Applications of Color Image Processing Using Three-Layer Cellular Neural Network Considering HSB Model

Takashi Inoue and Yoshifumi Nishio

Abstract— **In this study, we propose a three-layer cellular neural network dealing with color images considering the HSB model. Firstly, the conversion method of color images to three grayscale images using the HSB model is explained. Secondly, the structure of the proposed three-layer cellular neural network considering the HSB model is explained. Simulation results of some color image processing show the basic properties of the proposed network and its effectiveness.**

I. Introduction

CELLULAR Neural Networks (CNN) [1] were introduced by Chua and Yang in 1988. The idea of the CNN was inspired from the architecture of the cellular automata and the neural networks. Unlike the conventional neural networks, the CNN has local connectivity property. Since the structure of the CNN resembles the structure of animals' retina, the CNN can be used for various image processing application [2]-[4] including character extractions [5][6]. Further, the CNN can be utilized to produce some kinds of pattern generation [7][8] and the texture segmentation [9].

The humans retina have an ability distinguishing colors and consist of three types of cells called "cone" responding to the three primary colors and a cell called "rod" responding to the amount of light. The humans can recognize colors by the function of the cone cells. Roska et al. have proposed a concept using a three-layer CNN processing the three primary colors in [10]. They have confirmed that the three-layer CNN could produce half-toned images of color images. However, after their pioneering work, there have not been many researches on the CNN dealing with color images effectively.

In the previous study, we have proposed the three-layer cellular neural networks considering three primary colors (RGB-CNN) [11]. Before the processing, a color image is divided into three primary colors and they are converted to three grayscale images using RGB model. This preprocess corresponds to the function of the cone cells of the retina. These three images are inputted to the three-layer cellular neural networks. In our RGB-CNN, the connections between the three layers play an important role. Namely, the three layers do not operate independently but all the outputs influence to the other layers. By using RGB-CNN, we obtained the effective edge detection results and interesting output characteristics. However, we think that the RGB-CNN is different from the perception of human. In the RGB model, the blue-element value of yellow is zero. However, when human recognize the color of yellow, the blue-element may have some amount of value. Additionally, the color of white in RGB model has the value of (R: 255, G: 255, B: 255). On the flip side, the pure colors of R, G and B have values of (R: 255, G: 0, B: 0), (R: 0, G: 255, B: 0) and (R: 0, G: 0, B: 255), respectively. Namely, we can not distinct white from red by using only the red-element value in the RGB model. The color image processing using CNN imitating the humans recognition method of color may have more powerful color image processing applications.

In this study, we concentrate on the HSB model in conversion from color image to grayscale image. The HSB model comprises three components of "Hue", "Saturation" and "Brightness". The color in the RGB model is defined by combination of three primary colors. On the other hand, the principle of the HSB model is similar to the perception of human, and human recognize color as "What is a color ?", " How much is vibrant ?", " How much is bright ?". Namely, the CNN considering the HSB model (HSB-CNN) may be more similar to the processing of human than the CNN considering the RGB model. We carry our some examples of color image processing using the proposed HSB-CNN and confirm its basic properties and the effectiveness.

In the Sec. 2, we review the basic of the standard CNN. In the Sec. 3, we show the conversion method of color images using the HSB model and the structure of the proposed HSB-CNN. In the Sec. 4, simulation results of some color image processing are shown. Section 5 concludes the article.

II. Cellular neural networks[1]

In this section, we describe the basic structure of the CNN. The CNN has M by N processing unit circuits called cells. Cells are arranged in a reticular pattern to M line N row. We represent a cell $C(i, j)$ using a variable i which denotes vertical position and a variable j which denotes horizontal position. The cell contains linear and nonlinear circuit elements. The CNN is an array of cells. Each cell is connected to its neighboring cells according to a template. Usually, the template is the same for all cells except for boundary cells. The CNN has the features of time continuity, spatial discreteness, nonlinearity and parallel processing capability.

The state equation and the output equation of the cell are shown as follows.

Department of Electrical and Electronic Engineering, Tokushima University, 2-1 Minami-Josanjima, Tokushima 770-8506, Japan (email: {takashi, nishio}@ee.tokushima-u.ac.jp).

978-1-4244-3553-1/09/$25.00 ©2009 IEEE

State equation:

$$\frac{dv_{xij}}{dt} = -v_{xij} + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} A_{(i,j;k,l)}v_{ykl}(t) + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} B_{(i,j;k,l)}v_{ukl}(t) + I \quad (1)$$

Output equation:

$$v_{yij}(t) = \frac{1}{2}(|v_{xij}(t) + 1| - |v_{xij}(t) - 1|) \quad (2)$$

where v_x, v_y and v_u represent a state, an output and an input of cell, respectively. In the equation (1), A is the feedback template and B is the control template. These and bias I are collectively called general template.

The $r - neighborhood$ of the cell $C(i,j)$ in the CNN is defined by

$$Nr(i,j) = \{C(k,l) \mid \max\{|k-i|, |l-j|\} \le r, \quad 1 \le k \le M; \quad 1 \le l \le N\} \quad (3)$$

where r is a positive integer number. In our study, we fix the value of r as 1.

III. Proposed color image processing method (HSB-CNN)

In the original CNN, color images are converted to grayscale images before the processing. The grayscale images do not have color information. Therefore, the conventional CNN is impossible to consider color information. The proposed HSB-CNN is a three-layer CNN considering HSB model.

A. HSB Model

The HSB model comprises of the three components "Hue", "Saturation" and "Brightness".

1) Hue:

$$H(i,j) \quad (0^\circ \le H \le 360^\circ). \quad (4)$$

The Hue H is shown by a degree between 0° and 360°. Figure 1 shows the hue circle in the HSB model. In the hue circle, the three primary colors red, green and blue are represented by 0° (= 360°), 120° and 240°, respectively.

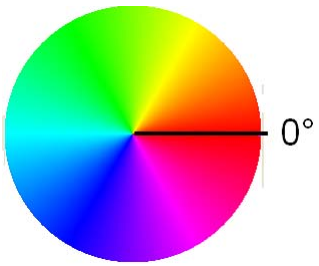

Fig. 1. Hue circle.

2) Saturation:

$$S(i,j) \quad (0 \le S \le 1.0). \quad (5)$$

The Saturation S is shown by a positive value between 0.0 and 1.0. Figure 2 shows the aspect of the saturation. When the value of S is near 1.0, color looks a pure color, while near 0.0, color looks a muddy color.

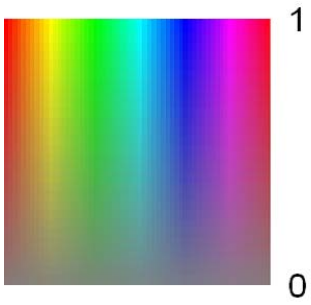

Fig. 2. Saturation.

3) Brightness:

$$B(i,j) \quad (0 \le B \le 1.0). \quad (6)$$

The Brightness B is shown by a positive value between 0.0 and 1.0. Figure 3 shows the aspect of the brightness. When the value of B is near 1.0, color becomes bright, while near 0.0, color becomes dark.

Fig. 3. Brightness.

In the HSB model, any colors are represented by these three values H, S and B.

B. Conversion method from a color image to grayscale images

In this subsection, we describe the conversion method using HSB model from a color image to grayscale images. First, we consider the conversion from a color image to grayscale images corresponding to the hue value. Because a grayscale image cannot be represented by a value of degree, the hue value needs to be converted from degree to a real value in a certain interval. Further, the hue value is divided into three components corresponding to the three primary colors, in order to make it easy to process color images. The conversion equations of the hue are shown as follows.

Conversion Equation of Hue:

Hue for Red:

$$H_R(i,j) = \cos(H(i,j)^\circ) \quad (-1.0 \le H_R \le 1.0). \quad (7)$$

Hue for Green:

$$H_G(i,j) = \cos(H(i,j)^\circ + 120^\circ) \quad (-1.0 \le H_G \le 1.0). \quad (8)$$

Hue for Blue:

$$H_B(i,j) = \cos(H(i,j)^\circ + 240^\circ) \quad (-1.0 \leq H_B \leq 1.0). \tag{9}$$

Note that we use the cosine function in Eqs. (7), (8) and (9) in order to extract the three-color components. This means that, for example, even "green" color includes some "red" component.

Second, we combine the three values of H, S and B. The combination equations are shown as follows.

Combination Equation for Red:

$$\begin{aligned} G_R(i,j) &= H_R(i,j) \times 0.6 \\ &+ ((S(i,j) - 0.5) \times 2) \times 0.2 \\ &+ ((B(i,j) - 0.5) \times 2) \times 0.2. \\ &\quad (-1.0 \leq G_R \leq 1.0). \end{aligned} \tag{10}$$

Combination Equation for Green:

$$\begin{aligned} G_G(i,j) &= H_G(i,j) \times 0.6 \\ &+ ((S(i,j) - 0.5) \times 2) \times 0.2 \\ &+ ((B(i,j) - 0.5) \times 2) \times 0.2. \\ &\quad (-1.0 \leq G_G \leq 1.0). \end{aligned} \tag{11}$$

Combination Equation for Blue:

$$\begin{aligned} G_B(i,j) &= H_B(i,j) \times 0.6 \\ &+ ((S(i,j) - 0.5) \times 2) \times 0.2 \\ &+ ((B(i,j) - 0.5) \times 2) \times 0.2. \\ &\quad (-1.0 \leq G_B \leq 1.0). \end{aligned} \tag{12}$$

After the combination in Eqs. (10), (11) and (12), G_R, G_G and G_B become the values of the cells in three grayscale images. The S and B have values between 0.0 and 1.0. However, the value of grayscale images and the hue value after the conversion are between -1.0 and 1.0. Therefore, we need to convert the values of S and B to values between -1.0 and 1.0. Additionally, the values of H, S and B should be weighted before the combination. Because we consider some image processing in which the difference of colors is important, we consider that H is most important and that the weight of H should be larger than the other two.

1) Example of conversion to grayscale images: Figure 4 shows an example of the conversion of a color image to three grayscale images using the conventional RGB model. In this conversion, a color value is divided into three primary colors (red, green and blue) and they are represented by grayscale images.

In Figs. 4(b), (c) and (d), we can see that the three grayscale images are similar except only very pure colors (e.g. red bridge handrails). This is because the RGB model does not consider the effect of the brightness and hence colors with similar brightness give similar values to the three components.

Figure 5 shows an example of the conversion of the same color image to three grayscale images using the proposed HSB model Eqs. (4)-(12). In Figs. 5(b), (c) and (d), we can see large difference of the grayscale images. Namely, red parts (bridge handrails) have large values in Fig. 5(b), green parts (trees) have large values in Fig. 5(c), and blue parts (sky) have large values in Fig. 5(d). We can say that the conversion using the HSB model is more effective than the RGB model in color image processing.

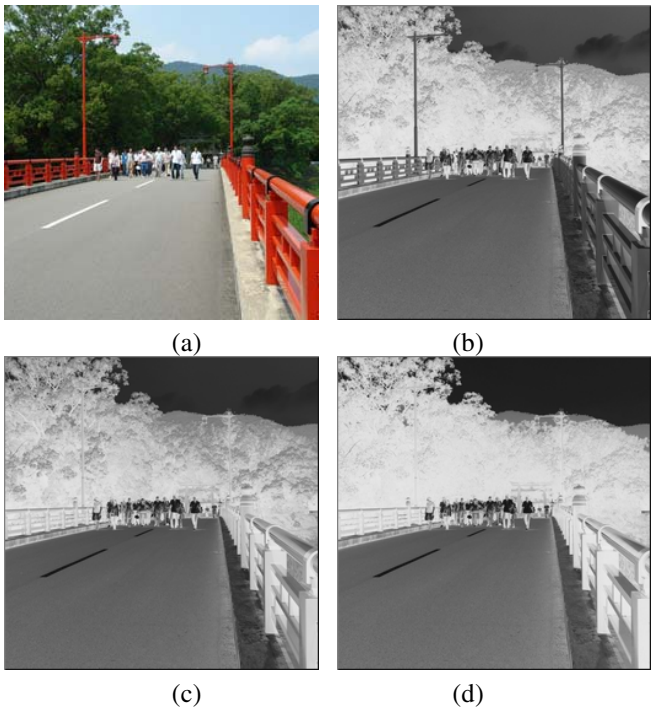

Fig. 4. Example of conversion to grayscale images using RGB model. (a) Color input image. (b) Grayscale image for red. (c) Grayscale image for green. (d) Grayscale image for blue.

C. Structure of Three-Layer CNN (HSB-CNN)

In this subsection, we describe the structure of a three-layer CNN processing the three grayscale images prepared by the proposed HSB model, called HSB-CNN. Figure 6 shows the structure of the proposed HSB-CNN. The HSB-CNN consists of three single-layer CNNs. Each layer is coupled to another layer with the connection template (C_R, C_G, C_B). Because of these connection templates, the red layer has a direct influence to the green layer. Also, the green layer has a direct influence to the blue layer, and the blue to the red. We can say that all layers have mutual influences each other.

The dynamical system equations associated with the HSB-CNN are described by the following equations:
(1) Three first-order differential equations:

$$\begin{aligned} \frac{dv_{xR,ij}}{dt} &= -v_{xR,ij} + \sum_{k=i-r}^{i+r} \sum_{l=j-r}^{j+r} A_{R(i,j;k,l)} v_{yR,kl}(t) \\ &+ \sum_{k=i-r}^{i+r} \sum_{l=j-r}^{j+r} B_{R(i,j;k,l)} v_{uR,kl}(t) \\ &+ \sum_{k=i-r}^{i+r} \sum_{l=j-r}^{j+r} C_{B(i,j;k,l)} v_{yB,kl}(t) + I_R \end{aligned} \tag{13}$$

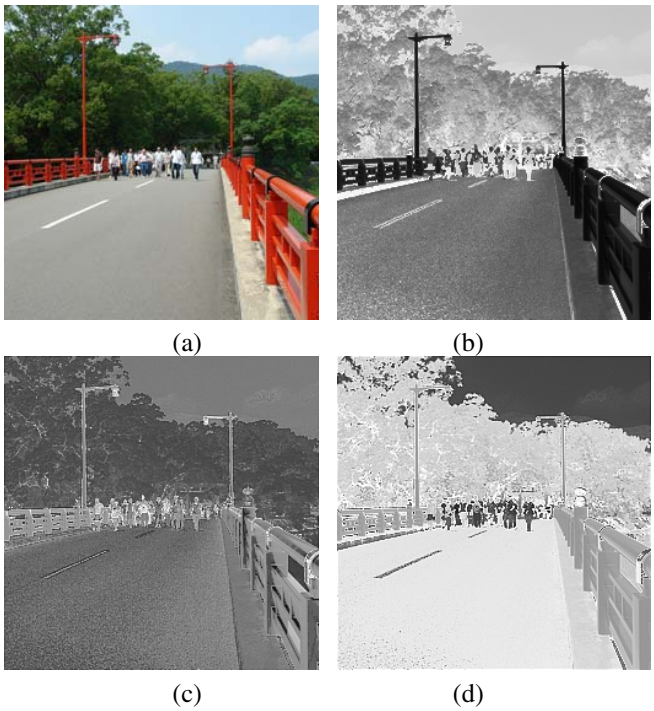

Fig. 5. Example of conversion to grayscale images using HSB model. (a) Color input image. (b) Grayscale image for red. (c) Grayscale image for green. (d) Grayscale image for blue.

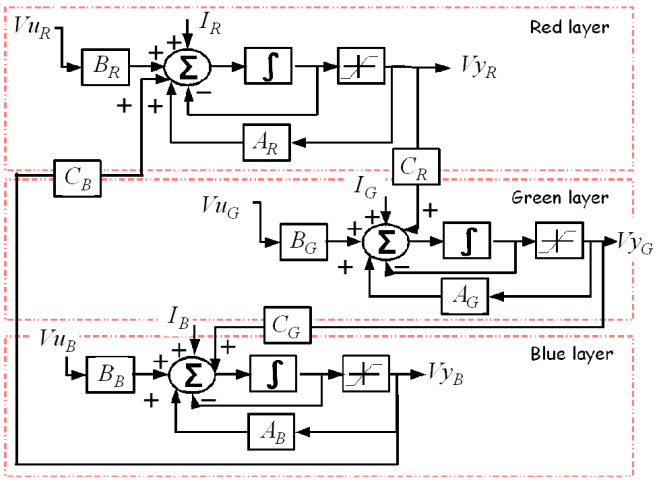

Fig. 6. Structure of HSB-CNN.

$$\frac{dv_{xG,ij}}{dt} = -v_{xG,ij} + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} A_{G(i,j;k,l)}v_{yG,kl}(t) + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} B_{G(i,j;k,l)}v_{uG,kl}(t) + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} C_{R(i,j;k,l)}v_{yR,kl}(t) + I_G \quad (14)$$

$$\frac{dv_{xB,ij}}{dt} = -v_{xB,ij} + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} A_{B(i,j;k,l)}v_{yB,kl}(t) + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} B_{B(i,j;k,l)}v_{uB,kl}(t) + \sum_{k=i-r}^{i+r}\sum_{l=j-r}^{j+r} C_{G(i,j;k,l)}v_{yG,kl}(t) + I_B \quad (15)$$

(2) Three output equations:

$$v_{yR,ij}(t) = \frac{1}{2}(|v_{xR,ij}(t) + 1| - |v_{xR,ij}(t) - 1|) \quad (16)$$

$$v_{yG,ij}(t) = \frac{1}{2}(|v_{xG,ij}(t) + 1| - |v_{xG,ij}(t) - 1|) \quad (17)$$

$$v_{yB,ij}(t) = \frac{1}{2}(|v_{xB,ij}(t) + 1| - |v_{xB,ij}(t) - 1|) \quad (18)$$

where v_x, v_y, and v_u represent a state, an output and an input of cell, respectively. In the equations (13), (14) and (15), C are the connection templates introduced to couple the three layers.

Figure 7 shows the processing of the HSB-CNN. Before the processing, a color image is converted into three grayscale images using the proposed HSB model Eqs. (4)-(12). These three images are inputted to the HSB-CNN as v_{uR}, v_{uG}, and v_{uB}. Finally, we can obtain three possible output images from the HSB-CNN.

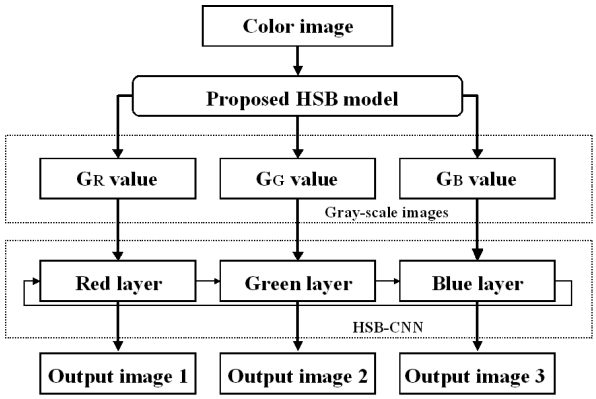

Fig. 7. Processing of HSB-CNN.

IV. Simulation Results

In this section, we show the applications of color image processing using HSB-CNN by computer simulation. All the original templates in this section are found in [12].

A. Task 1: Color binarization

First, we consider the task of color binarization, namely a color image is converted to a binary image as follows. Only colors near pure red, pure green or pure blue are extracted as a black pixels.

We use the following template from the template library [12].
Grayscale to Binary Threshold Template:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I. \qquad (19)$$

In the original CNN, by using this template, grayscale images can be converted to binary images according to the threshold value which is given as the value of I.
Color Binarization Template of HSB-CNN:
Template of Red layer : Template (19).
Template of Green layer : Template (19).
Template of Blue layer : Template (19).
Coupling template:

$$C_R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{cases} b = 4 & if \quad v_{yR,ij} \geq 0.9 \\ b = 0 & otherwise \end{cases}$$

$$C_G = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{cases} b = 4 & if \quad v_{yG,ij} \geq 0.9 \\ b = 0 & otherwise \end{cases}$$

$$C_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{cases} b = 4 & if \quad v_{yB,ij} \geq 0.9 \\ b = 0 & otherwise \end{cases} \qquad (20)$$

Remark: The HSB-CNN usually gives three different output images from the three layers. However, in this task, only one output image is needed. Therefore, we apply threshold type function to the coupling templates in HSB-CNN. By using this threshold type of template (20), output images of all layers in HSB-CNN become the same.

Figure 8 shows the simulation results of color binarization of "mandrill" image using template (20) in HSB-CNN. In this simulation, we change the threshold value of I in the templates of the three layers. From these results, we can see that the regions near pure colors of red, green and blue remain in the output and the area is changed according to the threshold value. Namely, we can say that the conversion from grayscale images to binary images is expanded to color images using the proposed HSB-CNN.

B. Task 2: Color edge detection

Secondly, we consider the task of color edge detection, namely edges of a three primary colors are extracted.

We use the following template from the template library [12].
Edge detection :

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad I = -1. \qquad (21)$$

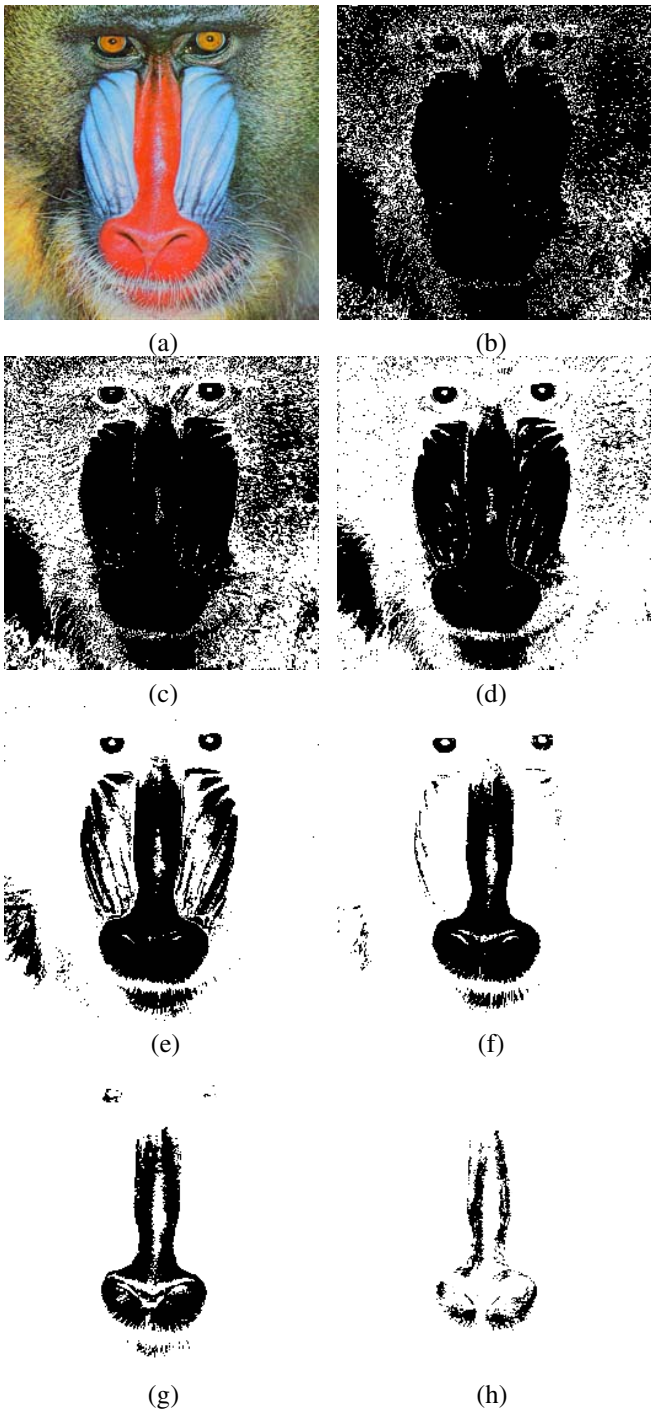

Fig. 8. Task 1 (mandrill): Color binarization using template (20) in HSB-CNN. (a) Input color image. (b) Output image for $I = -0.3$. (c) $I = -0.4$. (d) $I = -0.5$. (e) $I = -0.6$. (f) $I = -0.7$. (g) $I = -0.8$. (h) $I = -0.9$.

The "Edge Detection" template can detect the edges of binary images in the original CNN.

In this simulation, we set the template (19) or the template (21) to each layer. For example, we set the template (19) as the template of red, and the template (21) as the templates of green and blue layers. The threshold value of "Grayscale to Binary Threshold" template (template (19)) is set as

$I = -0.4$. By the above template setting, we expect that the region near pure red in color input image becomes black, and the edges of color input image can be detected at the same time.

Color Edge Detection Template of HSB-CNN:

Template of Red layer : Template (19) or (21).
Template of Green layer : Template (19) or (21).
Template of Blue layer : Template (19) or (21).
Coupling template:

$$C_R = \begin{bmatrix} 0 & a & 0 \\ a & b & a \\ 0 & a & 0 \end{bmatrix} \begin{cases} a = 1,\ b = 4 & if\ v_{yR,ij} \geq 0.9 \\ a = b = 0 & otherwise \end{cases}$$

$$C_G = \begin{bmatrix} 0 & a & 0 \\ a & b & a \\ 0 & a & 0 \end{bmatrix} \begin{cases} a = 1,\ b = 4 & if\ v_{yG,ij} \geq 0.9 \\ a = b = 0 & otherwise \end{cases}$$

$$C_B = \begin{bmatrix} 0 & a & 0 \\ a & b & a \\ 0 & a & 0 \end{bmatrix} \begin{cases} a = 1,\ b = 4 & if\ v_{yB,ij} \geq 0.9 \\ a = b = 0 & otherwise \end{cases} \quad (22)$$

Remark: In general, edge detection results include some spot noises. In order to solve this problem, we add the noise removal effect to the coupling templates.

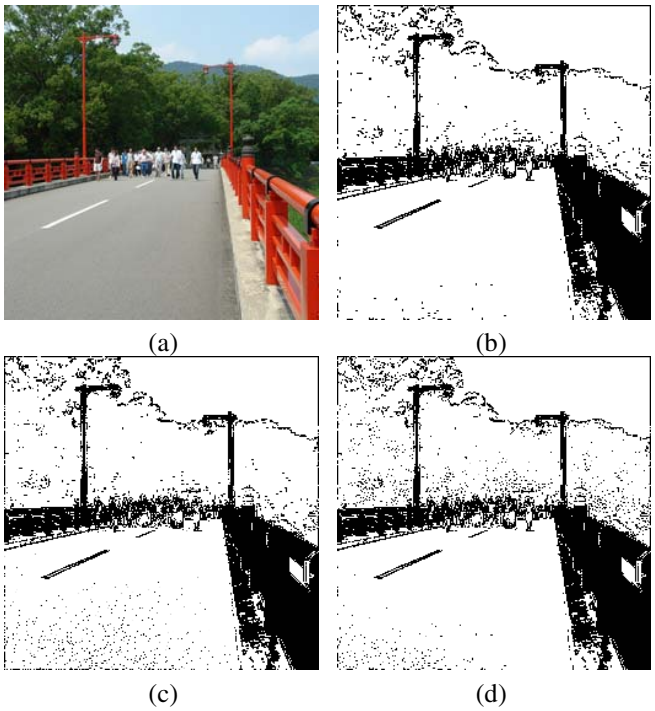

Fig. 9. Task 2a (bridge): Color edge detection using template (22) in HSB-CNN. (a) Input color image. (b) Output image of red layer ("Grayscale to Binary Threshold" template). (c) Output image of green layer ("Edge Detection" template). (d) Output image of blue layer ("Edge Detection" template).

Figure 9 shows the simulation results of color edge detection of "bridge" image using template (22) in HSB-CNN. In this simulation, we set the "Grayscale to Binary Threshold" template to the red layer, and "Edge Detection" template to the green and the blue layers. In Fig. 9, we can see that the output images of all layers are slightly different. In Figs. 9(c) and (d), the region of "road" have some noise. However, by the noise removal effect of the coupling templates, the output image of the red layer Fig. 9(a) has few noise. From these results, we can say that the output image from the layer with "Grayscale to Binary Threshold" template in HSB-CNN has few noise and becomes the optimal output image in edge detection.

Figure 10 shows the effect of changing the layer with "Grayscale to Binary Threshold" template. Namely, we set that the one of three layers has "Grayscale to Binary Threshold" template and the other two layers have "Edge Detection" template. Figure 10(b) shows the simulation result when the red layer has "Grayscale to Binary Threshold" template. Hence, the result is the same as Fig. 9(b). Figures 10(c) and (d) show the simulation results when the green and the blue layers have "Grayscale to Binary Threshold" template, respectively. We can see that green region and blue region can be detected respectively and the edges of color input image can be detected. Also, output images have few noises by the noise removal effect of the coupling templates.

Figure 11 shows the simulation results for another color image "Venice". We can confirm that the regions of red, green and blue can be successfully detected and edges of color input image can be detected in all output images.

C. Task 3: Color parts extraction

In this subsection, we apply the results in the previous subsections (Task 1: Color binarization or Task 2: Color edge detection) to extract the color parts. Some images may have regions as sky, sea, forest, human, building and so on, and one may need to extract only these specific regions. Regardless of the purposes, extracting some regions of images is one of the most important tasks in image processing. When human recognize some regions, informations of texture, color and perspective of objects are considered. In these information, color is easier to recognize than other information. For example, we consider the region of sea as blue region instead of wave pattern, the region of forest is green instead of the shape of a tree.

Figures 12 and 13 show the simulation results obtained from the results of Figs. 10 and 11 (Task 2: Color edge detection). For example, Fig. 12(b) is the image extracted from the original input image Fig. 10(a) such that only the pixels corresponding to the black pixels in Fig. 10(b). As a result, Fig. 12(b) includes only the red part "bridge handrails" and all the edges of the input image. Similarly, Figs. 12(c) and 12(d) include the green part "forest" and the blue part "sky" with all the edges, respectively.

Next, Fig. 14 shows the simulation results obtained from the results of Fig. 8 (Task 1: Color binarization). In this

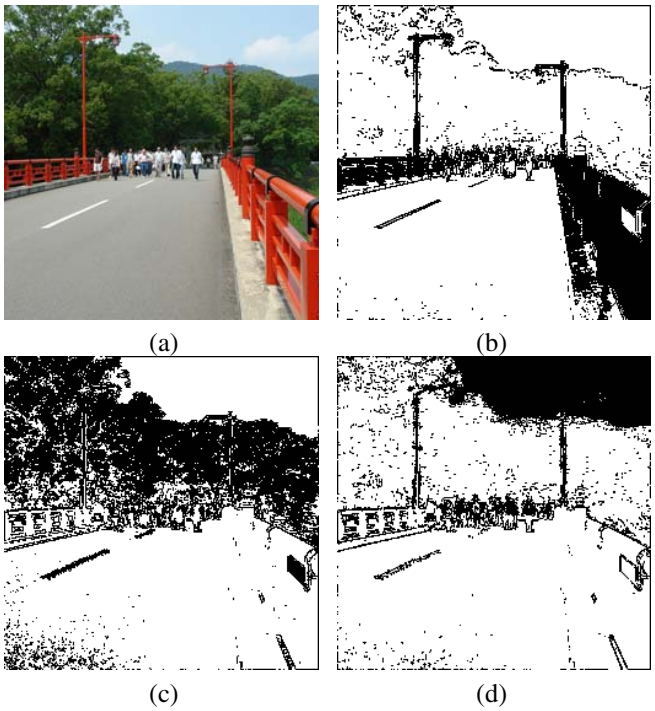

Fig. 10. Task 2b (bridge): Color edge detection with different template combinations. (a) Input color image. (b) Output image (red layer: template (19), green and blue layers: template (21)). (c) Output image (green layer: template (19), red and blue layers: template (21)). (d) Output image (blue layer: template (19), red and green layers: template (21)).

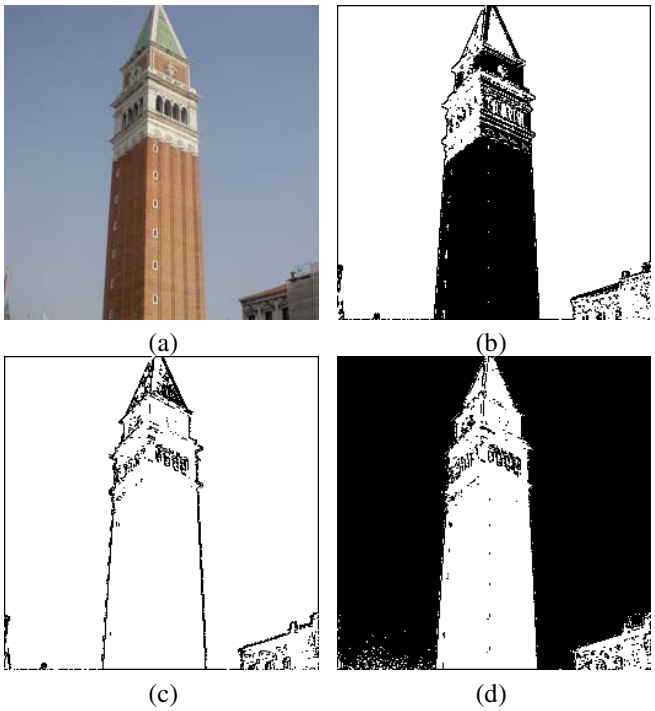

Fig. 11. Task 2b (Venice): Color edge detection with different template combinations. (a) Input color image. (b) Output image (red layer: template (19), green and blue layers: template (21)). (b) Output image (green layer: template (19), red and blue layers: template (21)). (b) Output image (blue layer: template (19), red and green layers: template (21)).

simulation, because we use the results of Fig. 8, the results include parts near three pure colors. Also, only small regions very close to pure colors remain when the threshold value is small.

V. Conclusions

In this study, we have proposed a three-layer CNN dealing with color images considering the HSB model. The principle of the HSB model is similar to the perception of human. We realize the CNN with the structure similar to the human's recognition method by considering the HSB model. By computer simulation of some processing tasks for color images, we have confirmed that the HSB-CNN is effective for color image processing. Although this article reports only some examples of applications, we consider that the HSB-CNN can be available to various color image processing.

Acknowledgment

The authors would like to thank Prof. Kristina Kelber of University of Applied Sciences Dresden for her valuable comments.

References

[1] L.O. Chua and L. Yang, "Cellular Neural Networks:Theory," IEEE Trans. Circuits Syst., vol. 32, pp. 1257-1272, Oct. 1988.
[2] F. Dirk and T. Ronald, "Coding of Binary Image Data using Cellular Neural Networks and Iterative Annealing," Proc. of ECCTD'03, vol. 1, pp. 229-232, Sep. 2003.
[3] M. Namba and Z. Zhang, "Cellular Neural Network for Associative Memory and Its Application to Braille Image Recognition," Proc. of IJCNN'06, pp. 4716-4721, Jul. 2006.
[4] H. Koeppl and L.O. Chua, "An Adaptive Cellular Nonlinear Network and its Application," Proc. of NOLTA'07, pp. 15-18, Sep. 2007.
[5] T. Kozek, K.R. Crounse, T. Roska and L.O. Chua, "Smart Image Scanning Algorithms for the CNN Universal Machine," Proc. of NOLTA'95, vol. 2, pp. 707-712, 1995.
[6] J. Kishida, C. Rekeczky, Y. Nishio and A. Ushida, "Feature Extraction of Postage Stamps Using an Iterative Approach of CNN," IEICE Trans. on Fundamentals, vol. E79-A, no. 10, pp. 1741-1746, Oct. 1996.
[7] K.R. Crounse and L.O. Chua, "Methods for Image Processing and Pattern Formation in Cellular Neural Networks: A Tutorial," IEEE Trans. Circuits Syst., vol. 42, no. 10, pp. 583-601, Oct. 1995.
[8] K.R. Crounse, L.O. Chua, P. Thiran and G. Setti, "Characterization and Dynamics of Pattern Formation in Cellular Neural Networks," International Journal of Bifurcation and Chaos, vol. 6, no. 9, pp. 1703-1724, 1996.
[9] C.T. Lin, C.H. Huang and S.A. Chen, "CNN-Based Hybrid-Order Texture Segregation as Early Vision Processing and Its Implementation on CNN-UM" IEEE Trans. Circuits Syst., vol. 54, no. 10, pp. 2277-2287, Oct. 2007.
[10] T. Roska, A. Zarandy and L.O. Chua, "Color Image Processing by CNN," Proc. of ECCTD'93, pp. 57-62, Aug. 1993.
[11] T. Inoue and Y. Nishio, "Edge Enhancement of Color Image by Three-Layer Cellular Neural Network Considering Three Primary Colors" Proc. of NOLTA'08, pp. 540-543, Sep. 2008.
[12] Cellular Sensory Wave Computers Laboratory Computer and Automation Research Institute Hungarian Academy of Sciences, "Cellular wave Computing Library (Template, Algorithms, and Programs) Version 2.1"

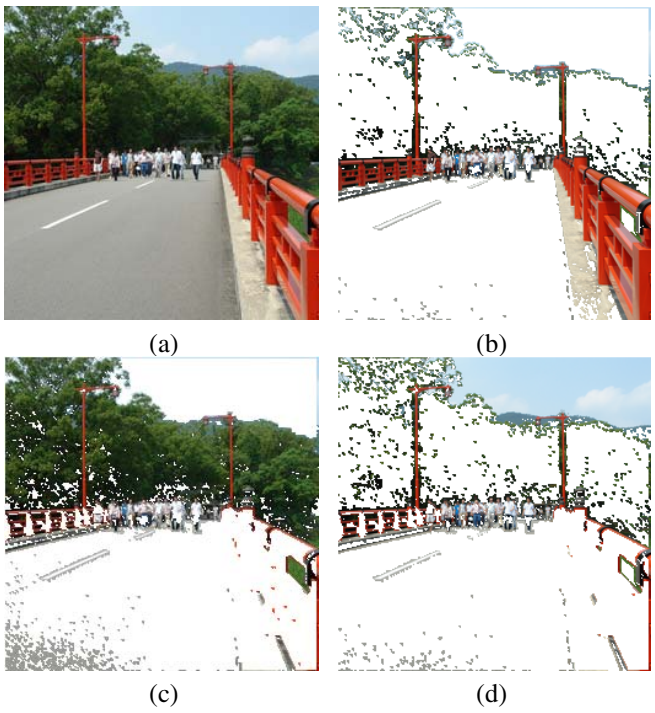

Fig. 12. Task 3 (bridge): Color parts extraction from images in Fig. 10. (a) Input color image. (b) Red color extraction using simulation result of Fig. 10(b). (c) Green color extraction using simulation result of Fig. 10(c). (d) blue color extraction using simulation result of Fig. 10(d).

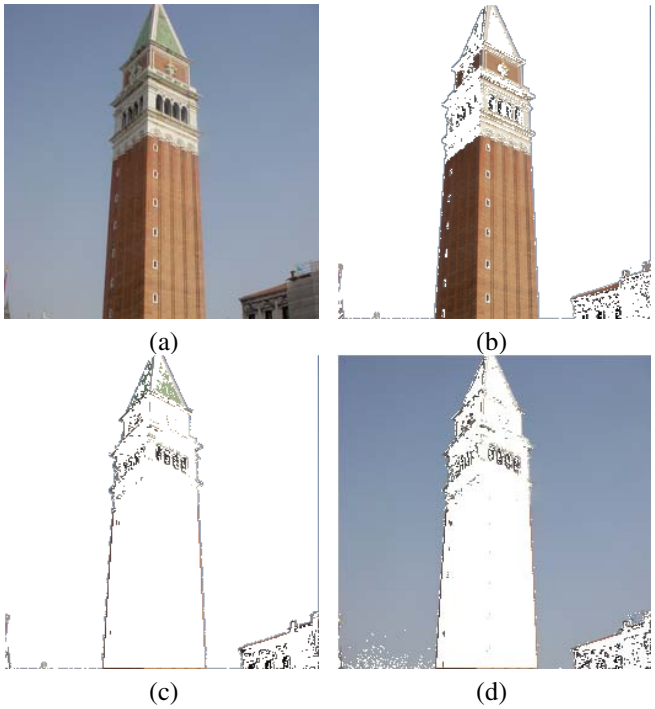

Fig. 13. Task 3 (Venice): Color parts extraction from images in Fig. 11. (a) Input color image. (b) Red color extraction using simulation result of Fig. 11(b). (c) Green color extraction using simulation result of Fig. 11(c). (d) Blue color extraction using simulation result of Fig. 11(d).

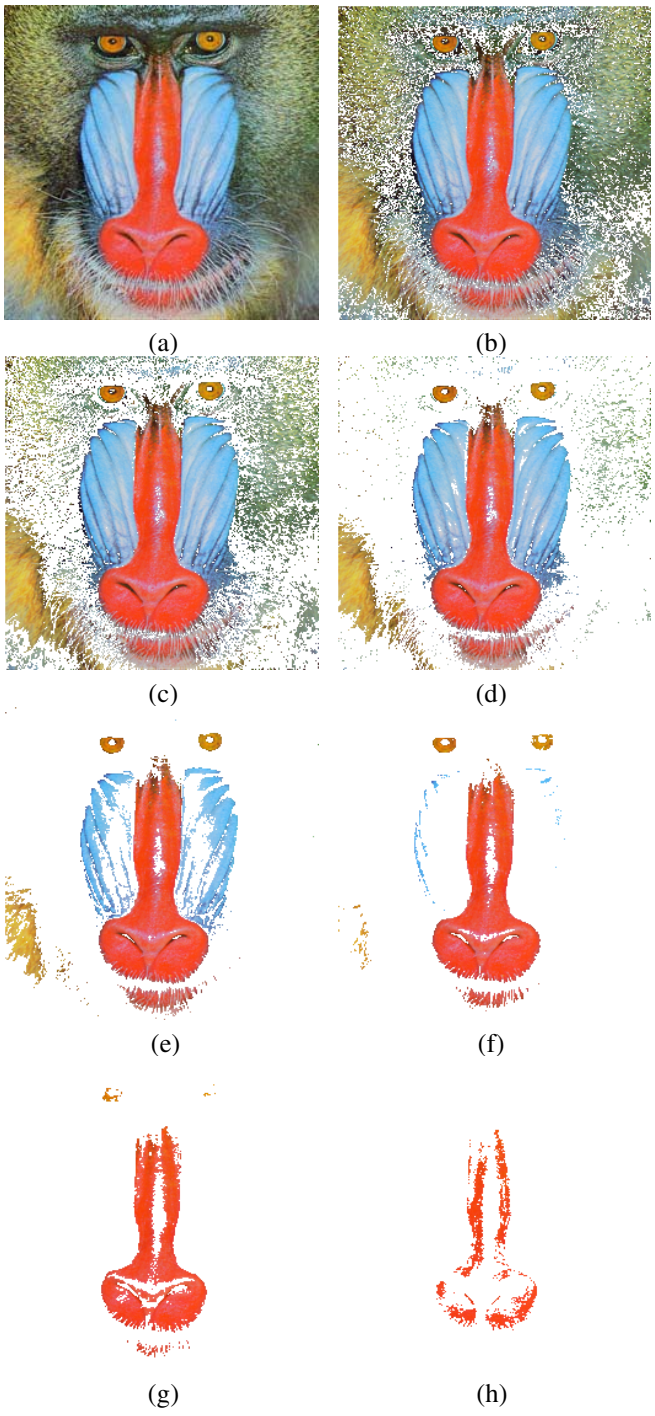

Fig. 14. Task 3 (mandrill): Color parts extraction from images in Fig. 8. (a) Input color image. (b) Extraction result using simulation result of Fig. 8(b). (c) Extraction result using simulation result of Fig. 8(c). (d) Extraction result using simulation result of Fig. 8(d). (e) Extraction result using simulation result of Fig. 8(e). (f) Extraction result using simulation result of Fig. 8(f). (g) Extraction result using simulation result of Fig. 8(g). (h) Extraction result using simulation result of Fig. 8(h).