A Unified Approach for Image Style Generation

Yuichi Nitto Koji Nakamaru Yoshio Ohno
Graduate School of Science and Technology, Keio University
3-14-1 Hiyoushi, Kohoku-ku, Yokohama-shi, Kanagawa 223-8522
email:{nitto | maru | ohno}@on.ics.keio.ac.jp

Abstract

There are many research for transfering a photographic image into an image with a certain style of paintings. Most of such research, however, simulate one specific style. We present a unified approach for image style generation.

We propose an algorithm for automatic generation of a transformed image by giving two images: an original image to be transformed and a reference image. The most dominant factors that influence the impression of an images include the color distribution and the style of the edges. The proposed algorithm checks the color distribution of the original image and transforms the colors so that their distribution approaches to that of the reference image. The algorithm also checks the edge style of the reference image and apply that style to the original image.

Our algorithm consists of two parts: the color handling part and the edge handling part. In the color handling part, we make the histograms of the original image and the reference image in the YUV color space. Then by shifting the histograms, we change the colors of original images uniformly.

In the edge handling part, the edge magnitude of the original image is converted using the edge magnitude histogram of the reference image. We use the most frequently appeared HLS color in the edge pixels as the edge color of the original image.

For each pixel of the original image, either the transformed color or the decided edge color is chosen using the specified threshold values.

We succeeded in the automatic generation of the image with a specified style by giving only two images. By giving various reference images, we can obtain images with various styles from one original image automatically.

1 Introduction

The main research goal of Computer Graphics has been the Photorealistic Rendering (PR) for making a realistic image like a photograph.

In the recent years a lot of resarch is done for Nonphotorealistic Rendering (NPR) to obtain an image that expresses information or feeling in nonphotorealistic ways.

The difference between PR and NPR corresponds to the difference between a photographic image and a painted image. PR is suitable for the purpose to express some shape or texture as it is. On the other hand, NPR is suitable for the purpose to emphasize or to simplify information as you like. Therefore, NPR has advantages over PR for transmitting information in some cases.

The techniques of NPR can be divided into two categories: techniques using a 2D image as input, and techniques using a 3D model as input. NPR techniques can also be divided into interactive techniques and automatic generation algorithms.

Many NPR algorithms simulate some traditional painting process, e.g., oil painting, water-color drawing, pen-and-ink drawing and woodcut printing. Some NPR technique sticks to a style of some specific artist, e.g., Gogh and Renoir.

However, many NPR algorithms can simulate only one style. For each artist or for each style, one dedicated algorithm must be developed. Also, many NPR algorithms focus on the generation of brush strokes and ignore the "color style" of the artist.

We present an automatic method to convert a photo image (an original image) into an image with different feeling specified by other image (a reference image) [8].

2 Related Works

Some techniques generate an image based on 3D model. Praun et al.[1] present a technique for rendering a 3D model by "hatching" instead of shading. Lake [2] studys an algorithm that renders a 3D model with "toon" shading.

Kalnins et al.[3] present an interactive NPR technique. They realize real-time rendering of a 3D model by drawing strokes directly on the model displayed on a monitor by a pointing tool such as a tablet.

Hertzmann [5] gives an algorithm to generate a paint image from a 2D image drawn with brush strokes. Artists draw a lot of brush strokes for obtaing a drawing. Hertzmann simulates this process on the computer display. His system expresses the brush strokes as spline curves with various lengths. Thick short strokes are used for rough sketches, and the detailed parts are drawn with thin long strokes. The input image is approximated sucessively by decreasing the theickness of the strokes. This technique succeeded the simulation of the painting style with brush strokes, but it cannot be applied to the styles that have no brush strokes such as woodcut printing like "Ukiyoe."

There are research for the NPR algorithms based on 2D images. Curtis et al.[4] simulate the style of water-color painting. Salisbury et al.[6] simulate the style of pen-and-ink drawing. These methods generate excellent drawings but they are restricted to certain style of paintings.

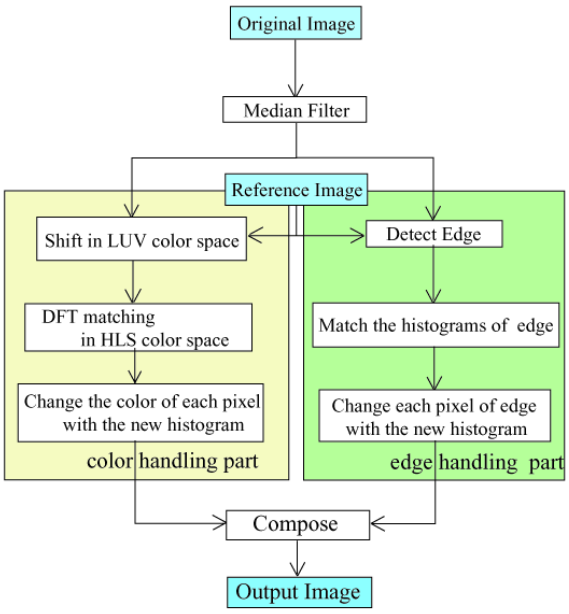


Figure 1: the proposed algorithm

Chang et al.[7] study a system for converting the colors of an image. Their research is based on a 2D input image and focuses on the colors. In their system, colors of the input image are classified into eleven categories manually. Then the colors in each category are converted based on the color distribution in that category. This method sometimes converts a color in unintended way of the user mainly because the unsuitable classification of the color. Also this research takes no consideration on edges.

3 Paint-Image Generation

As described in the previous section, most NPR algorithms simulate only one particular style of painting. Our system simulates various styles. When two images (an original image and a reference image) and some threshold values are given, it converts the original image into an image with the contents of the original image and with the style of reference image.

The process of our algorithm is summarized in Fig. 1.

The algorithm is divided into two main parts: the color handling part and the edge handling part.

In the color handling part, intensity histograms of the original and reference images are constructed for each component Y, U and V in the YUV color space. Then, by blending the histograms of the original image with those of the reference image, we obtain new histograms. The colors of the original pixels are transformed based on the new histograms.

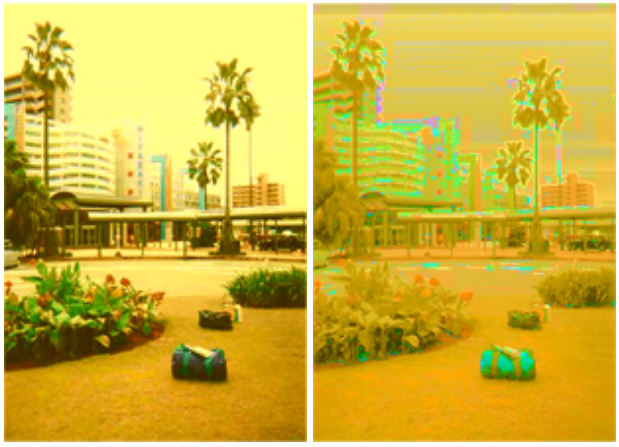
In the edge handling part, we detect the edges in the original and reference images. By using the histograms of edge magnitude, the edge expression of the original image are changed to approximate the edge expression of the reference image.

3.1 Color Handling Part

In this section, we describe the algorithm of color handling part.

- Construct the histograms of the colors of the original and reference images in the YUV color space. The histograms are shifted so that the medians of the histograms match.
- Blend the DFTs of the original image histograms with DFTs of the reference image histograms using the user spcified parameters in the YUV color space.
- Applying DIFT to the blended DFTs, a new histogram is obtained. Each HLS color in the original image is transformed based on the histogram.

3.1.1 Consideration on the Color Spaces



(a) YUV (b) HLS

Figure 2: Difference between YUV and HLS color spaces.

In our algorithms, we use YUV and HLS color spaces. In the YUV color space, Y is luminance while U and V are the color components. HLS color model represents a color as a combination of Hue (H), Lightness (L) and Saturation (S). These color spaces are more suitable for the expression of human feeling than the RGB color space.

In Fig. 2, YUV and HLS color spaces are compared. We transform the image like Gogh's "Sunflower" using the YUV space histograms only (Fig. 2(a)) and using the HLS color space histograms only (Fig. 2(b)). The image transformed in YUV is more yellowish than in HLS, while the image in HLS is more vivid as oil painting than in YUV. It is important to have both features such as "totally and uniformly yellowish" and "vivid", so we decided to use YUV for histogram shifting and HLS for one blending.

3.1.2 Transformation of Colors

Our technique for transforming the colors along a histogram follows the algorithm of histogram equalization. As shown in Fig. 3, we sort the pixel intensities in the original image into the increasing order, and map the pixel values to the intensity values based on the frequency in the new histogram.

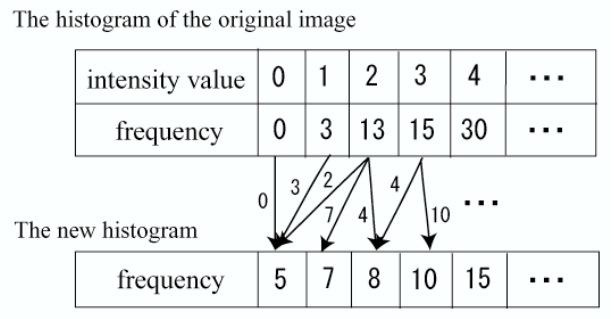


Figure 3: Transformation of the colors based on a histogram.

3.1.3 Histogram Shifting in YUV

The histogram should be shifted in YUV color space before blending, because we aim at the effect of total and uniform color modification. For example, the histogram of Y component will be shifted as follows: let Md_{inY} be the median of original image histogram, Md_{tarY} be that of the reference image, and α_Y be a user-specified parameter. Then the shift distances S_{inY} and S_{tarY} of the histograms are given by Equation (1):

$$\begin{cases} S_{inY} = \alpha_Y (Md_{tarY} - Md_{inY}) \\ S_{tarY} = (1 - \alpha_Y)(Md_{inY} - Md_{tarY}) \end{cases} \quad (1)$$

3.1.4 Histogram Blending in HLS

We want to transform the colors so that the histogram of the original image resembles to that of the reference image. But this approach sometimes produces undesirable result. For example, such situation happens when the original image has "blue" pixels but the reference images doesn't have them. To avoid such cases, we transform the color of the original image based on the blended histogram, not on the original histogram.

We apply DFT to the frequency distributions of the histograms using:

$$F(n) = \sum_{k=0}^{N-1} x(k) e^{-j\frac{2\pi}{N}nk} \quad (2)$$

where $x(k)$ is the frequency of intensity value k.

Then we blend the DFTs of the original image histogram and the reference image histogram. For the H component, for example, let $F_{inH}(n)$ be the DFT of the original image, $F_{refH}(n)$ be that of the reference image, $F_{outH}(n)$ be that of the image to be generated, and α_H be a user-specified parameter value. The following formula is used for the blending:

$$F_{outH}(n) = (1 - \alpha_H) F_{inH}(n) + \alpha_H F_{refH}(n) \quad (3)$$

Finally, we get a new histogram as the DIFT of $F_{outH}(n)$ obtained above. Based on the new histogram, the colors of the pixels in the original images are transformed.

3.2 Edge Handling Part

The algorithm for the edge handling part can be summarized as follows:

- Transform the both images to grayscale.
- Detect the edge of both images using Kirsch operators.
- Decide the color to be used for the edges in the transformed image.
- Modify the histogram of edge magnitude of the original image so that it matches to the similar histogram of the reference image.

3.2.1 Edge Magnitude

We detect the edge magnitude G using Kirsch operator. Kirsch operators detect features of eight directions using a 3×3 kernel.

We normalize the detected edge magnitude values G to the range $(0, 255)$ by

$$G' = 255 \times (G - G_{min})/(G_{max} - G_{min}) \quad (4)$$

where, G_{min} and G_{max} are the minimum and the maximum of G, respectively.

3.2.2 Decision of Edge Line Color

In the previous section, we obtained the edge magnitude in the original image. Then we decide the color that is used for the edge lines in the transformed image.

We search for the pixels that have higher edge magnitude than a user-specifed threshold value th_{color} and preserve the colors of such pixels. We make histograms in HLS color space on these pixels and use the most frequently used value as the edge line color in the transformed image.

3.2.3 Matchimg of the Edge Magnitude Histograms

Some painting styles draw edge lines clearly, others do not. Therefore we detect the edge line style in the reference image and simulate it in the transformed image.

In our method, to simulate the edge line style, we modify the edge magnitude distribution in the original image so that it matches to the distribution of the reference image.

For this purpose, different from the color handling part, we simply transform the edge magnitude in the original image along the histogram of edge magnitude in the reference image.

3.3 Composition of Color Parts and Edge Parts

In order to compose the image made in the color handling part with that in the edge handling part naturally, we use two thresholds th_{min} and th_{max} specified by the user. Let g' be the normalized value of G, C_{line} be the color of edge line, and C_{color} be the color of a pixel obtained in the color handling part. The following color composition function is used to obtain the color C_{out} of a pixel:

$$C_{out} = \begin{cases} C_{color} & (0 \le g' < th_{min}) \\ (1 - \beta) C_{color} + \beta C_{line} & (th_{min} \le g' \le th_{max}) \\ C_{line} & (th_{max} < g' \le 1) \end{cases} \quad (5)$$

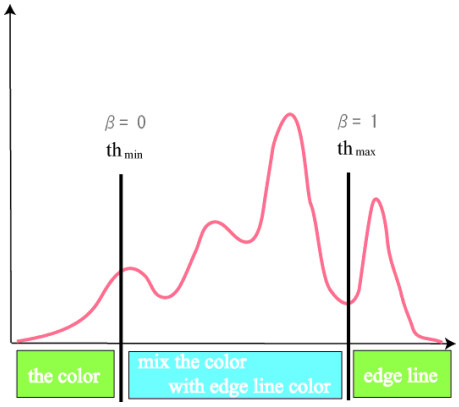


Figure 4: Composition of color parts and edge parts.

Where

$$\beta = \frac{g' - th_{min}}{th_{max} - th_{min}} \tag{6}$$

As shown in Fig. 4, we decide the color of pixel using th_{min} and th_{max}. If g' of a pixel is smaller than th_{min}, we keep the color of the pixel. If g' is larger than th_{max}, we use the edge line color. Otherwise, we blend the color of the pixel with the edge line color.

The effect of the thresholds can be stated as follows.

- When th_{min} is large, the color is emphasized; when small, the edge is emphasized.
- When th_{max} is large, the edge line are drawn roughly; when small, they are drawn in detail.

4 Results

We show the original images in Figs. 5 and 6, and the generated images in Figs. 7, 8, 9 and 10. Gogh's "Sunflower" is used as the reference image for Fig. 7. The reference image for Fig. 8 is a water painting, the reference image for Fig. 9 is a Ukiyoe painting, and the reference image for Fig. 10 is Millet's "Les Glaneuses".

As seen in Fig. 7, the generated image is "totally and uniformly yellowish" as Gogh's masterpiece. This is the effect of shifted histogram in YUV color space. Also, Fig. 8 is generated with the feature of water paintings. This effect is from the blended histogram in HLS color space. The emphasis on the edge line in Fig. 9 results from both parameters, th_{min} and th_{max}, set lower. For Fig. 10, we set the parameters α_H, α_L and α_S smaller so that the image is changed to be somewhat dark. In other words, we successfully keep down the effect of oil paintings by choosing smaller values for the parameters.

Figure 5: Original image 1.

Figure 6: Original image 2.

Figure 7: Result from the reference image "Sunflowers" by Gogh : $\alpha_H = 0.4$, $\alpha_L = 0.9$, $\alpha_S = 0.9$, $th_{color} = 0.92$, $th_{min} = 0.06$, $th_{max} = 0.53$.

Figure 8: Result from a water painting : $\alpha_H = 0.3$, $\alpha_L = 0.9$, $\alpha_S = 0.9$, $th_{color} = 0.92$, $th_{min} = 0.22$, $th_{max} = 0.80$.

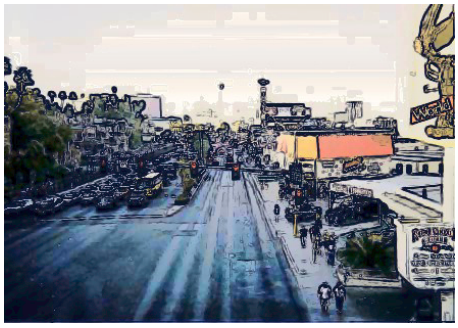


Figure 9: Result from a reference Ukiyoe painting: $\alpha_H = 0.7$, $\alpha_L = 0.9$, $\alpha_S = 0.9$, $th_{color} = 0.92$, $th_{min} = 0.06$, $th_{max} = 0.41$.

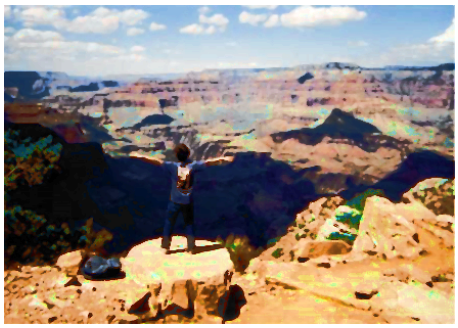


Figure 10: Result from a reference image “Les Glaneuses” by Millet : $\alpha_H = 0.1$, $\alpha_L = 0.3$, $\alpha_S = 0.3$, $th_{color} = 0.92$, $th_{min} = 0.22$, $th_{max} = 0.61$.

5 Discussion

We realize a system for modifying the original image so that the overall feeling of the image becomes similar to that of a reference image. The effect of the algorithm can be controlled by the adjustment of the threshold values. The algorithm proceeds automatically. The amount of computation is not large, and each image can be obtained within a minute. The obtained images show the feeling of the reference image faithfully.

Our algorithm could be improved by taking the information on the relation between each pixel and its neighborhood in consideration. In addition, by including some mechanism for the determination of suitable parameter values, the usefulness of our algorithm will be increased.

Acknowledgements

This research is partly supported by Keio University Special Grant-in-Aid for Innovative Collaborative Research Projects.

References

[1] Emil Praun, Matthew Webb, Adam Finkelstein, Hugues Hoppe : Real-Time Hatching, Proceedings of SIGGRAPH 2001, pp. 581–586.

[2] Adam Lake : Stylized Rendering Techniques for Scalable Real-Time 3D Animation, Proceedings of NPAR 2000, pp. 13–20.

[3] Robert D. Kalnins, et al. : WYSIWYG NPR: Drawing Strokes Directly on 3D Models, Proceedings of SIGGRAPH 2002, pp. 755–762.

[4] Cassidy J. Curtis, et al. : Computer Generated Watercolor, Proceedings of SIGGRAPH 97, pp. 421–430.

[5] Aaron Hertzmann : Painterly Rendering with Curved Brush Strokes of Multiple Sizes, Proceedings of SIGGRAPH 98, pp. 453–466.

[6] Michael P. Salisbury, et al. : Orientable Textures for Image Based Pen-and-ink Illustration, Proceedings of SIGGRAPH 97, pp. 401–406.

[7] Youngha Chang, Suguru Saito, Masayuki Nakajima : A Framework for Transfer Colors Based on the Basic Color Categories, 2003 Computer Graphics International, pp. 176–181.

[8] Tomoko Ueno, Yuichi Nitto, Koji Nakamaru, Yoshio Ohno : A Unified Approach for Paint-Image Generation, Proc. ITE Winter Annual Convention 2003, pp.72(in Japanese).