to the red band. From such undesirable distributions of histograms, we are expected to encounter some difficulty in color recognition. Table 1 shows the average hue values from pixels randomly selected from each color ring of the same kind of color code as one shown in Fig. 3(a). From the table we observe that the errors are well over the 50 % of the 8-bit scale.

Table 1 Colors used for a color code and their component values read from pixels randomly taken from each color region (B, M, C, G, and R stand for blue, magenta, cyan, green, and red, respectively.)

Colors (BGR)	Average values read
B = 255, 0, 0	183, 154, 142
M = 255, 0, 255	164, 140, 250
C = 255, 255, 0	242, 205, 129
G = 0, 255, 0	184, 196, 151
R = 0, 0, 255	194, 152, 208

Therefore, it may be erroneous to recognize colors from some samples extracted from a region of a color. Instead, we should include as many samples as possible from each region in the computation and process them statistically in order to ensure the credibility of the recognition. In this respect, an accurate segmentation would be the most fundamental step for accurate color recognition.

In this paper, we suggest an effective way to detect the boundaries between color regions in a color code. Because the detection of boundaries in color space may not be desirable due to the colors' severe distortion, we try an alternate way that transforms a color image into component images and then detect and combine boundaries in the component images.

The rest of this paper consists of the background knowledge in chapter 2, experiments in chapter 3, experimental results in chapter 4, analyses and discussion in chapter 5, and conclusion in chapter 6.

2. BACKGROUND

In this chapter we explain about such topics as color models, histogram, and Canny edge detection.

2.1 Color Models

Figs. 4 (a) and (b) show the RGB color cube and HSI (Hue-Saturation-Intensity) color model derived from the RGB model [3, 4], respectively.

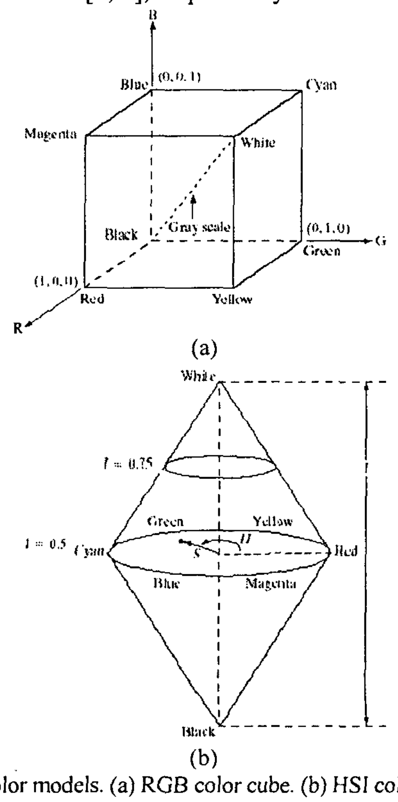


Fig. 4 Color models. (a) RGB color cube. (b) HSI color mode.

In Fig. 4(b) the circular plane is perpendicular to the I axis. The I component of this color model is the average of the R, G, and B components from Fig. 4(a).

2.2 RGB-YIQ Transformation

YIQ format is used in the NTSC color system of color TV broadcasting. It stands for Luminance-Inphase-Quadrature, where the choice of the letters YIQ is conventional [4]. Y is similar to the luminance that human recognizes, and I and Q are related to the color information. One of the advantages of this format is that gray-scale information is separate from color data, so the same signal can be used for both color and monochrome television sets. The components are obtained from the RGB components using the transformation:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} .299 & .587 & .114 \\ .596 & -.275 & -.321 \\ .212 & -.523 & .311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

Note that the elements of the first row sum to 1 and the elements of the next two rows sum to 0.

2.3 Canny Edge Detection Algorithm

Thre are various methods for edge detection including the popular one using Sobel operators [2]. In 1986, Canny set the three goals of edge detectors, and suggested an optimal algorithm to achieve them [1]. In this research we employ the Canny edge detection algorithm, and here we briefly introduce its procedures [7]:

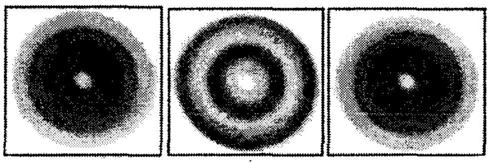
(1) Read in parameters (σ, TH, and TL). (Each parameters will be discussed later.)
(2) Input color image.
(3) Generate a 1-D Gaussian filter mask, G: Gx and Gy
   - σ determines the mask size: the samller the σ, the smaller the mask.
(4) Take the first derivatives of G along the x- and y-axes to generate 1-D mask G': Gx' and Gy'
(5) Convolve the image with G: (= Low pass filtering)
   Ix = I * Gx
   Iy = I * Gy
(6) Convolve the low-pass filtered image with G':
   Ix' = Ix * Gx'
   Iy' = Iy * Gy'
(7) Compute the lengths of the gradient vectors at each pixel and create an edge image:
   $M(x, y) = \sqrt{I_x'(x,y)^2 + I_y'(x,y)^2}$
   - M is large for edge pixel
(8) Perform the thinning on the edge image: by using the nonmaximum suppression technique [7].
(9) Remove noise from the thinned edge image and connect edges: by using the Hysteresis thresholding technique [7].

3. EXPERIMENTS

The color code recognition can largely be divided into three steps: that is, the extraction of the region of color code in an image, the segmentation of color regions in a color code, and the recognition of the color of each color region. Though the recognition problem starts from locating the color code in an image (it can be achieved, for example, by using H component and S component), we skip the locating procedures in this paper to focus on the second step.

We first transform the color code image into two gray-level images using HSI and YIQ color model. That is, we extract I component and Y component from the color image. We should use at least two different kinds of gray level images because such transformation function as Eq. (1) may map different RGB colors to the same gray level. In that case, neighboring color regions may not be distinguished in the gray-level image, even though they have different colors.

Fig. 5 shows a color code used in the experiment, and its I-component, and Y-component images, respectively. Fig. 5 (a) was obtained by cropping the color code area from a real image including an object with the color code sticker put on. Though we skip explanation about the steps for detecting the area of color code in an image, we can efficiently perform it using H-component and S-component images [5, 6].



(a) (b) (c)
Fig. 5 A color code and its component images: (a) A color code. (b) I-component image. (c) Y-component image

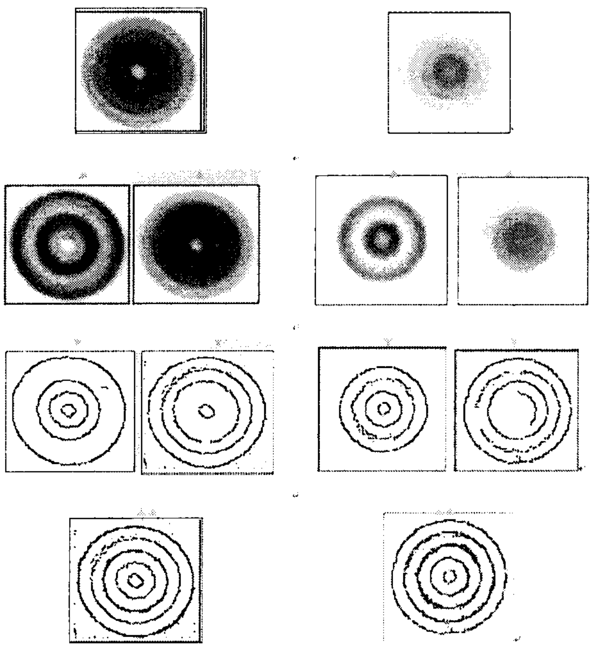
We apply Canny edge detector to the gray-level images to detect boundaries between color regions. We try different values of σ, which determines the width of the Gaussian filter in the step of low-pass filtering (Step 5), and different values of the upper-level threshold (TH) in the step of Hysteresis thresholding (Step 9). We usually use low σ and TH values to expose and detect most of boundaries. And we finally combine the results from the two gray-level images to create an optimal boundary image by the logical AND between the two boundary images.

4. EXPERIMENTAL RESULTS

In Fig. 6(a), the three images in the first and second rows are the same as those in Fig. 5. The images in the third row show the results of performing Canny edge detection on the images in Figs. 5 (b) and (c), respectively. That is, they are the edge detection results with I-component and Y-component images, respectively. For the I-component image, we used σ of 2.0, which determined the size of Gaussian filter mask in the low-pass filtering step, and used TH_ratio of

0.5, which determined the upper-level threshold TH, in the Hysteresis thresholding step. Here, TH_ratio stands for the ratio of the number of edge pixels whose magnitude, M, is greater than TH, to the total number of edge pixels. Thus, choosing smaller value for TH_ratio increases TH. The size of mask increases with σ, so if we increase the value, the result will show the stronger averaging effect, that is, smoother images. For Y-component image, we used 1.0 for σ and 0.7 for TH_ratio.

The bottom-most figure in Fig. 6(a) was obtained by merging the two edge images. That is, we performed logical AND operation between the two images. Fig. 6(b) shows the results with another test color code. The explanation for the result is similar to that for Fig. 6(a), except that we used σ = 2.0 and TH_ratio = 0.5 for both I- and Y-component images.



(a) For color code 1 (b) For color code 2

Fig. 6 The results of edge detections in I- and Y- component images and of merging the edge images.

5. ANALYSES AND DISCUSSION

From Fig. 6, we observe that the edges from the two component images from different color models were complementary, and combining the results successfully provided an optimal edge image.

Mainly due to the fact that such transformation functions as HSI and YIQ can map different RGB colors to the same component value, we couldn't see the boundaries between color regions in the transformed images in some cases, and couldn't detect such boundaries with the Canny edge detector, either. Lowering the values of such parameters as σ and TH to solve the problem would worsen it by adding more false boundaries than true boundaries, which would increase the amount of refining works to filter out false boundaries. Therefore, to make the method practical, we tried two component images which were complementing each other; then combined their results. This scheme effectively excluded the necessity of lowering the parameters too much.

As can be observed from the figures in the first row of Fig. 6, neighboring color regions caused interference (or color distortion) around the boundary between them. The interference could make it impossible to separate the color regions when the width of a region was narrow. We could observe that pictures containing color codes need be taken large enough to have each color region appear at least 7 pixels wide.

Since the detected boundaries are from images suffering from such distortions, the results can not be trusted 100 %. Thus, we suggest using as many pixels as possible from along the region in the middle of the neighboring boundaries and statistically processing them in the color code recognition.

6. CONCLUSION AND FUTURE STUDY

We proposed an effective way to detect boundaries between color regions, which is the most fundamental step for segmentation and eventually for color recognition. Instead of finding the boundary directly from a color image which usually contains severe color distortions, we extracted I and Y components from an RGB color image and applied Canny edge detector to the two gray-level images. The edge detection results with the two gray-level images complemented each other. Combining the edge-detection results provided an optimal boundary image of color code.

It is necessary to develop an algorithm which adaptively determines parameter values such as the size of mask in the low-pass filtering step and the upper-level threshold in the Hysteresis thresholding step. More researches required on such topics as removing false boundaries and connecting true boundaries, and statistically processing pixels and histograms from a color region to complete color code recognition.

References

[1] J. Canny, "A Computational Approach to Edge Detection", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698, 1986.

[2] R. O. Duda and P. E. Hart, "Use of the Hough Transformation to Detect Line and Curve in Pictures", *Comm. ACM*, vol. 15, no. 1, pp. 11-15, 1972.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed., Prentice-Hall, 2002.

[4] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*, Prentice-Hall, 2004.

[5] B. H. Kwon and H.-J. Yoo, "Efficient Method to Detect Color Codes – RHOW Alogrithm", *Trans. of KAIS*, vol. 5, no. 1, pp. 69-72, 2004.

[6] H. J. Lee, et. al., "The Extraction of Car-License Plates Using Combined Color Information of HSI and YIQ", *Trans. of Korean Information Processing Society*, vol. 7, no. 12, pp. 3995-4003, 2000.

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