International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:12, 2013

A Way of Converting Color Images to Gray Scale Ones for the Color-Blind -Applying to the Part of the Tokyo Subway Map-

Katsuhiro Narikiyo, Shota Hashikawa

Abstract—This paper proposes a way of removing noises and reducing the number of colors contained in a JPEG image. Main purpose of this project is to convert color images to monochrome images for the color-blind. We treat the crispy color images like the Tokyo subway map. Each color in the image has an important information. But for the color blinds, similar colors cannot be distinguished. If we can convert those colors to different gray values, they can distinguish them. Therefore we try to convert color images to monochrome images.

Keywords—Color-blind, JPEG, Monochrome image, Denoise.

I. Introduction

MONOCHROME images are shown only by brightness such as white, black, and grays. The brightness is called Gray value. When we convert color images to monochrome images, sometimes two different colors are printed with same gray value. In this case we cannot distinguish different colors. In order to convert them, we have to know how many colors are included in a color image.

Basic idea of the method is introduced with using a 4-color simple image[1]. If 4 colors are used in a image, we just give white, black, light gray, and dark gray for each color. It looks so easy. However a JPEG image[2], we usually use this format, includes a number of colors because of its compression way. In the following sections, we show an example image which originally has only four colors. While it is saved in JPEG format, it has 4,483 colors. We show how to reduce the number of colors step by step using it. Finally we can convert it to 4-leveled monochrome image.

As an application of the method, we try to treat the part of the Tokyo subway map shown as Fig. 1[3]. Note that Fig. 1 includes some Chinese characters but the meaning of the characters are not important but their shape complexity.

II. JPEG Color Image and Monochrome Image

There are some kinds of image formats. JPEG format is the most popular one. We treat the JPEG image in this paper as shown in Fig. 2(a). It looks only four colors included in it. If we convert this image to monochrome by the conventional way such as

$$Gray = 0.299R + 0.587G + 0.144B, \quad (1)$$

K.Narikiyo and S.Hashikawa are with the Department of Electronics Control Engineering, Hiroshima National College of Maritime Technology, 4272-1 Higashino, Osakikamijima, Toyota, Hiroshima, 725-0231 JAPAN e-mail: (see http://dep.hiroshima-cmt.ac.jp/∼control/staff/narikiyo/index.html).

Fig. 1. Part of Tokyo subway map

where R, G, and B are color component value for each pixel[4]. The converted monochrome image by this expression is shown as Fig. 2(b). The JPEG image usually has noises because of its compression way. Even a very simple image such as Fig. 2(a) contains so many colors. This image contains 4,483 colors. Therefore it is difficult to assign gray value to each color.

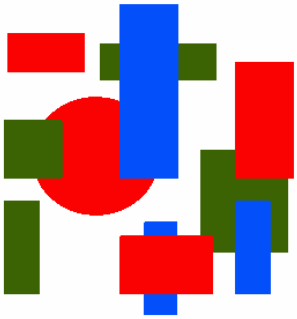
III. Assigning Gray Value to Each Color

When we convert color image to monochrome image for the color blinds, we think a way that we assign gray value to each color. As we mentioned before, even a simple color image includes so many colors. Therefore before assigning gray value to each color, we have to reduce the number of colors. We show the process of reducing the number of colors and assigning gray values as follows:

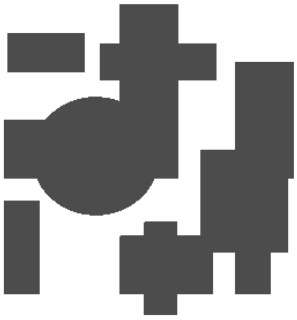
A) Detecting Edges
B) Splitting Surrounded Region by Edge
C) Painting Each Region with Average Color
D) Grouping Similar Color Regions
E) Painting Each Region with Average Color in a Group
F) The Edge Region Processing
G) Assigning Gray Values to Each Color

A. Detecting Edges

First, We distinguish color regions by detecting their edge using the Sobel amp. The Soble amp detects the border of

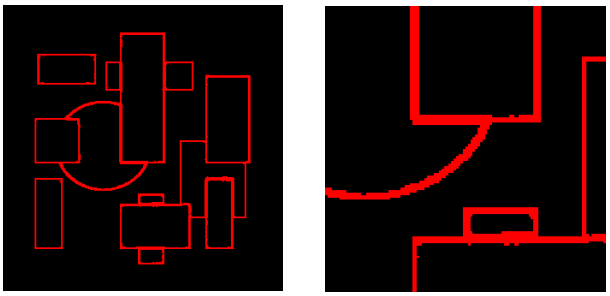


(a) JPEG Color Image



(b) Ordinary Monochrome Conversion

Fig. 2. JPEG color image and converted monochrome image



(a) Edge Regions (b) Zoomed

Fig. 3. Detecting Edge Regions

color regions. Detected edge regions are shown in red lines in Fig. 3.

B. Splitting Surrounded Region by Edge

There are surrounded regions by the edge. There are 14 surrounded regions. We put labels on each region as shown in Fig. 4.

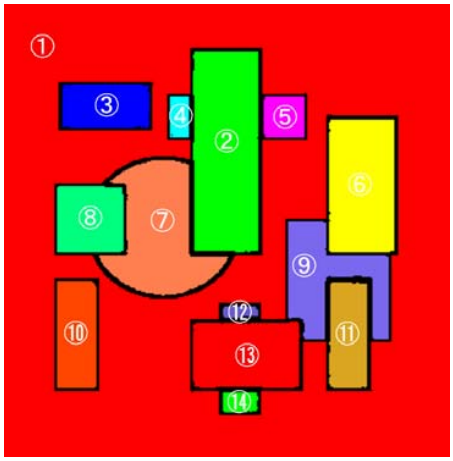


Fig. 4. Fourteen Regions

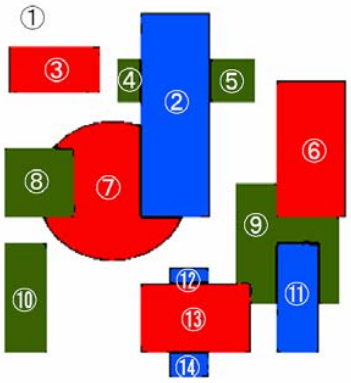


Fig. 5. Fourteen Colors and Edge in Black

C. Painting Each Region with Average Color

We calculate average colors for each of regions. Afterwards, we paint the regions by the average colors(Fig. 5). So that Fig. 5 contains 14 colors and black at the edge region.

D. Grouping Similar Color Regions

Table I shows RGB values for each of regions. Region 4 and 5 in Fig. 5 have different RGB vales because they come from the average calculation. These two regions must have same color. To solve this problem, we group color regions. For the grouping, we use the Cluster analysis[5]. Result of the Cluster analysis is shown as Fig. 6 and grouping result is at the last column in Table I . Then we again calculate the average color for each of groups.

E. Painting Each Region with Average Color in a Group

Table.II shows the average RGB values for each of groups. By painting these RGB vales we can obtain Fig. 8. It contains only 4 colors and black for the edge region.

TABLE I
FOURTEEN AVERAGE COLOR FOR EACH OF REGIONS

No.	R	G	B	Group
1	254	254	254	1
2	1	80	253	2
3	249	2	2	3
4	59	98	7	4
5	58	99	5	4
6	252	1	1	3
7	250	2	1	3
8	61	98	2	4
9	59	99	5	4
10	59	99	3	4
11	2	80	252	2
12	5	79	248	2
13	251	2	2	3
14	3	79	249	2

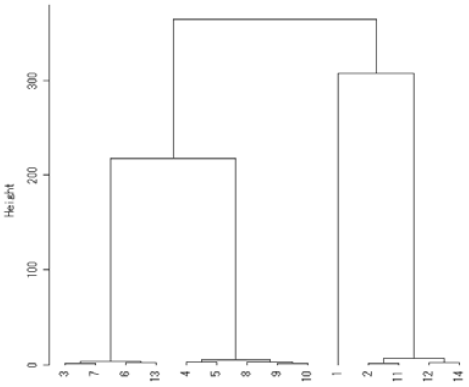


Fig. 6. Cluster Analysis's Output

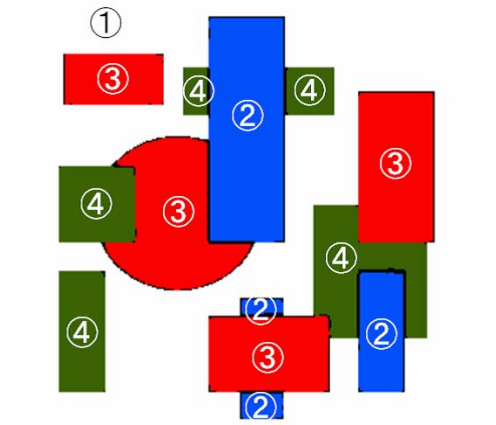


Fig. 7. Painting Each Region with Average Color in a Group

TABLE II
FOUR AVERAGE COLORS FOR EACH OF GROUPS

No.	R	G	B
1	254	254	254
2	3	80	250
3	250	2	2
4	59	99	4

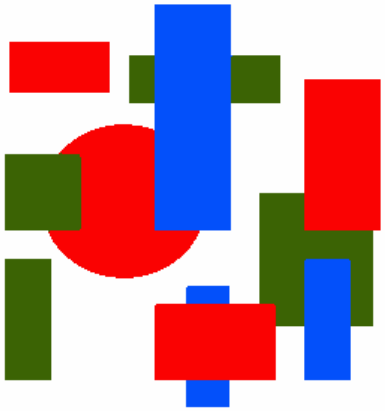


Fig. 8. After Edge Area Processing

F. The Edge Region Processing

Finally, we have to process the edge region. So far, each pixel except the edge region has one of four colors. We choose one color from four colors for each pixel on the edge. For the each pixel on the edge, we use the original color of the pixel and colors of its 8-neighborhood.

For the every pixel q on the edge region E in Fig. 3, let N be a set of 8-neighborhood pixels of q. Let a set P as

$$P = \{p | p \in N \text{ and } p \notin E\}. \tag{2}$$

Then number each element of P as

$$P = \{p_1, p_2, \ldots, p_n\}. \tag{3}$$

Also, the RGB values of the pixel p are expressed by $R(p)$, $G(p)$, and $B(p)$. For the all element of P, we define the difference of colors d_i by

$$\begin{aligned} d_i^2 &= (R(p_i) - R(q))^2 + (G(p_i) - G(q))^2 \\ &+ (B(p_i) - B(q))^2, \end{aligned} \tag{4}$$

where $R(q)$, $G(q)$, and $B(q)$ are the original RGB value of q. Then we calculate

$$m = \min_{1 \le i \le n} d_i. \tag{5}$$

We give the RGB vales of the pixel p_i which has $m = d_i$ to q. We show the result of this process in Fig. 8. It contains only four colors.

Fig. 9. Result of the Proposed Monochrome Conversion

Fig. 10. Part of Tokyo Subway Map Including 46,854 Colors

G. Assigning Gray Values to Each Color

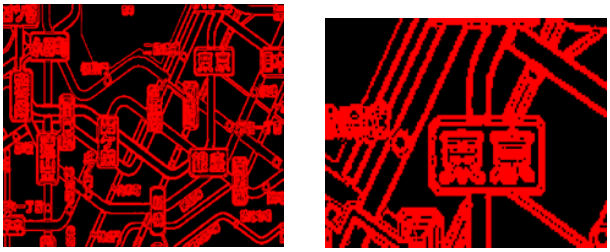
There are only four colors in Fig. 8. We can assign 4 gray values to each of colors. The background color is white. Blue is light gray. Red is dark gray. And Green is black. Gray values to each of colors are different. Therefore we can distinguish different color region in this image shown as Fig. 9.

IV. Application for the Tokyo Subway Map

Now we try to apply the method to the part of the Tokyo subway map shown as Fig 10. This image includes lines, characters, and 46,854 colors. It is too complicated to apply the method directly.

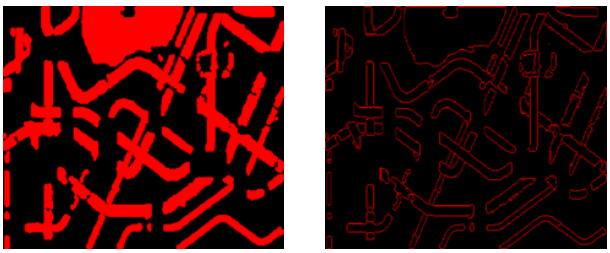
A. Detecting Edges

The edge region of the original image Fig.10 is shown as Fig.11. Especially the regions of characters are so complicated to split into isolated regions. To avoid this difficulty, we assume that the color of the characters is black and the background color is white or gray. Therefore we treat only the high color saturation area(Fig.12(a)). Edge area on the high color saturation area is shown as Fig.12(b).



(a) Edge Regions (b) Zoomed

Fig. 11. Detecting Edge Regions



(a) High Saturation Area (b) Edge Area

Fig. 12. Detecting Edge Regions on High Color Saturation Area

B. Painting Each Region with Average Color in a Group

Fig. 12(b) has 88 separated regions, then we calculate average colors for each of regions. Afterward, we paint each region by their average color(Fig. 13). So that Fig. 13 contains 88 colors and black at the edge region and low color saturation area.

C. Grouping Similar Color Regions

In the same manner described in the previous section, all of 88 regions are classified into 13 groups by using the Cluster analysis. We obtain 13 color regions shown as Fig. 14.

Fig. 13. Eighty Eight Colors on High Color Saturation Area

Fig. 14. Thirteen Colors on High Color Saturation Area

Fig. 15. Thirteen Gray Levels Assigned for Each Colors with Background

D. Assigning Gray Values to Each Color

Fig. 15 is the result we assigned different gray values for each color. It is difficult to distinguish lines because of many gray levels.

E. Using Some Kind of Hatching Patterns

Using only solid gray scale areas is difficult for us to distinguish for many levels. Instead, if we use also hatched area, we can easily distinguish different lines. Figure 16 is the result with some hatching areas and background.

V. Conclusion

We started discussion with the example of the 4 color JPEG image. First of all, we painted 14 regions with the average colors so that we got 14-color color image. And then we used cluster analysis and processed edge region. Then we could get 4-color image. And then, because the image has 4-colors only, we could assign gray values to each of colors. Finally we could convert color image to monochrome image for the color-blind. As an application, we applied our method to a part of the Tokyo subway map. We can make it monochrome with 13 gray level image, but it is difficult to distinguish 13 different gray levels. In this case we use not only gray levels but also some hatchings.

Fig. 16. Thirteen Gray Level with Hatching Area with Background

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 24650463.

References

[1] K. Narikiyo, K. Yamaoka, *A Way of Converting Color Images to Gray Scale Ones For The Color blinds - Reducing the number of colors for JPEG images* - World Academy of Science, Engineering and Technology Vol:60, pp.656–659, 2011.
[2] William B. Pennebaker, Joan L. Mitchell, *JPEG: Still Image Data Compression Standard (Digital Multimedia Standards)*, Springer, 1992.
[3] K. Narikiyo, N. Kobayakawa, *A Way of Converting Color Images to Gray Scale Ones For The Color blinds -Reducing the colors for Tokyo Subway Map*- World Academy of Science, Engineering and Technology Vol:72, pp.1412–1416, 2012.
[4] MVTec Software GmbH, *HALCON REFERENCE MANUAL*, MVTec Software GmbH, 1996.
[5] Michael J. Crawley, *The R Book*, Wiley, 2007.