Fundamental Studies on Non-Photorealistic Rendering

by

Qiang LI

Doctoral Dissertation

SUBMITTED FOR THE DOCTOR DEGREE OF ENGINEERING
TO THE DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING AND THE COMMITTEE ON GRADUATE STUDIES
OF NAGOYA INSTITUTE OF TECHNOLOGY

2007

Contents

List of Figures

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor, Professor Hidenori Itoh of the Department of Computer Science and Engineering at the Nagoya Institute of Technology for his constant academic guidance, indispensable help, and invaluable inspiration over the past three years.

I am especially grateful to Assistant Professor Tsuyoshi Nakamura of the Department of Computer Science and Engineering at the Nagoya Institute of Technology, for his invaluable ideas, important comments, academic guidance, and generous assistance during this work.

I am very appreciative of the constant guidance, support, encouragement, and help given by Assistant Professor Lifeng He of the Faculty of Information Science and Technology at Aichi Prefectural University, and that by Assistant Professor Yuyan Chao of the Faculty of Environment, Information, and Business at Nagoya Sangyo University throughout the past three years. I would like to thank the Japanese Ministry of Education for supplying me with the government scholarship that made it possible for me to complete this research.

I extend my heartfelt gratitude to Dr. Masayoshi Kanoh who is an Assistant Professor in the School of Life System Science and Technology at Chukyo University, and his many colleagues at the Itoh Laboratory who helped me a great deal in different areas. I would especially like to thank Messrs. Tuyosi Kawajima, Kuwayama Kiyotake and Noritake Koushirou.

I am also very grateful to Professor Sawari Nakagawa, for her many kindnesses, help, and suggestions during my time in Nagoya.

Finally, I would like to thank my wife Rong Zhang, my daughter Sitong Li, and all my relatives and friends who gave me much happiness and support. Without their unwavering support, love, and understanding, I would never have finished this work.

ABSTRACT

There are many computer-generated images around us, in movies, advertising, or on the Internet that are already being taken for granted, and what impresses most people is their photorealistic quality. A picture, as we have often been told, is worth a thousand words and the information conveyed by an image can have many different forms.

Non-photorealistic rendering (NPR) is an alternative to realistic depiction and is defined by what it is not, i.e., it concentrates less on the process and more on communicating the content of an image, bringing art and science closer together. Techniques that have long been used by artists can be applied to computer graphics to emphasize subtle attributes, and to omit extraneous information.

I developed a method of rendering images that look similar to those found in painterly art. When artists create painterly art, they express their own feelings and sensitivities depending on their own style. These styles can be distinguished by various elements, such as motifs, colors, deformations in shape, and texture. Many computer graphics (CG) researchers are interested in these characteristics and have devoted a great deal of time to developing techniques that will produce NPR.

CG researchers have explored many NPR algorithms, e.g., several methods of pasting textures from painterly art onto photographs have been suggested. Other methods of applying the coloration from painterly art to photographic images have also been put forward. Most previously proposed methods have used some absolute criteria, e.g., certain mathematical formulae to establish the nearest distance in *CIE L*a*b* color space to uniquely convert one color to another. However, I wanted to develop a method that could convert color depending on an individual's personal feelings and sensitivities, like a painter who creates pictures in the real world.

I present an algorithm in this thesis for altering the colors of a photograph using reference images obtained from painterly art. Color transfer is one of major themes of NPR, and it is one of the major tasks confronting CG designers. Examples of color transfer are changing a blue sky into a sunset and changing a dark skin color into a lighter shade. Although Adobe Photoshop is a popular tool utilized by designers to transfer colors, it does not work automatically. They have to manually transfer the colors themselves.

My algorithm can produce color transformations resulting from two input images, i.e., a target photograph and a reference painting. My method is based on interactive evolutionary computation (IEC) and it can produce a variety of images with transformed colors. The designer interactively selects various candidates from this variety until a final result is reached, which looks similar to the reference painting. I also discuss the exploitation of image-region matching for color transfer in this thesis.

In the following, I will explain the image-region matching algorithms and present some results on transferring color from reference paintings to target photographs to obtain resulting images. There is also a discussion on results obtained for the searching efficiency of the algorithms.

In the last chapter, I will also discuss my recent study on generating a calligraphic font that creates a scratched and blurred impression that was obtained using a reference font that was designed by a famous calligrapher. I will discuss the algorithm for generating the calligraphic font by transferring texture from a calligraphic font to a target font.

CHAPTER 1
INTRODUCTION

1.1. BACKGROUND

Wikipedia states: “The term “non-photorealistic rendering” [NPR] was probably coined by David Salesin and Georges Winkenbach in a 1994 paper.”

It also states: “The first conference on Non-Photorealistic Animation and Rendering included a discussion of possible alternative names. Among those suggested were “expressive graphics”, “artistic rendering”, “non-realistic graphics”, “art-based rendering”, and “psychographics”. All of these terms have been used in various research papers on the topic, but the term NPR seems to have none-the-less taken hold.”

Fig. 1 Photorealistic image

Fig. 2 Painterly art

Fig. 3 Non-photorealistic result

NPR is an alternative to photorealism. For example, Fig. 1 is a common photorealistic image, Fig. 2 is painterly art, and Fig. 3 is NPR produced from Fig. 1 using the reference of Fig. 2. We can see similarities between the reference painterly art and NPR. Figure 1 is a landscape in summer and Fig. 2 is a landscape in winter. We can see NPR can turn a summer landscape into a winter scene.

NPR has recently become an important field of tremendous interest in the graphics community with many papers on topics such as rendering with simulated watercolors, creating images in an impressionist style, and automatically extracting silhouettes. For example, some researchers have developed a method of applying the coloration of painterly art to photographic images. This method has been developed using certain mathematical formulae. The color space is automatically segmented into 11 categories based on the experimental results for the ratings of basic color categories for each test color. Then, for every pixel color value in the photograph, the algorithm finds its corresponding color in the reference painting. We can therefore produce an image whose color features are similar to those of the reference painting. [Proceedings of the Computer Graphics International (CGI'03) 1530–1052/03 @2003 IEEE]

Another method of color transfer from image-to-image is to process within a special color space; Ruderman et al. developed a color space, called $l\alpha\beta$, which minimizes the correlation between channels for many natural scenes. This space is based on research on data-driven human perception that assumes the human visual system is ideally suited to processing natural scenes. There is little correlation between the axes in $l\alpha\beta$ space, which lets us apply different operations in different color channels with a degree of confidence that objectionable cross-channel artifacts will not occur. Additionally, this color space is logarithmic, which as a first approximation means that uniform changes in channel intensity tend to be equally detectable. The color-transfer method is a simple algorithm for the $l\alpha\beta$ color space. As it is based on statistics, the mean and standard deviations along each of the three axes suffice. These measures for both the source and target images must be computed. It needs to be noted that the means and standard deviations for each axis are separately computed in the $l\alpha\beta$ color space.

First, the mean must be subtracted from the data points:

$$l^* = l - \langle l \rangle$$

$$\alpha^* = \alpha - \langle \alpha \rangle$$

$$\beta^* = \beta - \langle \beta \rangle$$

Then, Ruderman et al. scaled the data points comprising the synthetic image by factors determined by the respective standard deviations:

$$l' = \frac{\sigma_t^l}{\sigma_s^l} l^*$$

$$\alpha' = \frac{\sigma_t^\alpha}{\sigma_s^\alpha} \alpha^*$$

$$\beta' = \frac{\sigma_t^\beta}{\sigma_s^\beta} \beta^*$$

After this transformation, the resulting data points have standard deviations that conform to the photograph. Next, instead of adding the averages that they previously subtracted, they added the averages computed for the photograph. Finally, they converted the results back to RGB color space.

This method of color transfer between images is simple statistical analysis to impose one image's color characteristics on another in the $l\alpha\beta$ color space. I also intended to apply this method to my own algorithm for transferring colors from painterly art references to target photographs.

There is an important branch in the field of NPR that involves the transfer and synthesis of textures. Efros and Freeman presented a simple image-based method of generating a novel visual appearance in which a new image is synthesized by stitching together small patches of existing images. This process is called image quilting. They used quilting as a fast and very simple algorithm for synthesizing texture that could produces surprisingly good results. They then extended the algorithm to render the texture of one object with a texture taken from a different object. Their algorithm for synthesis was not one-pixel-at-a-time being something more than a single pixel, i.e., a "patch". The process of synthesizing texture would be akin to putting together a jigsaw puzzle, quilting together the patches, and making sure they all fitted together. Determining precisely what the patches for a given texture are and how they are put

together still remains unsolved. I will illustrate this method of “image quilting” in what follows.

B_i is defined as the unit of synthesis, being a square block of user-specified size in the input image. The next step is to introduce some overlap in the placement of blocks onto the new image, instead of picking a random block and searching its input image using some measure that agrees with its neighbors along the region of overlap. However, the edges between the blocks are still quite noticeable. Efros and Freeman then smoothed across these edges more systematically.

They let the blocks have ragged edges, which allowed them to better approximate features in the texture. Before placing a chosen block into the texture, they looked at the error in overlap between it and the other blocks. They found the minimum cost path through that error surface and declared it to be the boundary for the new block.

The minimum error-boundary cut is to make an incision between two overlapping blocks on pixels where two textures match the best (i.e., where overlap error is low). This can easily be done with dynamic programming (Dijkstra’s algorithm can also be used) [J. Davis. Mosaics of scenes with moving objects. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.,* 1998]

The minimal cost path through the error surface is computed as follows. If B_1 and B_2 are two blocks that overlap along their vertical edges with overlap regions of B_1^{ov} and B_2^{ov}, then the error surface is defined as e = $(B_1^{ov} - B_2^{ov})^2$. To find the minimal vertical cut through this surface, we traverse e ($i = 2..N$) and compute the cumulative minimum error, E, for all paths:

$$E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$$

The minimum value of the last row in E indicates the end of the minimal vertical path though the surface and one can trace back and find the path for the best cut. A similar procedure can be applied to horizontal overlaps. When there is both vertical and horizontal overlap, the minimal paths meet in the middle and the overall minimum is chosen for the cut.

The complete quilting algorithm is as follows:

- Go through the image to be synthesized in raster scan order in steps of one block (minus the overlap)
- Search the input texture for a set of blocks for every location that satisfies the overlap constraints (above and left) within some degree of error tolerance. Randomly pick one such block.
- Compute the error surface between the newly chosen block and the old blocks in the overlap region. Find the minimum cost path along this surface and make that the boundary for the new block. Paste the block onto the texture. Repeat.

The size of the block is the only parameter controlled by the user and it depends on the properties of a given texture, i.e., the block must be big enough to capture the relevant structures in the texture, but small enough so that the interaction between these structures is left up to the algorithm.

This method has been applied to the transfer of textures, because the image-quilting algorithm selects output patches based on local-image information. They just augment the synthesis algorithm by requiring each patch to satisfy a preferred *correspondence map*, as well as satisfy the requirements for texture synthesis. The correspondence map is a spatial map if there is some quantity corresponding to the texture-source and controlling-target images. That quantity could include image intensity, blurred-image intensity, local-image orientation angles, or other derived quantities.

In many cases, the correspondence map represents the (luminance) image intensities. That is, bright patches of the target image and bright patches of the reference image are defined to have low error in correspondence. In texture transfer, the image being synthesized has to conform to two independent constraints: (a) the outputs must be legitimate, synthesized examples of the source texture, and (b) corresponding images must be mapped. The authors Efros and Freeman modified the error term in the image-quilting algorithm to be a weighted sum, i.e., α times the block-overlap matching error plus $(1-\alpha)$ times the squared error between corresponding map pixels within the source-texture block and those in

the current target-image position. Parameter α determines the tradeoff between texture synthesis and fidelity to the correspondence map for the target image.

Sometimes one pass through the image is not enough to synthesize a visually pleasing result because of the added constraints. In such cases, we can iterate synthesis over the image several times, reducing the size of the block with each iteration. The only change from the non-iterative version is that in satisfying the local-texture constraint the blocks are matched not just with their neighboring blocks in the overlap regions, but also with whatever was synthesized in this block in the previous iteration. This iterative scheme works surprisingly well; it starts out using large blocks to roughly assign where everything will go and then uses smaller blocks to ensure the different textures will fit together well.

In fact, despite its simplicity, this method works remarkably well when applied to texture synthesis, producing results that are equal to or better than the Efros & Leung family of algorithms but with improved stability and at a fraction of their computational cost.

I have also extended this method of synthesizing and transferring textures to a Chinese font to make it look like manually rendered calligraphy that looks scratched and blurred.

1.2. Motivation

There have recently been many papers on producing non-photorealistic images, and some of the algorithms presented in them can produce magnificent results. However, most of the methods that have been proposed have used some absolute criteria, e.g., certain mathematical formulae as I previously explained, i.e., color-transfer, texture-synthesis, and texture-transfer algorithms, where they can uniquely create a non-photorealistic impression.

In fact, when artists render painterly art, they can create it with their own feelings and sensitivities depending on their own individual styles. These styles can be distinguished by various elements, such as motifs, colors, deformations in shape, and textures. Graphic researchers are interested in these characteristics. Several researchers have developed numerous algorithms to produce NPR images. NPR is an alternative to photorealism and produces painterly images that feature various expressions similar to those used in actual

painterly art. As I have previously discussed, some methods of pasting the textures of painterly art to photographs have been suggested. Other methods of applying the coloration of painterly art to photographic images have also been suggested. I was interested in altering the colors of a photograph using a reference image taken from painterly art. Color transfer is one of the major themes of NPR, and it is one of the major tasks done by CG designers. Examples of color transfer are changing a blue sky into a sunset and changing a dark skin color into a lighter shade. Adobe Photoshop is a popular tool for color transfer and is being utilized by designers; however, it does not work automatically.

What I wanted was to develop a method that could convert color depending on the feelings and sensitivities of designers themselves, like a painter who makes a picture in the real world. My algorithm can produce a color-transformed image resulting from two input images, i.e., a target photograph and a reference painting. My method is based on interactive evolutionary computation (IEC) and can produce a variety of images with transformed colors. The designer interactively selects some candidates from these varieties to obtain the final results, which look similar to reference paintings. This is different from the previous methods I have mentioned in that it can produce many kinds of intermediate and final results. This method is also different from Adobe Photoshop, which needs special design skill to manually achieve color transfer. Users just input their own evaluations, then my system can create more satisfactory candidates automatically until they are satisfied with them.

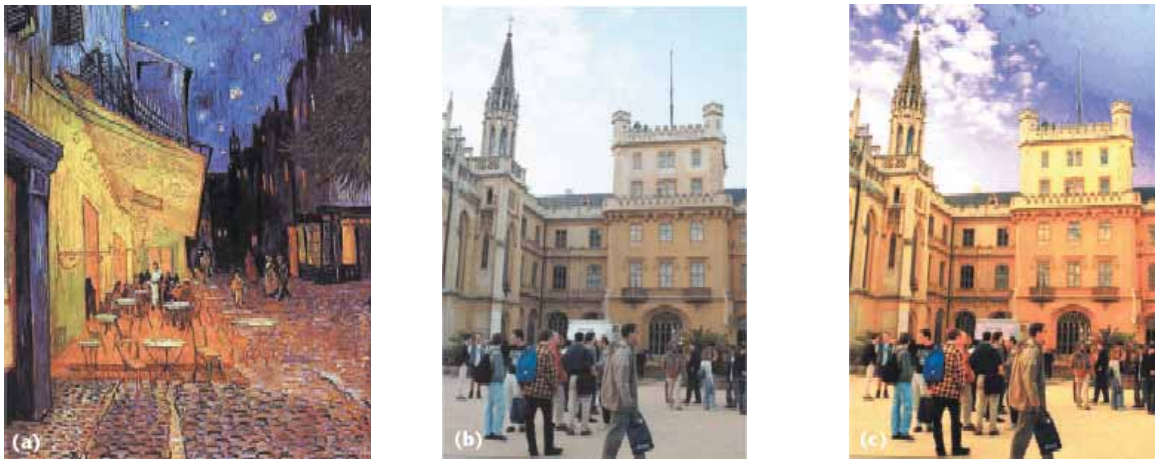
Although the uniqueness of the transformations is indeed practical and efficient in some cases, it does not always satisfy what the user wanted. Assuming that you have a photograph with a red sunset sky for a target and a painting with both a blue sky and a red bird for a reference. If you want to change the sunset of the target into the blue of the reference, what happens? As the distance in color between the sunset and the red bird is nearer than the one between the sunset and the blue sky, you cannot change the sunset into a blue sky. I believe that the process of detecting color pairs not only depends on the distance between colors but also on other factors, such as the artistic subjectivity of users.

I attempted to attain color transfer from a painting to a photograph with IEC [6–8]. My purpose was to supply a useful color transfer system to users. I believe that we can acquire color-transfer images that can create similar impressions to the reference images. Users in

some situations would be surprised by the breathtaking results that can be obtained from color transfer. This kind of software should be useful in many fields such as advertisements and movies.

1.3. Contrast With Other Methods

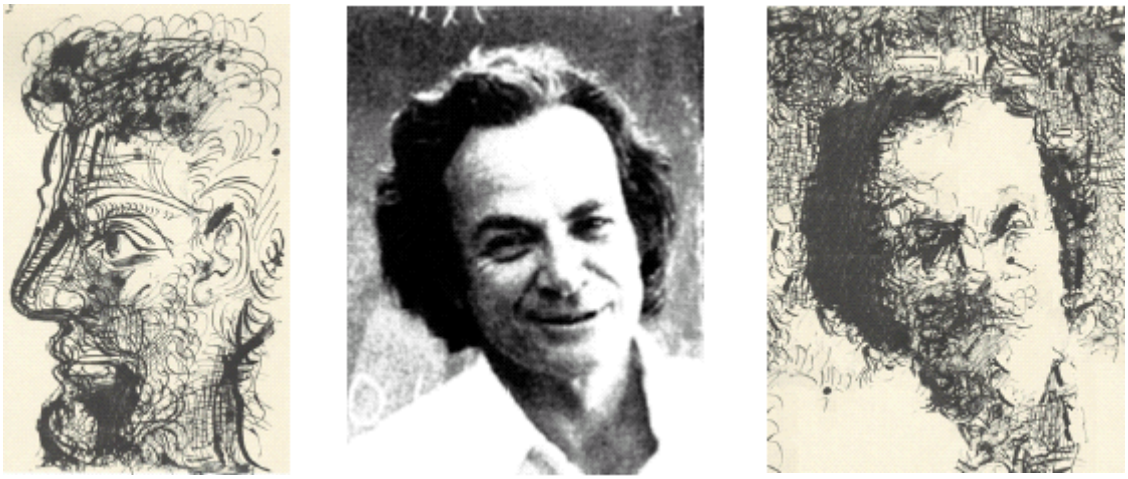
As I previously discussed, there have been many methods of producing NPR. Some researchers have applied coloration of painterly art to photographic images by taking statistics and color corrections into consideration on the basis of $l\alpha\beta$ color space, as can be seen from Fig. 4. For example, Reinhard, Ashikhmin, Gooch, and Shirley (2001) discussed "Color transfer between images" in IEEE Computer Graphics and Applications.



Painterly art　　Photograph　　Non-photorealistic image

Fig. 4 Color transfer

Another method of generating NPR using transfer-texture patches with image quilting is shown in Fig. 5. This is Efros and Freeman's "Image Quilting for Texture Synthesis and Transfer" in ACM Siggraph 2001.



Source texture image Target photograph Non-photorealistic result

Fig. 5 Texture transfer

The methods I previously mentioned are automatic, and can produce unique results. The transformation is indeed unique and effective in many cases. However, algorithms cannot always obtain the results that users need, and it is difficult for them to estimate the generated results. Adobe Photoshop is a popular tool utilized by designers for transferring colors; however, it does not work automatically. Users have to manually undertake color transfer themselves.

I developed a method that could transfer color from painterly art to a photograph with IEC. This method is easy to operate as candidates produced automatically by the system are evaluated by users, who award either 0 or 1 for fitness, i.e., 0 for unacceptable, and 1 for acceptable. Users can also input their own subjective requirements into the results because their evaluations will be reflected in the next generation until the final result is achieved. My method is between photo retouching and automatic color transfer.

1.4. Overview of Thesis

The remainder of the thesis is organized as follows. Chapter 2 introduces methods of color transfer between images with IEC. Color transfer and IEC are important terms in this chapter. The method of color transfer is based on statistical analysis (REINHARD, 2001), and the encoding of chromosomes. This is usually a biological term indicating a string of DNA. In this thesis “chromosome” is a technical term for IEC and a chromosome indicates a

solution to completing the region-matching table and is represented as a bit string. I will then discuss GA and provide some examples. Finally, I will discuss its efficiency.

Chapter 3 proposes another method of color transfer based on hierarchical image-region matching with IEC. It is based on the segmentation of hierarchical images, and IEC can also be done several times. I define T_r as the image-region matching table. T_{r1} is completed after the first IEC. We can use the completed T_{r1} and restrict the matching pair in the next phase table, T_{r2}, to reduce the search space. This is done the same way in the final phase for T_r. The user's history of evaluation is well reflected in the next generation using this method. I will also present some results, and discuss the efficiency of the hierarchical image-region matching algorithm.

Chapter 4 proposes a method of transferring texture from a Chinese font created by a famous artist to another Chinese font printed by computer. I only have some intermediate results as this project is not yet finished. However, I think by improving the algorithm, better results could be obtained. This method is based on the algorithm for synthesizing and transferring textures from the reference to the target image, with the correspondence map. It also satisfies the texture-synthesis requirements, which I previously mentioned, as presented by Efros and Freeman. However, their algorithm just produces general texture synthesis in regular raster scan order when the patches are quilted. I had to face the problem of structural abnormity with my method of producing a Chinese font, and the directivity of strokes used in it. A Chinese font can be produced with transferred texture that looks like one created by a famous artist if these two problems are solved.

CHAPTER 2

COLOR TRANSFER BETWEEN IMAGES WITH INTERACTIVE EVOLUTIONARY COMPUTATION (IEC)

2.1. Introduction

All artists in the real world render paintings in their own style, which can be distinguished by elements, such as motifs, colors, deformations in shape, and textures. Graphic researchers, however, have demonstrated many techniques that can produce NPR images. One aim of NPR, which is an alternative to photorealism, is to produce painterly images that feature expressions similar to those used in actual painterly art (Gooch, 2001). Some methods of applying the texture of an image to a photograph were previously suggested (the texture in an image has some shape and stroke characteristics that can make the photograph look much more like the image) (Hertzmann, 2001; Wang, 2004). Other methods of applying the coloration of painterly art to a photographic image have also been suggested (Chang, 2002; Reinhard, 2001). Altering the colors of an image is also one of the most common tasks in image processing, and this is the main interest of my work.

Fig. 1. Target photograph I_{in}

Fig. 2. Reference painting I_{ref}

Use of the color space should be considered when considering a method of applying the colors of a given painting to a photograph. Previous research used $l\alpha\beta$ and *CIE* $L^*a^*b^*$ color space (Reinhard, 2001; Chang, 2002) (this color space was developed by the International Commission on Illumination, usually known as CIE for its French-language name of Commission Internationale de l'Éclairage. It was determined by physiological measurements of human color vision. *L* represents the luminance of color, and *a* and *b*

represent color) (Chang, 2002; Reinhard, 2001). Both $l\alpha\beta$ and CIE $L^*a^*b^*$ color space are a kind of 3D color-space representation.

Fig. 3. Resulting image 1

Fig. 4. Resulting image 2

The l- axis represents an achromatic channel in $l\alpha\beta$ color space, while α and β channels are chromatic yellow-blue and red-green opponent channels. In *CIE* $L^*a^*b^*$, each axis of *L*, *a*, and *b* represents similar channels as well as $l\alpha\beta$. The first method is based

on simple statistical analysis in color space (REINHARD, 2001), whereas the second is based on color-categorization characteristics of human vision (CHANG, 2002). Although these methods have some differences in their approaches, they basically aim at similar goals.

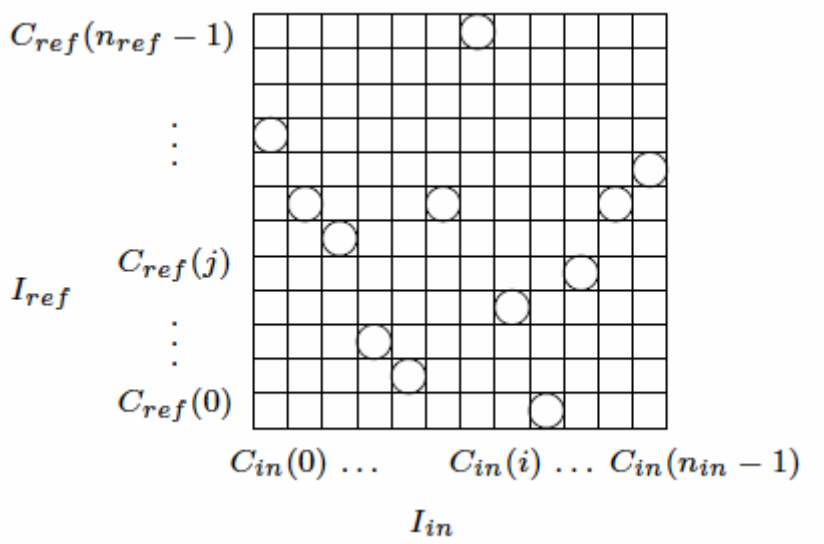


Fig. 5. Color matching table

Algorithms for applying the colors of a given painting evaluate the distance in colors between the painting and target photograph in color space, and they acquire color pairs to uniquely transform the colors. The transformation is indeed unique and effective in many cases. However, algorithms cannot always generate results that satisfy user requirements, and it is difficult for them to estimate these. I believe that the process of detecting color pairs not only depends on the distance between colors but also on other factors, such as the artistic subjectivity of users.

This thesis describes my attempts to transfer the colors of a painting to a photograph, where the subjectivity of users was the only criterion for assigning colors. Color-transfer processing exploits IEC (KATAGAMI, 2002; Takagi, 2001; TOKUI, 2000) and finds color correspondences between a painting and a photograph. IEC is applied in my algorithm to alter the photograph's colors. Users can repeat operations to evaluate the output for re-colored images and assign various fitness values, which are either 1 or 0. IEC actually requires users to undertake part of the color-transfer operations themselves by showing them a variety of candidate images. Thus, IEC is semi-automatic as it does not automatically do all the processing.

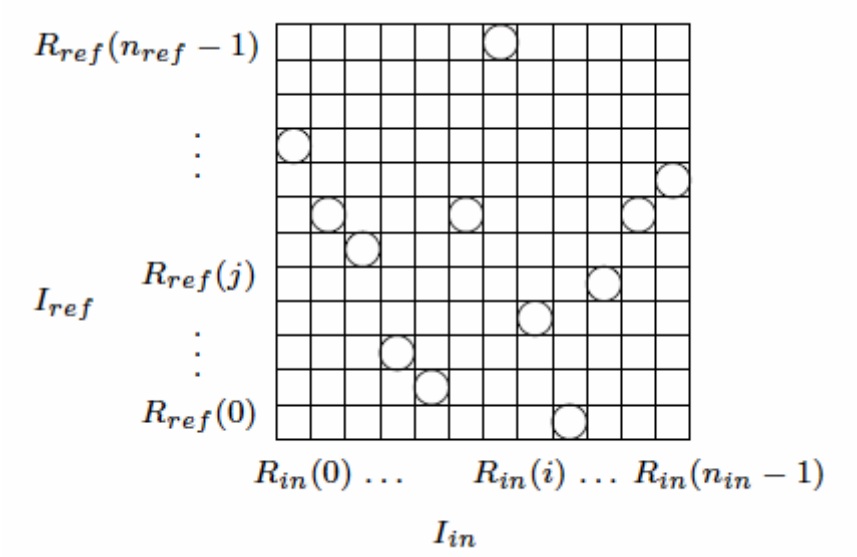


Fig. 6. Region matching table T_r

Figure 3 has an example of a color-transferred result obtained by exploiting IEC. Figure 1 is a target photograph, I_{in}, and Fig. 2 is a reference painting, I_{ref}. IEC transferred the color from Fig. 2 into Fig. 1 to produce the coloration in Fig. 3. Figure 4 is another example of a color-transferred result; however, I did not exploit IEC to produce this image. Instead, I calculated the Euclid distance in CIE $L^*a^*b^*$ color space and applied the nearest neighboring color in the painting (Fig. 2) to the corresponding one in the photograph (Fig. 1) in a similar way to that used previously (REINHARD, 2001; CHANG, 2002), which is given as

$$P_{out}(x, y) = P_{ref}(i, j) \quad \text{..... (1)}$$

This is where $P_{out}\ (x,\ y)$ is a pixel value for the $(x,\ y)$ coordinate on an output color-transferred result, and $P_{ref}\ (i,\ j)$ is a pixel value for the $(i,\ j)$ coordinate on reference painting I_{ref}. The pixel values correspond to points in $L^*a^*b^*$ color space. $P_{ref}\ (i,\ j)$ is calculated as

$$P_{ref}(i, j) = \arg\min_{P_{ref}(i,j)}(dist(P_{in}(x, y), P_{ref}(i, j))) \quad \text{,..... (2)}$$

where $P_{in}(x,\ y)$ is a pixel value for (x, y) on target photograph I_{in}, and dis($P_{in}(x,\ y)$, $P(i,\ j)$) represents the Euclid distance in *CIE* $L^*a^*b^*$ color space as follows. The $L_{in}(x, y)$, $a_{in}(x, y)$,

and $b_{in}(x, y)$ below are *L*, *a*, and *b* values for $P_{in}(x, y)$. Then, $L_{ref}(i, j)$, $a_{ref}(i, j)$, and $b_{ref}(i, j)$ are *L*, *a*, and *b* values for $P_{ref}(x, y)$.

$$dis\left(P_{in}(x, y), P_{ref}(i, j)\right) = d_L + d_a + d_b \quad \text{..... (3)}$$

$$d_L = \left(L_{in}(x, y) - L_{ref}(i, j)\right)^2,$$

$$d_a = \left(a_{in}(x, y) - a_{ref}(i, j)\right)^2, \quad \text{..... (4)}$$

$$d_b = \left(b_{in}(x, y) - b_{ref}(i, j)\right)^2.$$

We can see that there are visual differences between Figs. 3 and 4. The coloration between Figs. 3 and 2 is much more similar than that between Figs.4 and 2.

I will present problems and solutions in Chapters 2 and 3, where I discuss how IEC was exploited for color transfer. Chapter 4 has an overview of the algorithm. Chapter 5 presents some output examples produced by the algorithm and an evaluation of the usefulness of the IEC approach. Finally, I will discuss some future directions for color transfer. Figures 1–4 and 13–22 were originally created from color images. They can be seen in a Web version of Forma. URL is http://www.scipress.org/journals/forma/frame/19.html.

2.2. Color Transfer

Color transfer is a kind of combinatorial optimization problem. Previous work on this subject has dealt with color transfer as a color-matching problem between a photograph and a painting. There is an example of the color-matching problem in Fig. 5. $C_{in}(i)$ and $C_{ref}(j)$ represent the i-th color of target photograph I_{in} and the *j-th* color of reference painting I_{ref}. The circles in the table indicate $C_{ref}(j)$ has been assigned to its corresponding $C_{in}(i)$. The color-matching table has been completed using several criteria, e.g., the nearest distance in color space that was exploited to produce Fig. 4, as I previously mentioned.

I attempted to use a new type of table in this thesis, called a “region-matching table”, and denoted it by T_r. To begin with, I divided each image into several regions, which indicated areas in the image, i.e., the set of all regions is equivalent to a whole image. The regions in target photograph I_{in} are represented as $R_{in}(i)$ $(i = 0, \cdots, N_{in}-1)$ and the ones in reference painting I_{ref} are represented as $R_{ref}(j)$ $(j = 0, \cdots, N_{ref}-1)$. We can then determine pair $(R_{in}(i), R_{ref}(j))$, and assign a mean color, which is calculated by Eq. (5), to the corresponding $R_{in}(i)$. Figure 6 is an example of a region-matching table. The circles in the table indicate a color in $R_{ref}(j)$ has been assigned to $R_{in}(i)$.

$$\left\langle L_{ref}(j) \right\rangle = \frac{\sum_{x,y \in R_{ref}(j)} L_{ref}(x,y)}{n_{ref}(j)},$$

$$\left\langle a_{ref}(j) \right\rangle = \frac{\sum_{x,y \in R_{ref}(j)} a_{ref}(x,y)}{n_{ref}(j)}, \quad \text{..... (5)}$$

$$\left\langle b_{ref}(j) \right\rangle = \frac{\sum_{x,y \in R_{ref}(j)} b_{ref}(x,y)}{n_{ref}(j)},$$

where $n_{ref}(j)$ is the total number of pixels in $R_{ref}(j)$.

Region-matching table T_r provides more solutions to color-transfer problems than the color-matching table. That is, even if the regions in I_{in} have the same color, T_r can assign different colors to each region. Instead, it needs a larger search space, and this will of course take more time to acquire an optimum solution. There is one further problem that cannot be ignored. This depends on the IEC’s features. IEC produces color-transformed images semi-automatically. The users themselves provide a fitness function for IEC, and they have to repeat operations to evaluate color-transformed candidates until they obtain an optimum result that satisfies them. These repeated operations force them to do a lot of work if the search space is huge. Thus, I had to introduce a method of reducing the number of

evaluations. My approach was based on the intuitive idea that painters roughly assign colors on a canvas first and when they draw and begin to render a painting, they decide the colors for the finer regions.

On the basis of this idea, I first prepared coarse regions allowing me to roughly assign colors to regions. I adopted *k-means* to acquire the coarse regions, which is a method for segmenting images (TAKAGI, 1991). After that, I assigned fine colors to all regions based on statistical analysis (REINHARD, 2001).

Once IEC had finished assigning rough colors, matching pair $(R_{in}(i);R_{ref}(j))$ was determined. Fine colors were assigned between $R_{in}(i)$ and $R_{ref}(j)$. To achieve this, I wanted some aspects of the distribution for data points in $L^*a^*b^*$ space to transfer between $R_{in}(i)$ and $R_{ref}(j)$, where the means and standard deviations along each of the three axes were sufficient. Thus, I computed these measures for both $R_{in}(i)$ and $R_{ref}(j)$.

The precise process is as follows. In Eq. (6), each of $L_{in}(x,y)$, $a_{in}(x,y)$, and $b_{in}(x,y)$ represents an $L^*a^*b^*$ color member of (x, y) in $R_{in}(i)$. Then, each $\langle L_{in}(i)\rangle$, $\langle a_{in}(i)\rangle$, and $\langle b_{in}(i)\rangle$ represents a mean of the $L^*a^*b^*$ color member in $R_{in}(i)$. I computed the means and standard deviations for each axis separately in the $L^*a^*b^*$ space. I first subtracted the mean from the data at each point, (x, y):

$$L'(x,y) = L_{in}(x,y) - \langle L_{in}(i)\rangle,$$

$$a'(x,y) = a_{in}(x,y) - \langle a_{in}(i)\rangle, \quad \text{..... (6)}$$

$$b'(x,y) = b_{in}(x,y) - \langle b_{in}(i)\rangle,$$

$$x, y \in R_{in}(i).$$

I then scaled the data points comprising the synthetic region by factors determined by respective deviations:

$$L_{out}(x,y)=\frac{\sigma_{ref}^{L}(j)}{\sigma_{in}^{L}(i)}L'(x,y)+\left\langle L_{ref}(j)\right\rangle,$$

$$a_{out}(x,y)=\frac{\sigma_{ref}^{a}(j)}{\sigma_{in}^{a}(i)}a'(x,y)+\left\langle a_{ref}(j)\right\rangle, \quad \text{..... (7)}$$

$$b_{out}(x,y)=\frac{\sigma_{ref}^{b}(j)}{\sigma_{in}^{b}(i)}b'(x,y)+\left\langle b_{ref}(j)\right\rangle.$$

where $L_{out}(x,y)$, $a_{out}(x,y)$, and $b_{out}(x,y)$ represent the pixel values of *(x, y)* on color-transferred image I_{out}. $\sigma_{ref}^{*}(j)$ and $\sigma_{in}^{*}(i)$ $(*\in\{L,a,b\})$ are standard deviations in $R_{ref}(j)$ and $R_{in}(i)$ for each axis. After these transformation, the resulting data have standard deviations that conform to $R_{ref}(j)$.

2.3. IEC

2.3.1. Encoding a chromosome

IEC manages and completes region-matching table T_r as we can see from Fig. 6, where N_{ref} indicates the total number of I_{ref} regions and N_{in} indicates that for I_{in}. When the problem of assigning $R_{ref}(j)(j=0,\cdots,N_{ref}-1)$ to $R_{in}(i)(i=0,\cdots,N_{in}-1)$ in table T_r is considered, there are $(N_{ref})^{N_{in}}$ total combinations. My approach uses the table itself as a chromosome for IEC. The expression of a chromosome (even though “chromosome” is usually a biological term indicating a string of DNA, it is a technical term for IEC in this thesis, indicating a solution to completing T_r and is represented as a bit string.) is as follows:

$$\left\lfloor x_0, x_1, \cdots, x_{N_{in}-1} \right\rfloor,$$
$$0 \le x_n \le N_{ref}-1 (n=0,\cdots,N_{in}-1). \quad \text{..... (8)}$$

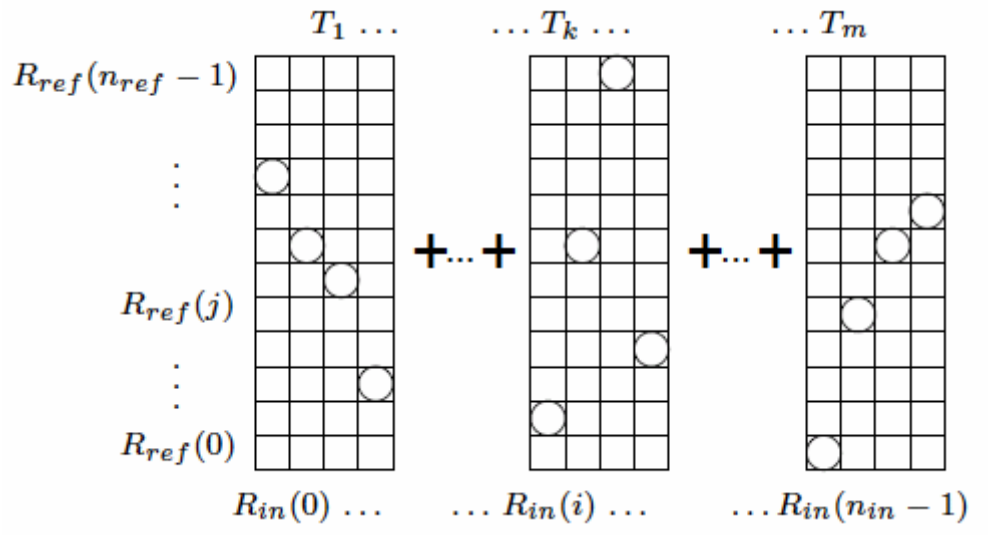


Fig. 7. Divided region-matching table

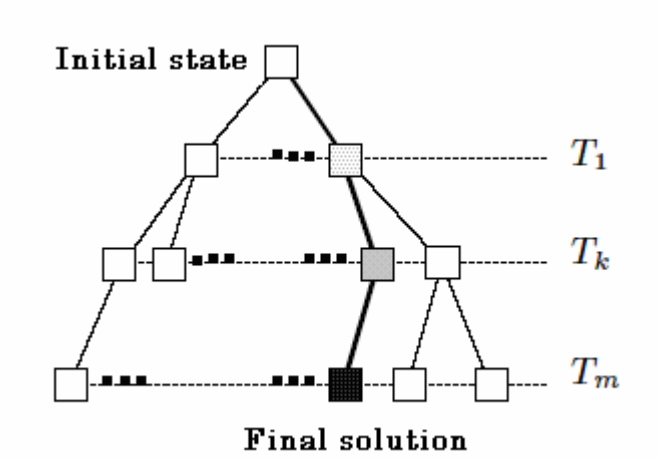


Fig. 8. Search tree to obtain final solution

The length of a chromosome will be $N_{in} \log N_{ref}$ bits, which is too long when the table is large. To solve this problem, my approach divides the whole table T_r, to *K*-small tables as shown in Fig. 7, if N_{in}, the total number of I_{in} regions, is beyond a threshold value. T_r^k indicates a *k*-th small table (k = 0,⋯,k − 1). Thus, each T_r^k has ($N_{in} \log N_{ref}$)/K bits as the length of a chromosome. Then, a user evaluates and completes T_r^0, and repeats the same operations for $T_r^1, \cdots, T_r^{K-1}$. This means that the user decomposes a whole path to a final solution into K-subgoals. This is a kind of depth-first search as shown in the search tree in Fig. 8, and users are only allowed to evaluate and complete T_r^k on the *k*-th layer of the tree.

Furthermore, as part of the preprocessing before a search is begun, I sorted the order of regions in area order. The order of regions is given by

$$sum_p\left(R_{in}(i)\right) \geq sum_p\left(R_{in}(i+1)\right), \quad \text{..... (9)}$$

where $sum_p\left(R_{in}(i)\right)$ is the total number of pixels in $R_{in}(i)$. Equation (9) determines the order of the horizontal axis throughout the entire region-matching table, T_r. Thus, a user begins to evaluate and complete a small table, T_r^0, which occupies the largest areas in I_{in}, then treats smaller tables, $T_r^1, \cdots, T_r^{K-1}$. This sorting is also based on the intuitive idea that painters tend to begin assigning colors to large areas first. I believe that the size of the area is an important factor in assigning colors and that larger regions have a higher visual impact. Therefore, my approach mimics a painter's process of drawing in the real world.

$$Chromosome(1): \left\lfloor x_0(1), x_1(1), \cdots, x_{N_{in}-1}(1) \right\rfloor$$
$$Chromosome(2): \left\lfloor x_0(2), x_1(2), \cdots, x_{N_{in}-1}(2) \right\rfloor$$
$$\vdots$$
$$Chromosome(M): \left\lfloor x_0(M), x_1(M), \cdots, x_{N_{in}-1}(M) \right\rfloor$$

Fig. 9. Population

$$Chromosome(a): \left\lfloor x_0(a), x_1(a), \cdots, x_n(a), x_{n+1}(a), \cdots, x_{N_{in}-1}(a) \right\rfloor$$
$$Chromosome(b): \left\lfloor x_0(b), x_1(b), \cdots, x_n(b), x_{n+1}(b), \cdots, x_{N_{in}-1}(b) \right\rfloor$$
$$\Downarrow$$
$$Offspring(a): \left\lfloor x_0(b), x_1(b), \cdots, x_n(b), x_{n+1}(a), \cdots, x_{N_{in}-1}(a) \right\rfloor$$
$$Offspring(b): \left\lfloor x_0(a), x_1(a), \cdots, x_n(a), x_{n+1}(b), \cdots, x_{N_{in}-1}(b) \right\rfloor$$

Fig. 10. Crossover

2.3.2. GA operations

The operations for IEC I established are illustrated by the algorithm below which runs for each T_r^k ($k = 0, \cdots, K-1$). However, we can use $K = 1$ to simplify explanation.

1. [Start] Initialize the initial adaptive probability by $P_0(i, j)$, which is calculated by Eq. (10). $P_0(i, j)$ indicates the probability that $R_{ref}(j)$ will be assigned to $R_{in}(i)$ in the initial generation. Generate a population ("population" indicates a set of chromosomes and is an IEC technical term.) of M chromosomes, illustrated in Fig. 9, under $P_0(i, j)$.

2. [Fitness] A user evaluates and assigns a fitness value of 0 or 1 to each chromosome in the population.

3. [New population] Generate a new population by executing the following steps.

   (i) [Selection] Select a parent chromosomes whose fitness value is 1 from a population, if there are more than two chromosomes whose fitness value is 1.

   (ii) [Crossover] Cross over with the crossover probability the parents will form new offspring ("Offspring" indicates a chromosome in the next generation, which is also an IEC technical term). The crossover probability decides whether crossover is executed or not, and in my system this was set to 0.5. Figure 10 illustrates the operation for crossover. When crossover is executed, a crossover point is determined at the locus between *n* and *n + 1*, which is randomly selected. Offspring (*a*) and (*b*) are generated from parent chromosomes (*a*) and (*b*).

   (iii) [Mutation (1)] Randomly generate new offspring under adaptive probability $P_g(i, j)$, which indicates probability in the *g*-th generation and is defined by Eqs. (10) and (11), except for offspring that the crossover formed.

$$Offspring(a): \left[x_0(a), x_1(a), \cdots, x_n(a), \cdots, x_{N_{in}-1}(a)\right]$$

$$\Downarrow$$

$$Offspring(a): \left[x_0(a), x_1(a), \cdots, y_n(a), \cdots, x_{N_{in}-1}(a)\right]$$

Fig. 11. Mutation (2)

   (iv) [Mutation (2)] Randomly mutate all the new offspring at each locus with a mutation probability of 0.025. Figure 11 illustrates *[Mutation (2)]* operation. When locus *n* is selected with the mutation probability, $x_n(a)$ changes random value $y_n(a)$.

   (v) [Accepting] Place the new offspring in a new population.

4. [Replace] Use a newly generated population for a further run of the algorithm.

5. [Update] Update adaptive probability $P_g(i, j)$ for the next generation according to Eq. 11.

6. [Test] Stop and return the best solution if a user finds a chromosome that satisfies his or her subjective criteria in the current population.

7. [Loop] Go to Step 2.

The definition of $P_0(i, j)$ is given below. It is based on the same idea that a larger area could create a higher visual impact.

$$P_0(i, j) = \frac{sum_p\left(R_{ref}(j)\right)}{\sum_{k=0}^{N_{ref}-1} sum_p\left(R_{ref}(k)\right)} \quad \text{..... (10)}$$

Adaptive probability $P_g(i, j)$ in the g-th generation is assigned to $R_{ref}(j)$ for $R_{in}(i)$. $P_{g+1}(i, j)$, which indicates the (g+1)-th generation, is given by

$$P_{g+1}(i, j) = \frac{q(i, j)}{\sum_{k=0}^{N_{ref}-1} q(i, k)}, \quad \text{..... (11)}$$

where ω is a coefficient in these equations to update adaptive probability, and determined as:

$$\omega = \begin{cases} \omega_1 : \text{for fitness} 1 \\ \omega_0 : \text{for fitness} 0 \\ 1.0 : \text{otherwise} \end{cases}$$

The relation between ω_1 and ω_0 is given by $\omega_1 > 1.0 > \omega_0 \geq 0.0$. For example, $\omega_1 = 2.0$ and $\omega_0 = 0.5$ in my experiments. Thus, note that the adaptive probabilities of regions included in a chromosome, where a user awarded 1 for fitness, increase in the next generation. The higher the adaptive probability, the larger the chance of being selected in [Mutation (1)]. The adaptive probabilities of regions included in a chromosome, on the other hand, where a user awarded 0 for fitness, decrease in the next generation. The history of user evaluations is well reflected in the evolution process.

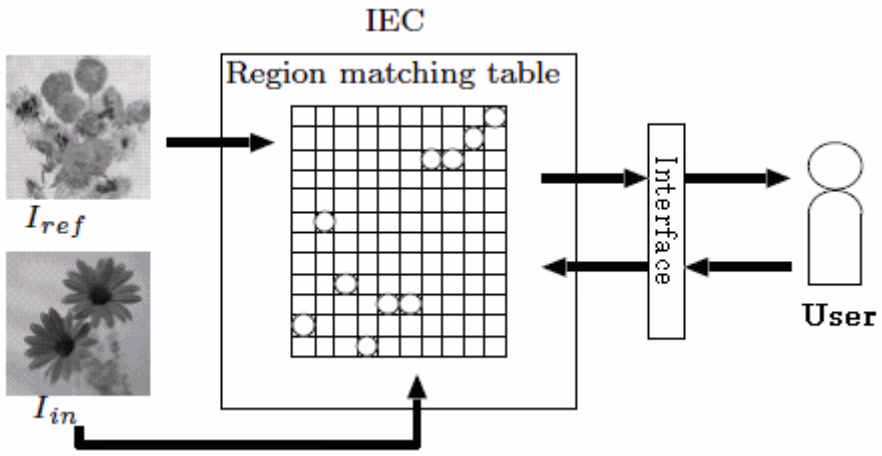


Fig. 12. System overview

2.4. System Overview

Figure 12 overviews the system I developed. A user prepares two images, i.e., target photograph I_{in} and reference painting I_{ref}. The system divides each image into coarse regions by exploiting image segmentation. The regions acquired from I_{in} and I_{ref} constitute the *xy*-axes of the region matching table T_r (see Fig. 6). The IEC interface is located between the users and T_r. They evaluate and complete T_r through the IEC interface. Of course, they do not need to handle T_r directly as the IEC interface of the system produces it, and users just have to award a fitness of 0 or 1 to them. The system produces new candidates based on fitness, and users evaluate them again. Such processing is repeated until a candidate satisfies their subjective criteria.

I will present some results produced by IEC in the next section. I will then discuss the efficiency of the algorithm. I will present problems with the algorithm in the last section, and what needs to be done in future work.

2.5. Experimental Results

I applied color-transfer processing to 10 pairs of images in my experiments with each pair made up of a target photograph and a reference painting. Figures 13–22 are examples generated by my color-transfer algorithm. The figures labeled *a* are the input photographs, and those labeled *b* are the reference painterly art. The figures labeled *c* are examples of color-transformed images resulting from the target photographs labeled *a*. The similarities between *b* and *c* can clearly be seen in each.

I asked users to assign the colors of the reference paintings to the target photographs using the IEC system under the condition that the result had to have a similar look and feel to the reference paintings. Color-transfer processing finished when an output result satisfied their own subjective criteria. Five trials for each pair of images were done, and five color-transformed resulting were acquired for each pair. The five results might have had slight visual differences although they were made using the same pair. This is because my system depended on the users' subjective criteria. This is one of my system's features that exploits IEC. Figures 24 and 25 plot the mean processing time and the mean for the total number of generations until the user was satisfied with each final result. My system operated on a PC with a Celeron 2.40-GHz CPU with 512 MB of memory, running Sun Microsystem's Solaris 9 operating system.

We can see from the targets and references for the experiments in the *a* and *b* figures that the compositions are comparatively simple. Thus, it was not difficult for users to consider the combinations they needed to assign color to the target photograph. The generation in Fig. 25 might represent the time a user spent in considering his or her choices. Figure 25 indicates that 8–32 generations are needed until an output result satisfies a user's own subjective criteria. The average is 18 to 19 generations. This is not as high as for the conventional IEC system. However, Fig. 24 indicates that is takes users a considerable amount of time to acquire the final result.

(1)
Fig. 13(a). Target photograph

(1)
Fig. 13(b). Reference painting

(1)
Fig. 13(c). Resulting image

(2)

Fig. 14(a). Target photograph

(2)

Fig. 14(b). Reference painting

(2)

Fig. 14(c). Resulting image

(3)

Fig. 15(a). Target photograph

(3)

Fig. 15(b). Reference painting

(3)

Fig. 15(c). Resulting image

(4)

Fig. 16(a). Target photograph

(4)

Fig. 16(b). Reference painting

(4)

Fig. 16(c). Resulting image

(5)

Fig. 17(a). Target photograph

(5)

Fig. 17(b). Reference painting

(5)

Fig. 17(c). Resulting image

(6)

Fig. 18(a). Target photograph

(6)

Fig. 18(b). Reference painting

(6)

Fig. 18(c). Resulting image

(7)

Fig. 19(a). Target photograph

(7)

Fig. 19(b). Reference painting

(7)

Figure 19(c). Resulting image

(8)

Fig. 20(a). Target photograph

(8)

Fig. 20(b). Reference painting

(8)

Fig. 20(c). Resulting image

(9)

Fig. 21(a). Target photograph

(9)

Fig. 21(b). Reference painting

(9)

Fig. 21(c). Resulting image

(10)

Fig. 22(a). Target photograph

(10)

Fig. 22(b). Reference painting

(10)

Fig. 22(c). Resulting image

2.6. Conclusion

In this thesis, I described my attempts to represent color transfer from a painting to a photograph to produce a resulting image with IEC. I acquired color-transferred images in the (c) figures that created a similar impression to the reference images in (b). The system using IEC only required users to push buttons in the system window on a display monitor. Such easy operations enabled them to award either 0 or 1 for fitness and evolve color-transferred images interactively. Although photo retouching tools, e.g. Adobe Photoshop, are also available to obtain colors interactively, users must acquire high-level skills and it is difficult for them to reflect their own subjective criteria in automatic color transfer. My method is positioned between photo retouching tools and automatic color transfer.

Some output examples from the experiments were obtained, and I evaluated the processing time as plotted in Fig. 23. Although the processing time largely depends on the PC specifications, the present system took too long for processing. The processing time is one of the most important factors in an IEC system because the faster the processing, the less the user has to do. This problem needs to be resolved immediately. I prepared comparatively simple images for the experiments, and I did not need to divide them into many regions. If the processing time can be shortened, more complex images can be handled. These problems need to be solved in future work.

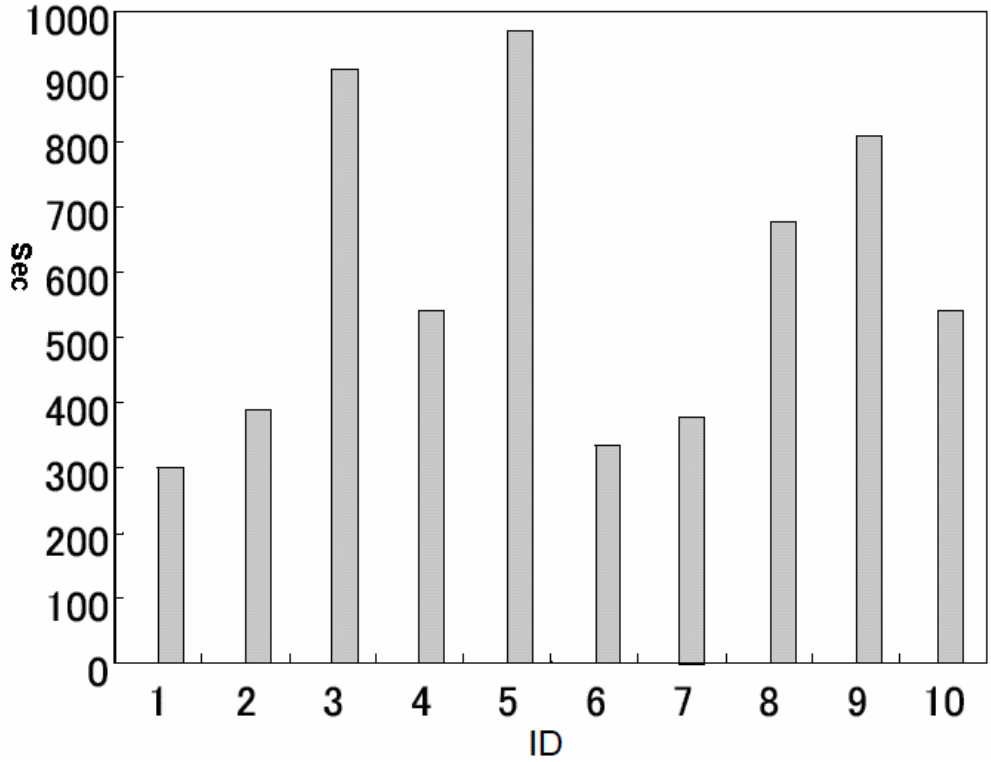


Fig. 23. Processing time to obtain satisfactory results

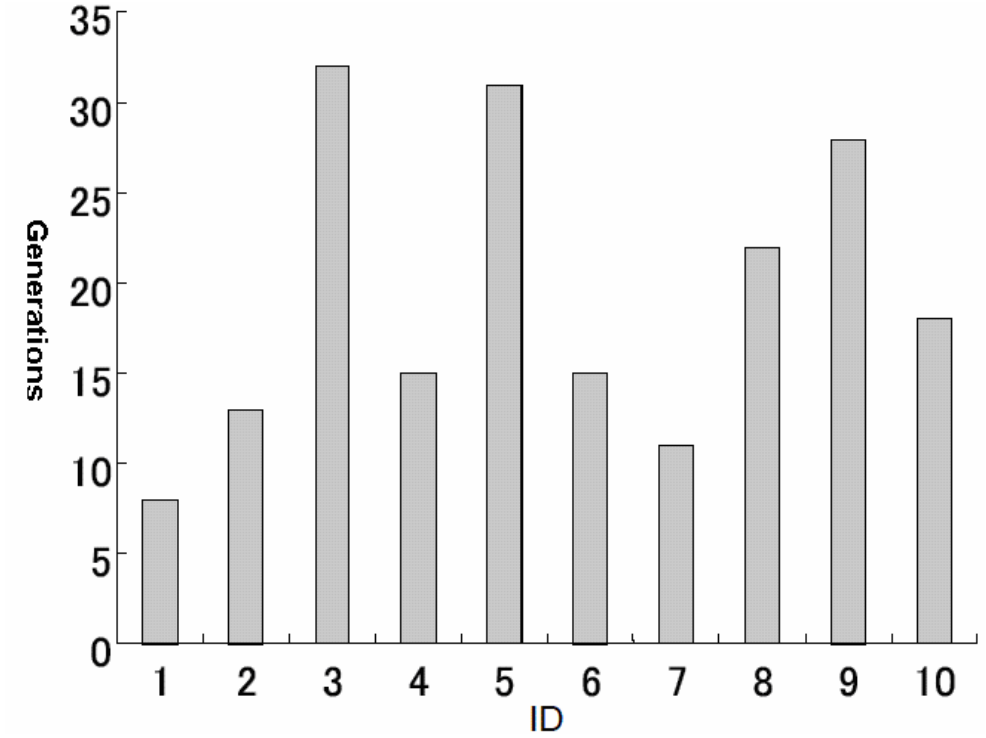


Fig. 24. Generations of IEC to obtain final satisfactory results

2.7. References

CHANG,Y., SAITO, S. and NAKAJIMA, M., Color transformation based on the basic color categories of a painting, *SIGGRAPH 2002 Conference Abstracts and Applications*, pp. 157–157, 2002.

A. Efros and W. Freeman. Image Quilting for Texture Synthesis and Transfer. In Proceedings of ACM Siggraph 01, pp. 341–346, 2001.

GOOCH, B. and GOOCH, A., *Non-Photorealistic Rendering*, A. K. Peters. Ltd., Natick, MA, USA, 2001.

HERTZMANN, A., JACOBS, C.E., OLIVER, N., CURLESS, B. and SALESIN, D.H., Image Analogies, *SIGGRAPH 2001 Conference Proceedings*, pp. 32–340, 2001.

KATAGAMI, D. and YAMADA, S., Interactive evolutionary computation for real robot from a viewpoint of observation, *7th International Conference on Intelligent Autonomous*, pp. 158–165, 2002.

REINHARD, E., ASHIKRHMIN, M., GOOCH, B., and SHIRLEY, P., Color Transfer between images, IEEE Computer Graphics and Applications, 21 (5), pp. 34–41, 2001

TAKAGI, H., Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation, *Proceedings of the IEEE, 89 (9)*, pp. 1275–1296, 2001.

TAKAGI, M. and SHIMODA, H., *Handbook of Image Analysis*, University of Tokyo Press, 1991.

TOKUI, N. and IBA, H., Music composition with interactive evolutionary computation, *3rd International Conference on Generative Art*, 2000.

WANG, B., WANG, W., YANG, H., and SUN, J., Efficient example-based painting and synthesis of 2D directional texture, *IEEE Transactions on Visualization and Computer Graphics*, 10 (3), pp. 266–277, 2004.

CHAPTER 3
COLOR TRANSFER BASED ON HIERARCHICAL IMAGE-REGION MATCHING WITH IEC

3.1. Background

I discussed a method of transferring color from reference paintings to target photographs with IEC in the previous chapter, superior to those presented by other researchers who have also transferred color between images. In the previous chapter, I introduced the region matching table, T_r. In this chapter, I will discuss the transfer of color between images using IEC, based on image-region matching. I developed a new concept of a "hierarchical image-region matching table, T_r" to reduce the IEC processing time when users need to derive more complex images. I think that the hierarchical image-region matching table is better than the region-matching table, although there is still room for improvement. I will describe my algorithm for color transfer using the hierarchical image-region matching table in this chapter, and some experiments conducted with this method. I will then discuss the efficiency of the algorithm.

3.2. Introduction

When artists render painterly art, they can express their feelings and sensitivities based on their individual styles. These can be distinguished by various elements, such as motifs, colors, deformations in shape, and texture. Graphic researchers are interested in

these characteristics and have developed numerous algorithms to produce NPR images. NPR is an alternative to photorealism and produces painterly images that feature various expressions similar to those used in the actual painterly arts [1]. Some methods of pasting the textures of painterly art to photograph had previously been suggested [2–3]. Other methods of applying the coloration of painterly art to photographic images have also been suggested [4–5]. I was interested in altering the colors of a photograph using painterly art as a reference image. Color transfer is one of the major themes of NPR, and it is one of the major tasks faced by CG designers. Examples of color transfer are changing a blue sky into a sunset and changing a dark skin color into a lighter shade. Although Adobe Photoshop is a popular tool for color transfer and is utilized by designers, it does not work automatically. They have to transfer the colors manually.

Before explaining my work of transforming color from painterly art to a photograph, the color space needs to be determined. In my thesis, I used *CIE L*a*b** color space, which is a kind of 3D color-space representation, where the *L**-axis represents an achromatic channel, while *a** and *b** channels are chromatic channels. One of the most common methods of color transformation in *CIE L*a*b** color space uses the nearest distance between the painting and photograph, and can uniquely and automatically acquire color pairs. The transformations are indeed unique and efficient in some cases, but they do not always satisfy what users require. Assuming that you prepare a photograph with a red sunset for the target and a painting with both a blue sky and a red bird for reference. Even if you want to change the sunset of the target into the blue of the reference, what happens? As the color distance between the sunset and the red bird is nearer than the one between the sunset and the blue sky, the sunset cannot be changed into a blue sky. I believe that the process of detecting color pairs not only depends on the distance between colors but also on other factors, such as the user's artistic subjectivity.

In this chapter, I describe my attempts to represent the transfer of color from a reference painting to a target photograph with IEC to produce a resulting image [6–8]. My purpose was to provide users with a useful system for transferring colors. I acquired images with transferred colors that created similar impressions to the reference images. Users would be surprised by the astonishing transfer of color in some areas of the

resulting image. This kind of software should be extremely useful in many fields such as advertising and movies.

Fig. 1: Target photograph I_{in}

Fig. 2: Reference painting I_{ref}

Fig. 3: Result

Fig. 4: Another result

I used IEC in my system, which only requires users to push buttons on the windows of their display monitors. They only need to determine 0 or 1 for fitness and they can interactively evolve the color transfer in their images. The fitness of 0 or 1 depends on their individual subjectivity and feelings. The “0” means unacceptable and the “1” means acceptable. Although photo retouching tools such as Adobe Photoshop can also be used to obtain colors interactively, users need to acquire a high degree of skill, and automatic color transfer makes it difficult to reflect their subjective criteria. My method is positioned between photo retouching tools and automatic color transfer.

I applied IEC in my algorithm to detect color pairs between a reference painting and a target photograph to alter the color in the photograph. The users themselves provided the fitness function for IEC. They could repeat the operations to evaluate the

output re-colored images and assign the fitness values, which are 1 or 0. IEC actually requires users to do some of the color-transfer operations themselves by showing them a variety of candidate images. Thus, my algorithm does not do all the processing automatically; it needs the users' evaluations, and leaves the color-transfer work to the computer. In this way, the color-transfer images result from the users' individual subjective feelings and sensitivities. Of course, users may not always obtain the same results for a color-transferred image from the same reference painting and target photograph.

Figure 1 shows a target photograph, I_{in}, and Fig. 2 shows a reference painting, I_{ref}. Figure 3 has an example of the result of color transfer by exploiting the IEC method. Figure 4 has another example of a color-transfer result; however, I did not exploit IEC to produce this image. Instead of IEC, I calculated the Euclid distance in *CIE L*a*b** color space and applied the nearest neighboring color in the painting (Fig. 2) to the corresponding one in the photograph (Fig. 1) in a similar way to that previously used [4–5], as follows.

$$p_{out}(x, y) = p_{ref}(i, j), \quad \text{..... (1)}$$

where $p_{out}(x, y)$ is a pixel value for the *(x, y)* -coordinate in an output color-transfer result, and $P_{ref}(i, j)$ is a pixel value for the *(i, j)* -coordinate on reference painting I_{ref}. The pixel value corresponds to a point in *L*a*b** color space. $p_{ref}(i, j)$ is calculated as:

$$p_{ref}(i, j) = argmin\left(dist\left(p_{in}(x, y), p_{ref}(i, j)\right)\right), \quad \text{(2)}$$

where $p_{in}(x, y)$ is a pixel value for *(x, y)* on target photograph I_{in}, and $dist(p_{in}(x, y), p_{ref}(i, j))$ represents the Euclid distance in *CIE L*a*b** color space as given below. The following $L_{in}(x, y)$, $a_{in}(x, y)$, and $b_{in}(x, y)$ are the *L*, *a*, and

b values for $p_{in}(x, y)$. Then, $L_{ref}(i, j)$, $a_{ref}(i, j)$, and $b_{ref}(i, j)$ are the *L*, *a*, and *b* values for $p_{ref}(i, j)$.

$$dist\left(p_{in}(x, y), p_{ref}(i, j)\right) = d_L + d_a + d_b, \quad (3)$$

$$d_L = \left(L_{in}(x, y) - L_{ref}(i, j)\right)^2,$$

$$d_a = \left(a_{in}(x, y) - a_{ref}(i, j)\right)^2, \quad \text{.....} (4)$$

$$d_b = \left(b_{in}(x, y) - b_{ref}(i, j)\right)^2.$$

We can see the visual differences between Figs. 3 and 4. The coloration between Figs. 2 and 3 is much more similar than that between Figs. 4. and 2.

I will talk about image segmentation in the following sections, which is basic to preparing the image-region matching table [9]. I will then present the problem with searching the result and propose a solution, which is a hierarchical region-matching table. After that, I will talk about the process of IEC and present an overview of the algorithm. Finally, I will present some experimental outputs produced by the algorithm and discuss its efficiency. I will discuss some future directions for color transfer in the final section.

3.3. Image Segmentation

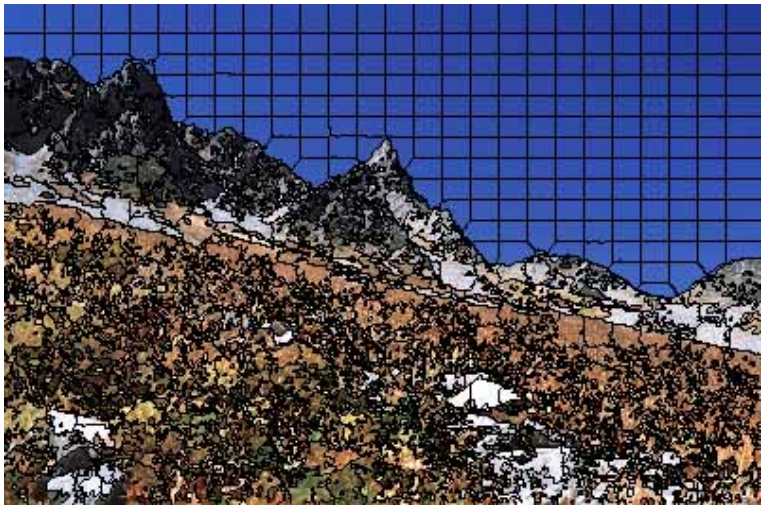


Fig. 5: Segmented image by K-mean

When artists in the real world render painterly art, they apply some color to the coarse regions. For example, the sky in a painterly landscape will always be blue or other predictable colors and the sea will always be blue or green. The colors of faces, within races, are also roughly similar. If we effectively use region-matching between painterly art and the target photograph, we can obtain a resulting image that has the coloration of the reference painterly art. For example, Fig. 3 has the coloration of the reference painterly art in Fig. 2 because the sky, mountains, and foreground are a good match. Based on this, we should first segment the image into regions with objective shapes.

I first used the K-mean to segment the image into several small regions, as seen in Fig. 5. This is calculated as follows:

1. Prepare some seeds in the image.

2. Calculate the distance between all pixels from the seeds.

3. Pixels belong to seeds where this distance is minimal.

4. Calculate the centers of all small regions, taking them as the new seeds.

5. Go to Step 3 and calculate the distance again if this is less than a constant and the loop is over. The constant is one of the parameters in my system.

The distance is calculated as:

$$Dist\left(p\left(x,y\right),p_{seed}\left(i,j\right)\right)=A_1d_X+A_2d_Y+A_3d_L+A_4d_a+A_5d_b$$

$$d_X=\left(X\left(x,y\right)-X_{seed}\left(i,j\right)\right)^2,$$

$$d_Y=\left(Y\left(x,y\right)-Y_{seed}\left(i,j\right)\right)^2,$$

$$d_L=\left(L\left(x,y\right)-L_{seed}\left(i,j\right)\right)^2, \quad \text{..... (5)}$$

$$d_a=\left(a\left(x,y\right)-a_{seed}\left(i,j\right)\right)^2,$$

$$d_b=\left(b\left(x,y\right)-b_{seed}\left(i,j\right)\right)^2.$$

Fig. 6: First segmentation of image

This is where *p(x, y)* is the pixel value for *(x, y)* in the image and $p_{seed}(i, j)$ is the seed value for *(i, j)* in the image. *X(x, y)* and *Y(x, y)* are the axial values for *p(x, y)*. *L(x, y)*, *a(x, y)*, and *b(x, y)* are the *L*, *a*, and *b* values for *p(x, y)*. $X_{seed}(i, j)$ and $Y_{seed}(i, j)$ are the axial values for $P_{seed}(i, j)$. $L_{seed}(i, j)$, $a_{seed}(i, j)$, and $b_{seed}(i, j)$ are the *L*, *a*, and *b* values for $p_{seed}(i, j)$. A_1, A_2, A_3, A_4, and A_5 are the parameters.

The segmented image in Fig. 5 cannot be directly used for IEC; we need to unify these small pieces side by side if both colorations are similar. We can then obtain the resulting image in Fig. 6 that has a coarsely objective shape. This method of segmenting images will also be discussed later. I will next discuss the relationship between image segmentation and color transfer.

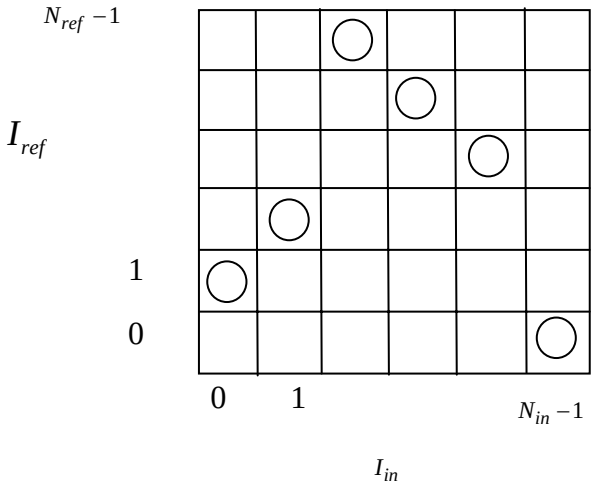


Fig. 7: Image-region matching table, T_r

My color-transfer algorithm is a kind of combinatorial optimization problem. Previous work on this subject has dealt with color transfer as an "image-region matching table" [9] between a photograph and a painting. Figure 7 has an example of image-region matching table T_r. The regions in target photograph I_{in} are represented as $I_{in}(i)(i = 0, \ldots, N_{in} - 1)$ and the ones in reference I_{ref} are represented as $I_{ref}(j)(j = 0, \ldots, N_{ref} - 1)$. We then determine pair $(I_{in}(i), I_{ref}(i))$ and assign a mean color, which is calculated with Eq. 6. $L_{ref}(x, y)$, $a_{ref}(x, y)$, and $b_{ref}(x, y)$ are the values for pixels in *CIE L*a*b** color space, where $n_{ref}(j)$ is the total number of pixels

in $I_{ref}(j)$. The circles in the image-region matching table indicate a color in $I_{ref}(j)$ has been assigned to $I_{in}(i)$. This image-region matching table is useful for transferring colors, e.g., it is possible to closely match the shapes of the sky and mountains (Fig.1) in the target photograph to the sky and houses with hills (Fig. 2) in the reference painting.

$$\langle L_{ref}(j)\rangle = \frac{\sum_{x,y\in I_{ref}(j)} L_{ref}(x,y)}{n_{ref}(j)}$$

$$\langle a_{ref}(j)\rangle = \frac{\sum_{x,y\in I_{ref}(j)} a_{ref}(x,y)}{n_{ref}(j)} \quad \text{..... (6)}$$

$$\langle b_{ref}(j)\rangle = \frac{\sum_{x,y\in I_{ref}(j)} b_{ref}(x,y)}{n_{ref}(j)}.$$

However, this method causes another problem with the search space. That is, if the number of divided regions, N_{in} and N_{ref}, become larger, the search space $\left(N_{ref}\right)^{N_{in}}$ will become huge, and it will of course take more time to acquire an optimum solution. We thus have to find a better method of solving this problem. My approach is based on the intuitive idea that painters first assign rough colors on the canvas and when they draw and render the painting they determine the coloration for the finer regions. Based on this, I roughly divided the image, and then divided these coarse regions into finer regions. That is, I first prepared the coarse regions to assign rough colors, and I then prepared the finer regions. The main point here is that the process of assigning colors is hierarchical. I divided the image and prepared finer regions three times in my experiments using the method I have described. This process is illustrated in Fig. 8.

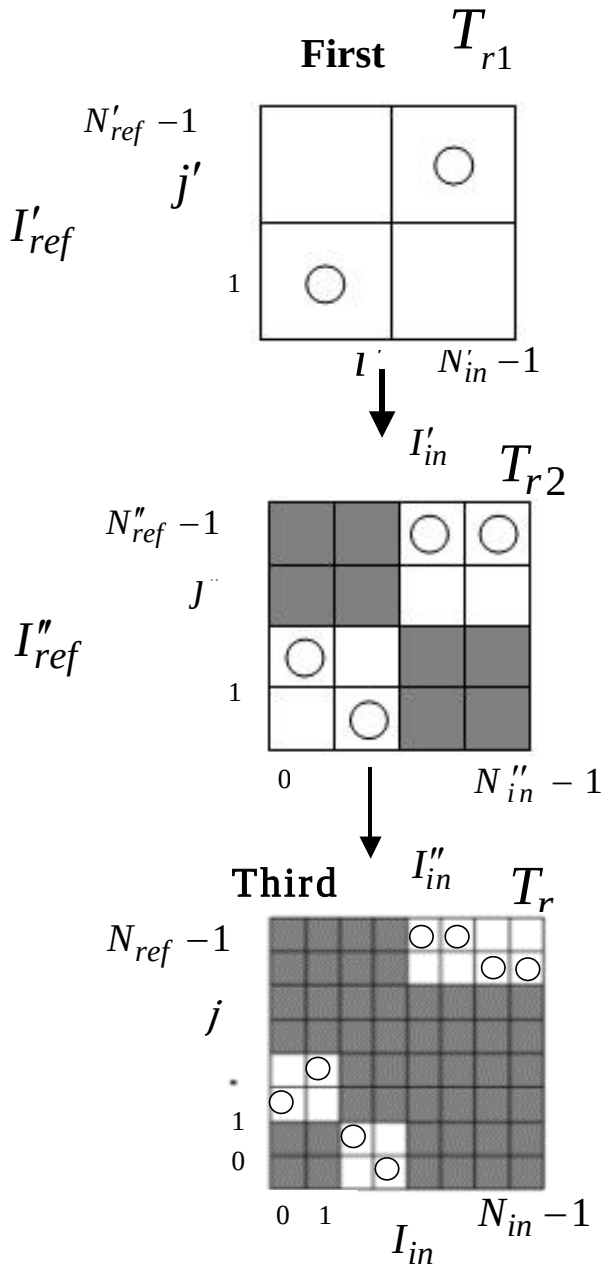


Fig. 8: Process of hierarchical image-region matching

The $I'_{in}(i')$ is the $i'-th$ region in the first divided target image and $I'_{ref}(j')$ is the $j'-th$ region in the first divided reference image. In the second dividing phase, $I''_{in}(i',i'')$ is the $i''-th$ region in the $I'_{in}(i')$ and $I''_{ref}(j',j'')$ is the $j''-th$ region in the $I'_{ref}(j')$. In the third dividing phase, $I_{in}(i',i'',i)$ is the $i-th$ region in the $I''_{in}(i',i'')$ and $I_{ref}(j',j'',j)$ is the $j-th$ region in the $I''_{ref}(j',j'')$. The gray areas in the image-region matching table represent the search

spaces eliminated through the previous steps. We can see the search space has been reduced with this method. When IEC finishes, all the matching pairs ($I_{in}(i',i'',i)$, $I_{ref}(j',j'',j)$) are decided. Based on these pairs, we can achieve finer color assignment.

3.4. IEC

3.4.1. Chromosome management

IEC manages and completes region-matching table T_r , where N_{ref} and N_{in} indicate the total number of $I_{ref}{}'s$ pixel regions and that of $I_{in}{}'s$. My approach uses the table itself as a chromosome for IEC. The expression of a chromosome is:

$$\left\lfloor x_0, x_1, \cdots, x_{N_{in}-1} \right\rfloor,$$

$$0 \le x_n \le N_{ref} - 1 \qquad (n = 0, \ldots, N_{in} - 1). \quad \text{..... (7)}$$

As I previously mentioned, I divided both the target photograph and reference painting three times to render finer regions. The image-region matching table and the chromosome were managed three times, from smaller to larger. This means the user does IEC three times as we can see from the image-region matching process in Fig. 8.

3.4.2. GA processing

The IEC operations I established can be illustrated as follows:

1. Initialize the initial adaptive probability, p_0, which is evaluated by the distance in color between $I'_{in}(i')$ and $I'_{ref}(j')$; this method is based on the idea that similar objects have similar colors in real photographs or in painterly art . For example, the color of the sea in both a photograph and a painting are a similar blue. This means that the probability that $I'_{ref}(j')$ will be assigned to $I'_{in}(i')$ will be higher if the color of $I'_{ref}(j')$ is closer to $I'_{in}(i')$. The IEC generates a population of M chromosomes under p_0 in this way.

2. Evaluate and award a fitness value of 0 or 1 to each chromosome in the population.

3. Generate a new population by executing the following steps.

   (i). Select parent chromosomes whose fitness value is 1 from a population, if there are more than two chromosomes whose fitness value is 1.

   (ii). Cross over the parents to form new offspring with the crossover probability. The crossover probability decides whether crossover is executed or not, and in my system this is set to 0.5. When crossover is executed, the crossover point is established at the locus between n and $n+1$, and n is the crossover point selected randomly in the chromosomes.

   (iii). Generate new offspring by randomly mutating all the offspring at each locus with the mutation probability. The mutation probability is set at 0.025.

   (iv). Replace the new offspring in a new population.

4. Use the new population that has been generated for a further run of the algorithm.

5. Stop and return the optimum solution if a user finds a chromosome that satisfies his subjective criteria in the current population.

6. Else go to Step 2.

T_{r1} is completed after these IEC operations. I used the completed T_{r1} and restricted the matching pair in the next phase table, T_{r2}, to reduce the search space. The final phase for T_r is done in the same way. These IEC operations were done in each table. The history of user evaluations in the evolution process is accurately reflected in the next generation. Figure 9 has examples of offspring generated by the third image-region matching table, T_r, which have been assigned mean colors.

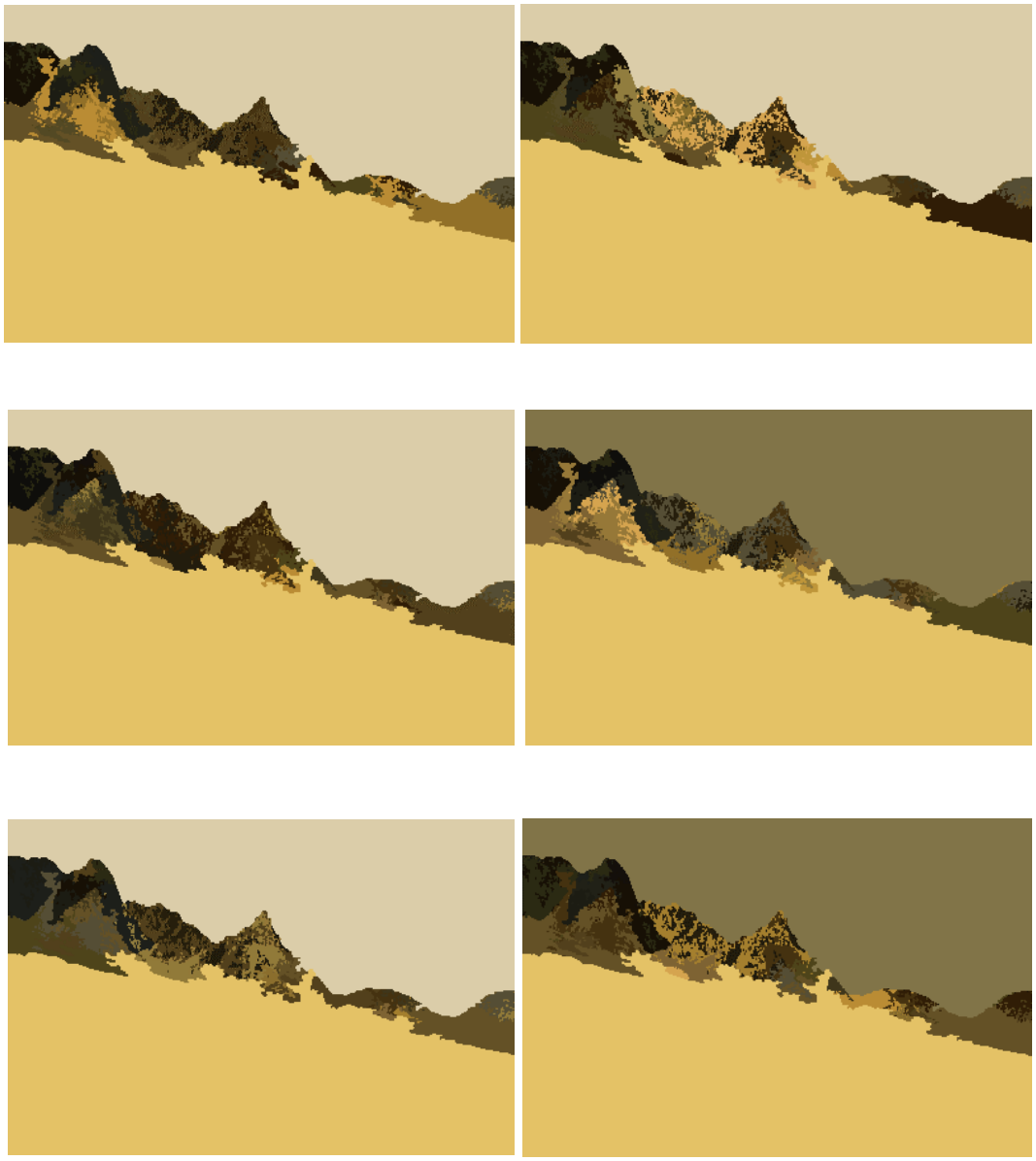


Fig. 9: Offspring generated by T_r

3.5. Color Transfer

After the IEC operations, the result image is coarsely color-assigned image with the mean colors of the reference painterly art, I_{ref}. This image-region matching table and the coarsely color-assigned image mean the color transfer relationship between target image I_{in} and reference painting art I_{ref}. Finer color-assigned operation is necessary after IEC in my algorithm. This approach is also based on the intuitive idea

that the painters roughly assign colors on a canvas first and when they draw and render a painting they determine the colors for the finer regions.

Some researchers have transferred color between images based on statistical analysis [5]. My aim was to introduce a simpler method of transferring color into my algorithm. This calculation is also based on the image-region matching table. It is given as:

$$p_{out}\left(I_{in}(i)\right) = p_{ref}\left(I_{ref}(j)\right),$$

$$p_{ref}\left(I_{ref}(j)\right) = \frac{n_{ref}(j)}{n_{in}(i)} \cdot S_{in(i)}(s). \quad \text{..... (8)}$$

$P_{out}\left(I_{in}(i)\right)$ is the pixel in the target-image region of $I_{in}(i)\left(i = 0,\ldots,N_{in}-1\right)$. $P_{ref}\left(I_{ref}(j)\right)$ is the pixel in the reference-painterly-art region of $I_{ref}(j)\left(j = 0,\ldots,N_{ref}-1\right)$. $n_{in}(i)$ is the total number of pixels in the region of $I_{in}(i)$. $n_{ref}(j)$ is the total number of pixels in the region of $I_{ref}(j)$. We then sort the pixel points in both regions of $I_{in}(i)$ and $I_{ref}(j)$ in the *CIE L*a*b** color space using the brightness value of L. $S_{in(i)}(S)$ is the *s-th* pixel in the sorted pixels of region $I_{in}(i), \left(s = 0,\ldots,n_{in}(i)-1\right)$. Using Eq. 8, we move the pixels from region $I_{ref}(j)$ to $I_{in}(i)$.

Of course, users can still obtain different results with IEC even though the same input target and reference images are used. This is illustrated in Fig.10; the results are similar, but not exactly the same.

Fig. 10: Similar results

3.6. System overview

Figure 11 overviews the system I developed. The user prepares two images, i.e., a target photograph, I_{in}, and a reference painting, I_{ref}. The system divides both images into coarse regions by exploiting the image segmentation that I previously mentioned. The regions acquired from I_{in} and I_{ref} constitute the xy-axes of the region-matching table, T_r. The IEC interface is located between the user and T_r. He or she evaluates and completes T_r through the IEC interface. Of course, the user does not need to handle T_r directly. He or she sees and confirms the color-assigned images through the interface, which are candidates for the solution the system has produced, and the user just awards a fitness of 0 or 1 to them. The system produces new candidates based on the fitness, and the user evaluates them again. Such processes are repeated until a candidate satisfies his or her subjective criteria.

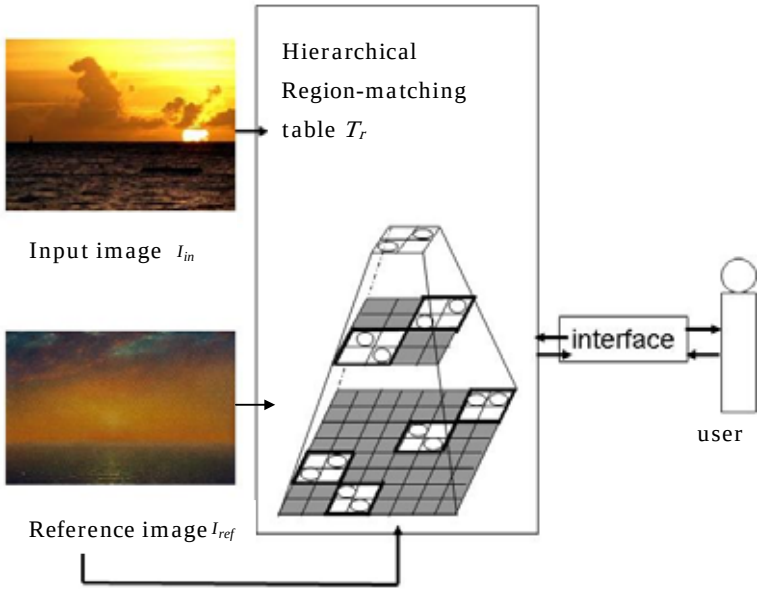


Fig. 11: System overview

3.7. Discussion of Results

Twenty graduate-school students of the Department of Computer Science and Engineering at the Nagoya Institute of Technology tested my system and obtained the following results. Figures 12 to 19 have eight sets of images. Each set has a target photograph, a reference painting, and a resulting image. Clear similarities between the

references and results can plainly be seen. These similarities not only concern color but also feelings. The system also has another special characteristic. Although users create results using the same pair of images (a target and a reference), their visual results might differ slightly, because the system outputs results based on their individual subjective criteria. I previously mentioned that IEC generates a population of *M* chromosomes with initial adaptive probability P_0, which is evaluated by the distance in color in the *CIE L*a*b** color space between $I_{in}(i)$ and $I_{ref}(j)$.

Figure 20 plots the mean processing time and Fig. 21 plots the mean total generations until users were satisfied with their final results. In both figures, the horizontal axis is the *ln(z)* and *z* is the search space, $\left(N_{ref}\right)^{N_{in}}$. Each x' out of {a', b', ..., g', h'} is a result produced with the initial adaptive probability, and each x" out of {a", b", ..., g", h"} is a result produced without it, only randomly. Both x' and x" are created using the same pair of a target (x) and a reference (x) illustrated in Figs.12 to 19. Of course, there are some visual difference between x' and x" as I previously mentioned. We can see that the processes with P_0 are more efficient than those without P_0. As we can observe that P_0 is evaluated by the distance in color between $I'_{in}(i')$ and $I'_{ref}(j')$ in the *CIE L*a*b** color space, then if $I'_{in}(i')$ is nearer to $I'_{ref}(j')$, the probability of $I'_{ref}(j')$ being assigned to $I'_{in}(i')$ is higher. Although this works well in many cases, it is not optimal. As it did not work as well as I wanted in some other cases, I need to find a more efficient method in future work. The search space for both x' and x" depends on target (x) and reference (x) because search space z = $\left(N_{ref}\right)^{N_{in}}$. Although the set of images from Figs. 12 to 19 are arranged in ascending order of z, it may be difficult to recognize the search space at a glance. As we can see from Figs. 20 and 21, x' and x" are also arranged in order. The processing time includes the calculation time for the computer, and the total thinking and operating time spent by users in considering and determining the fitness values. Although the total generation has no factors corresponding to users' thinking time, this is similar to the processing time required by the computer. The processing time and total generation increase much more as the search space increases. Of course, there are some occasional exceptions when the processing time and total generation are less with smaller search spaces, because the user needs to spend less time thinking even though the searching space is much bigger.

My system operated on a PC with a Celeron 2.4-GHz CPU and 512 MB of memory, running Sun Microsystem's Solaris 9 operating system.

Figure 22 has some more evaluation results for the system. When test subjects finished creating five color-transfer images, I required them to evaluate the system's validity or effectiveness in transferring colors. They awarded grades based on a five-point Likert scale from A to E. A was the best and E was the worst. Before grading, I explained that they could evaluate the system by awarding a grade depending on their own subjective criteria.

We can see that about half awarded the best grade (Fig. 22). There were only twenty subjects at that time; however, I was surprised by the results. Some also made comments and suggestions as to why they awarded scores.

1. Those who graded the system with an “A” thought color transfer were a good idea in that they could effectively exert their imaginations and feel like artists who applied their paintings to a canvas.

2. Although those who graded it with a “B” approved of its transfer of colors, they thought it could not transfer any of the important textural characteristics in the reference painting. They thought the system could be improved if it could transfer textures.

3. Those who graded it with a “C” thought that it took too long to search for satisfactory results in some cases. However, they would have liked to use it more if the search time could be reduced because it would not then be too hard for them to transfer colors.

4. Those who graded the system with a “D” thought that too many strange candidates appeared during IEC processing. They thought if it could reduce the number of peculiar candidates, appropriate candidates would appear sooner, and it would then be more beneficial.

5. Those who graded it with an “E” thought it was very unsatisfactory; they claimed it was useless because they could not obtain any results unless they initially divided the image well.

Based on these comments, I came to the following conclusions.

3.8. Conclusions

I proposed an algorithm in my thesis that could transfer color from a painting to a photograph with IEC. This method is also based on hierarchical image segmentation. In some cases, I could not divide an image sufficiently well and thus could not obtain a satisfactory result. I intend to continue my research to develop a better and more efficient way of dividing an image.

Another problem was the processing time. Some output examples were obtained in the experiments, and I evaluated the processing time, as shown in Fig. 20. Although this largely depends on the PC specifications, my system takes far too long for processing. The processing time is one of the most serious factors affecting an IEC system because the faster the processing, the fewer the manual tasks involved. I need to resolve this problem immediately. I prepared comparatively simple images for the experiments, and I did not need to divide these into many regions. If the processing time can be reduced, more complex ones can be handled. I are also considering some other methods where the system can process regions when they appear to satisfy the region-matching results; it can then continue to treat these regions with GA. I intend to continue to develop a more efficient region-matching method in future work.

The last problem is the transfer of textures. My system just transferred colors on the basis of region-matched information recorded on the chromosome. Region-matched information can also be applied to transferring textures. If we want to transfer textures from a reference painting to a target image to produce a resulting image, we have to utilize region-matching information, e.g., if there are sea and sky in both the target photograph and the reference painting, we obviously need to transfer texture from sky-to-sky and sea-to-sea; if not, the result will appear bizarre. My research has taken us

a step closer to making texture transfer feasible. However, these problems have also been left to be addressed in future work.

3.9. Addenda: Contrast Between Methods in Chapters 2 and 3

Chapters 2 and 3 discuss two methods of transferring colors from painterly art to a photograph to produce NPR. Although both these methods can produce satisfactory results, there are three major differences, which are listed below.

1. First, the candidates produced by both methods were different. Those produced with the method in Chapter 2 were not colored entirely. However, those produced with the method in Chapter 3 were colored throughout. Therefore, the system in Chapter 3 is more efficient and less time consuming than that in Chapter 2.

2. Second, the processing time for the method in Chapter 3 was less than that for the one in Chapter 2. This time included the calculation time for the computer and the total thinking and operating time by users, who needed to consider and determine the fitness values. They obviously needed more time for thinking about and considering candidates when they used the system in Chapter 2, because these were not colored entirely.

3. Third, the method discussed in Chapter 3 could effectively reduce search space. The gray areas in the hierarchical region-matching table were the search spaces that were eliminated.

In many cases, the processing time in Chapter 3 was equal to or less than the processing time in Chapter 2. I input same pairs of images in both of the systems in Chapters 2 and 3, to compare the processing time. Fig.12 to19 are images which I input into both of the systems in chapter 2 and 3. Fig. 23 plots the mean processing time with both methods in chapter 2 and 3. In the figure, the horizontal axis is the *ln(z)* and *z* is the search space, $\left(N_{ref}\right)^{N_{in}}$. Each x out of {a, b, ..., g, h} is a result produced with the method in chapter 2, and each x" out of {a", b", ..., g", h"} is a result produced with the

method in chapter 3. I stopped the process if the time was over 800 seconds. It is obvious that the method in chapter 3 is faster than the method in the chapter2.

3.10. References

1. GOOCH, B. AND GOOCH, A., Non-Photorealistic Rendering, A. K. Peters Ltd., Natick, MA, USA, 2001.

2. HERTZMANN, A., JACOBS, C.E., OLOVER, N., CURLESS, B. AND SALESIN, D.H., Image Analogies, Siggraph 2001 Conference Proceedings, pp. 327–340, 2001

3. WANG, B., WANG, W., YANG, H. AND SUN, J., Efficient Example-based Painting and Synthesis of 2D Directional Texture, IEEE Transactions on Visualization and Computer Graphics, 10 (3), pp. 266–277, 2004

4. CHANG. Y., SAITO, S. AND NAKAJIMA, M., Color Transformation Based on Basic Color Categories of a Painting, Siggraph 2002 Conference, Abstracts and Applications, pp. 157–157, 2002

5. REINHARD, E., ASHIKRHMIN, M., GOOCH, B. AND SHIRLEY, P., Color Transfer between Images, IEEE Computer Graphics and Applications, 21 (5), pp. 34–41, 2001

T. Kawashima, M. Kimura, T. Nakamura, H. Itoh, and L. He, Painterly Rendering Designed for Impressionist-Art Representation with Multi-resolution Color Matching, The 7th International Conference on Intelligent Autonomous, pp. 158–165, 2004

6. Takagi, H., "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation." Proceedings of the IEEE. 89, 9, pp. 1275-1296, 2001.

7. KATAGAMI, D. AND YAMADA, S., Interactive Evolutionary Computation for Real Robot from the Viewpoint of Observation, 7th International Conference on Intelligent Autonomous (IAS-2002), pp.158-165, 2002.

8. TOKUI, N. AND IBA, H., Music Composition with Interactive Evolutionary Computation, 3rd International Conference on Generative Art, 2000.

9. Li , Q., Nakamura T., Chao Y., He L., and Hidenori Itoh, Color Transfer between Images with IEC, Forma. 2004.

Target (a)

Reference (a)

Result (a)

Fig. 12. Experimental result 1

Target (b)

Reference (b)

Result (b)

Fig. 13. Experimental result 2

Target (c)

Reference (c)

Result (c)

Fig. 14. Experimental result 3

Target (d)

Reference (d)

Result (d)

Fig. 15. Experimental result 4

Target (e)

Reference (e)

Result (e)

Fig. 16. Experimental result 5

Target (f)

Reference (f)

Result (f)

Figure 17: Experimental result 6

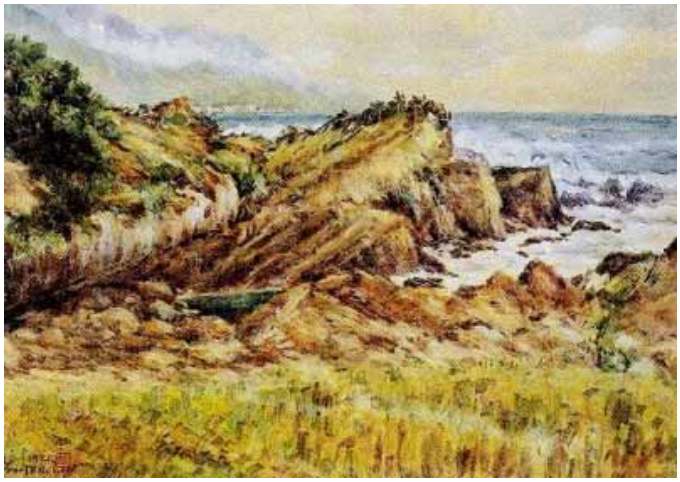
Target (g)

Reference (g)

Result (g)

Fig. 18. Experimental result 7

Target (h)



Reference (h)

Result (h)

Fig. 19. Experimental result 8

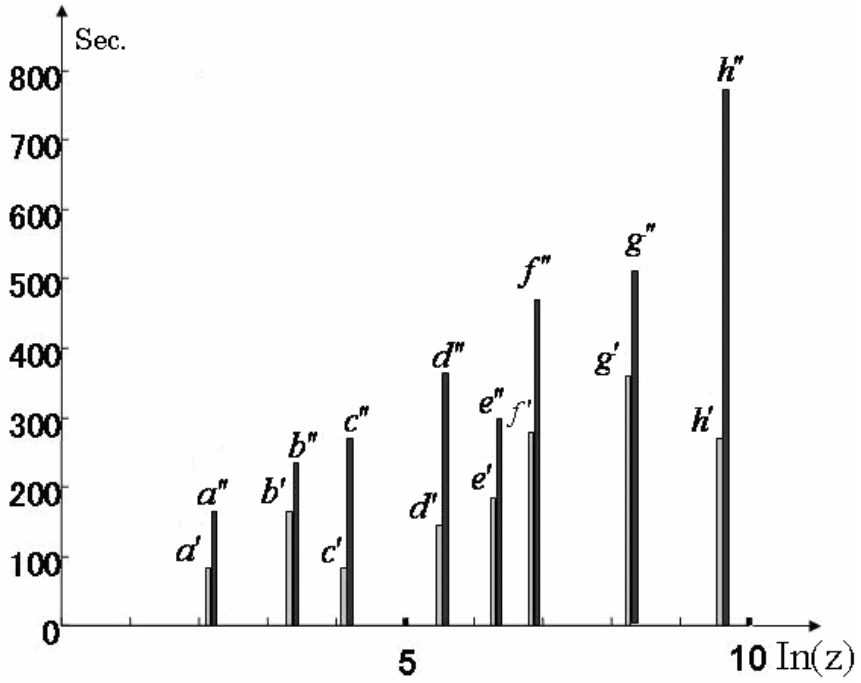


Fig. 20. Processing time

(z = $\left(N_{ref}\right)^{N_{in}}$)

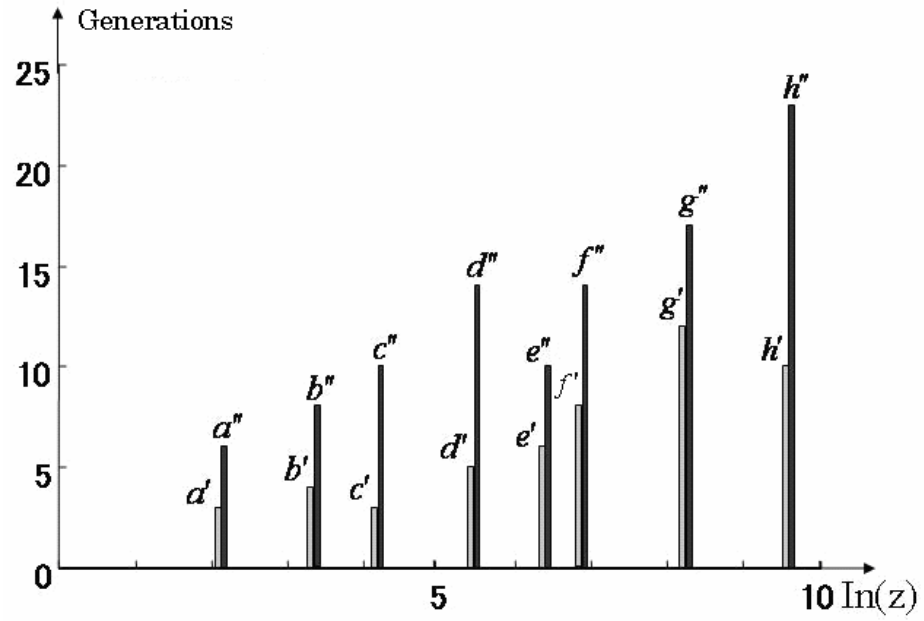


Fig. 21. Generations

(z = $\left(N_{ref}\right)^{N_{in}}$)

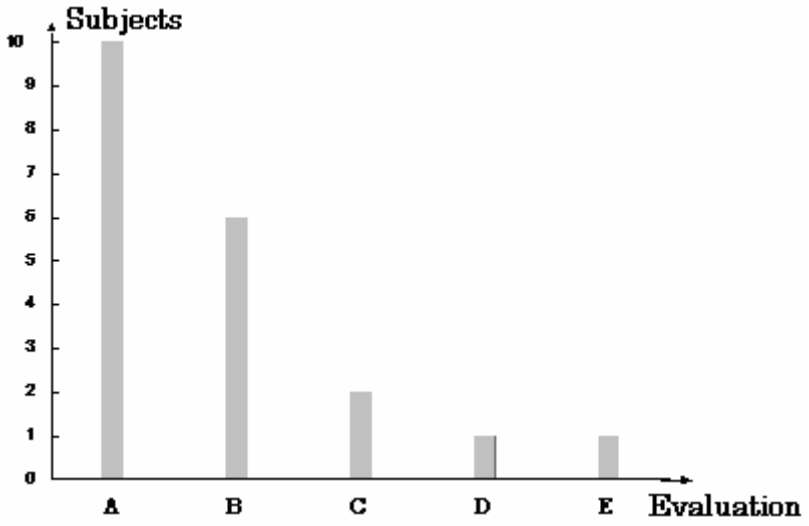


Fig. 22. Evaluation results

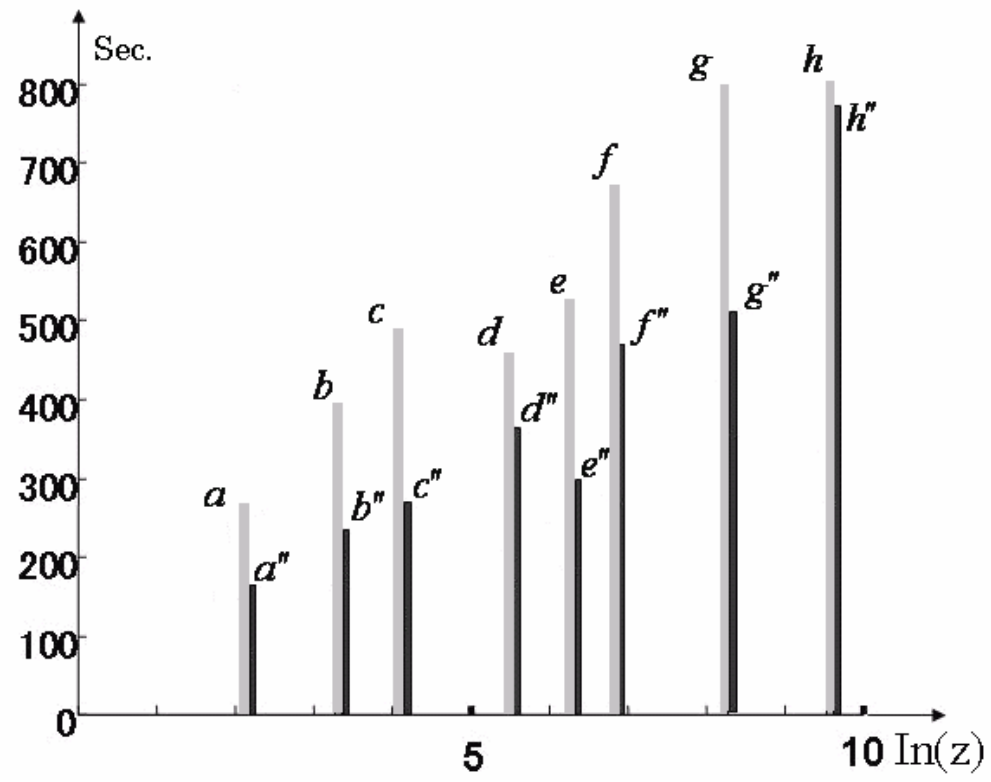


Fig. 23. Compare of processing time

$$(z = \left(N_{ref}\right)^{N_{in}})$$

CHAPTER 4
FUNDAMENTAL STUDY ON GENERATING CALLIGRAPHY USING FONTS MANUALLY PREPARED BY ARTISTS AS REFERENCES

4.1. INTRODUCTION

This chapter presents an approach to generating artistic calligraphy, which looks scratched or blurred, like those rendered by famous calligraphers. Although I have obtained some intermediate results for this project, final results still need to be obtained. If this algorithm can be more finely tuned, far superior results can be attained. In the following, I will explain the method and the algorithm, and discuss future problems with this project.

There are many calligraphic fonts such as TrueType and Bitmap, but the software cannot generate calligraphic fonts that look scratched and blurred. Assistant Professor Tsuyoshi Nakamura of the Department of Computer Science and Engineering at the Nagoya Institute of Technology has written a system that can generate various calligraphic characters that look scratched or blurred and has applied for a patent. Users have been satisfied with the speed with which it generates calligraphic characters. The system involves two processes. In the first, an original calligraphic font is input, and then transformed into a bitmap image. In the second, the skeleton of the image is found by using a thinning algorithm. Furthermore, brush-touch cursors are placed on each pixel of the skeleton [1-6]. The brush-touch cursors

are expressed as dot patterns. Calligraphic characters can be made to look scratched and blurred using these two processes. Various degrees of scratching and blurriness can be obtained for artistic calligraphic fonts by adjusting some parameters.

The brush-touch texture with this method involves using only one cursor, as seen in Fig. 1 below.

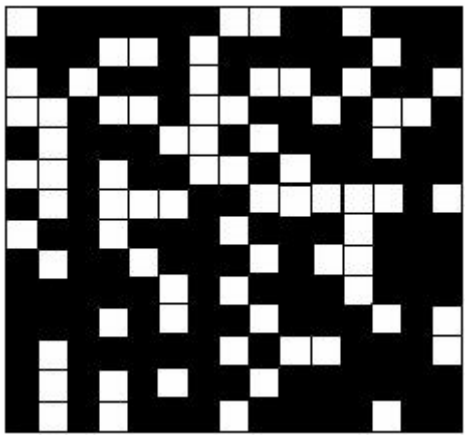


Fig. 1. Example brush-touch cursor

The system places this brush-touch cursor on each pixel of the skeleton to generate the artistic calligraphic fonts. As previously mentioned, various degrees of scratching and blurriness can be obtained for artistic calligraphic fonts by adjusting some system parameters. Although this method produces wonderful results in many cases, it creates a different look to the calligraphy done by renowned specialists. I wanted to develop a method that could extract the unique characteristics from calligraphic fonts created by famous calligraphers, and pass these characteristics on to original calligraphic fonts like TrueType or Bitmap. This method is similar to the texture synthesis I previously mentioned. Efros and Freeman [7-10] developed a method of synthesizing texture. It directly removed patches from given input images, then synthesized the patches to form a new image with the same texture as that in the input image. It could also transfer texture from one input image to a target image, to make it look like the given image. Its most important feature is that the same patches are used in both the input and synthesized images. I wanted to apply this method to obtain scratched and blurred results for the reference font created by a renowned calligrapher. Had I succeeded completely, I would have been able to generate a calligraphic font that would

have resembled a calligrapher's font more than the calligraphy font generated by the previous system.

The method I propose also encountered a problem with the directivity of calligraphic fonts. Common textures have no directivity, but directivity in calligraphic fonts is obvious and important. The generated font will appear peculiar if directivity in texture-synthesis processing is not taken into account.

The other problem I faced in this research was the integrity of the stroke; this meant that all the strokes in the generated font needed to look like those in the reference font, except for some patches. This was very difficult to achieve, because I could not identify whether all the strokes looked like the references from the intermediate results except for some patches. I need to change the algorithm to improve the quality of the results. The following explains my method.

4.2. FONT THINNING

I had to obtain the skeleton for the font, and the thinning algorithm I used to find it was the Hilditch [11] algorithm. I first converted the input font into a bitmap image. The blackness pixel in the bitmap image was set to 1, and the whiteness pixel in the bitmap image was set to 0.

The algorithm can be presented as follows. The outermost layer of the font was cut off step-by-step until the skeleton was reached; of course, the thickness of the skeleton was just one pixel. If the time to cut off the outside layer of the font is t ($t \geq 0$), the thickness of the skeleton will probably be t. Figure 5 has one example of a thinned font.

4.3. REFERENCE-FONT DATABASE

I built up a reference database that contained many script/calligraphic fonts. As these looked scratched and blurred, the following process has to be done on them:

1. Convert the image file into a bitmap file if it is not a bitmap file.
2. If part of the font looks scratched blacken it.

3. Thin the font using Hilditch algorithm.

4. Place the skeleton in the original script/calligraphic font.

5. Then find the contour line for the font using the blackened font.

These steps are in Figs. 2 to 6. Figure 7 shows how to build a scratched calligraphic-font database.

Fig. 2 Script/calligraphy font

Fig. 3 Converted to bitmap font

Fig. 4 Non-scratched font

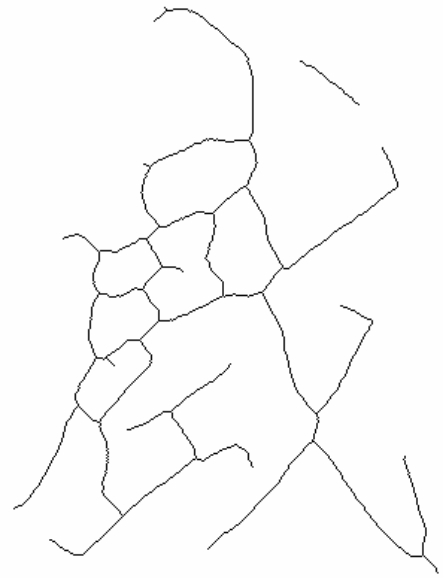


Fig. 5 Skeleton of font

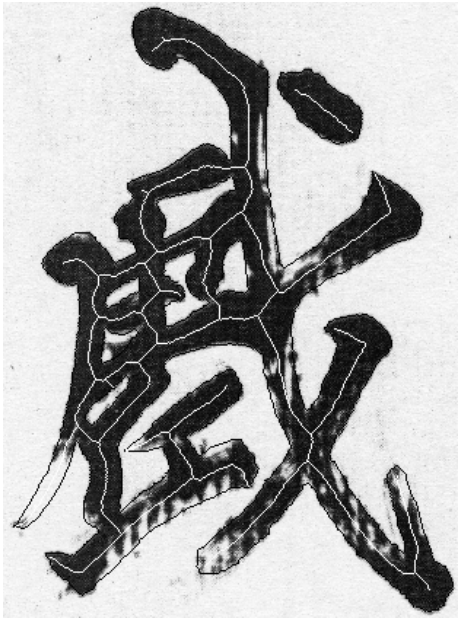


Fig. 6 Font with contour lines and skeleton

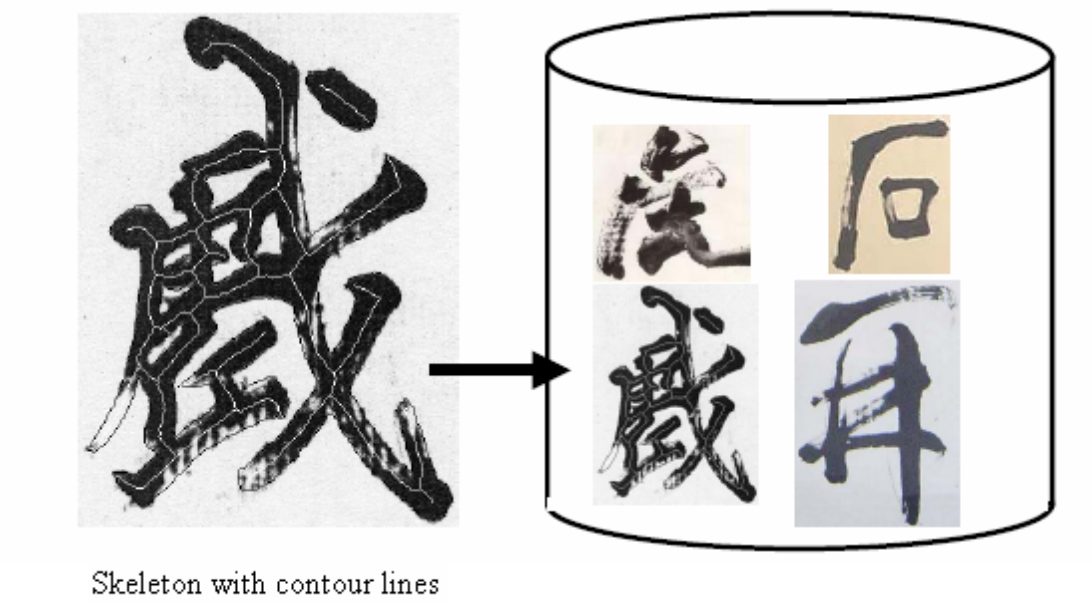


Fig. 7 Scratched calligraphic-font database

4.4. SEAMLESS TEXTURE MAPPING

4.4.1. Introduction of Quilting for Synthesizing Textures

I have already mentioned Efros and Freeman's [12-21] method of synthesizing textures. They removed patches of texture from given input images and then seamlessly quilted these

texture patches together. The quilting algorithm they presented found the minimal-cost path through the overlapping error surface. For example, if B_1 and B_2 are two blocks that overlap along their edges with B_1^{OV} and B_2^{OV} regions of overlap, then the error surface can be defined as $e = \left(B_1^{OV} - B_2^{OV}\right)^2$. I traversed $e \quad (i = 2..N)$ and computed the cumulative minimum error, *E*, for all paths to find the minimal cut through this surface, i.e.,

$$E_{i,j} = e_{i,j} + \min\left(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}\right) \quad \text{..... (1)}$$

The minimum value for the last row in E will indicate the end of the minimal path though the surface and one can track back and find the path for the best cut.

I applied this method to remove texture patches from a scratched calligraphic font saved in the database. I then placed a patch on the target font. To do this, I had to find the skeleton for the font. Figure 8 has a target font with skeleton lines.

Fig. 8 Target font with skeleton

4.4.2. Overview of Texture Mapping

I transferred a scratched texture from the reference font to the target font by quilting patches taken from the reference. This method is outlined in Fig. 9. This method should generate a scratched and blurred font similar to the reference, because it is generated by patches directly cut from the reference font. Efros and Freeman also presented a method of transferring textures. They augmented the synthesis algorithm by requiring each patch to satisfy a wanted correspondence map, $\vec{C}$, as well as satisfy the texture-synthesis

requirements. The correspondence map is a spatial map of some quantities corresponding to both the texture-source and controlling-target images. These could include image intensity, blurred-image intensity, local-image orientation angles, or other quantities that are derived. In their experiments on texture transfer, their correspondence maps were the (luminance) image intensities. The image being synthesized in their texture transfer had to conform to two independent constraints: (1) the output needed to be legitimate, synthesized examples of the source texture, and (2) that mapping of corresponding images needed to be done. They modified the error term for the image-quilting algorithm to be a weighted sum, i.e., α times the block overlap matching error plus (1- α) times the squared error between the correspondence-map pixel within the source texture block and those at the current target-image position. The parameter, α, determines the tradeoff between texture synthesis and fidelity to the correspondence map of the target image. In my research, the luminance of the font was black; therefore, α was set to 1. Texture synthesis was only determined using the texture patches without the correspondence map.

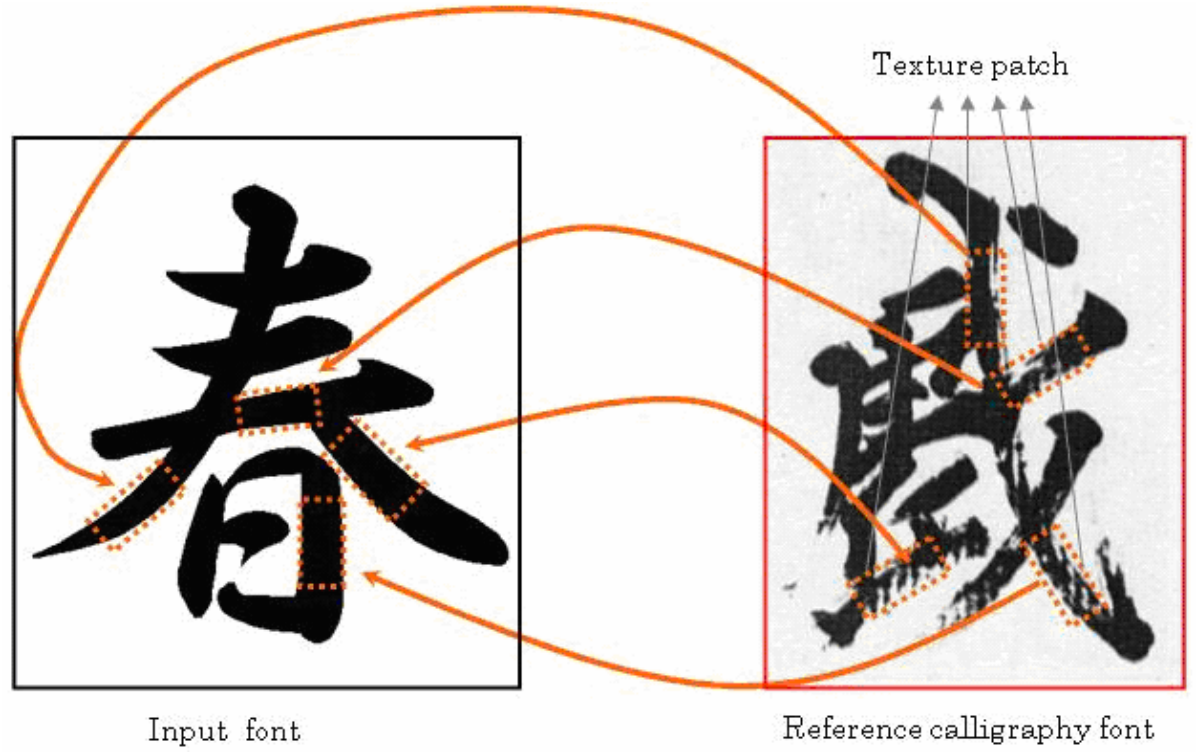


Fig. 9 Texture transfer to input font

4.5. Experimental Results

4.5.1. Intermediate Results

Texture patches were transferred from the reference to the target font in the intermediate results. As this experiment was not completely finished, I just transferred texture patches to one of the strokes. What I wanted was to transfer the entire stroke from the reference to the target stroke; this meant I wanted to transfer the entire character in the stroke. However, the results were not as I expected, only some patches looked like the reference stroke not the whole stroke. The intermediate results reveal that this algorithm can quilt patches seamlessly, but it cannot identify the entire character of the stroke in my target stroke. The algorithm has to be improved to solve this problem. Before discussing the improvements, I will explain how the intermediate results are generated.

4.5.2. Method of Quilting Patches

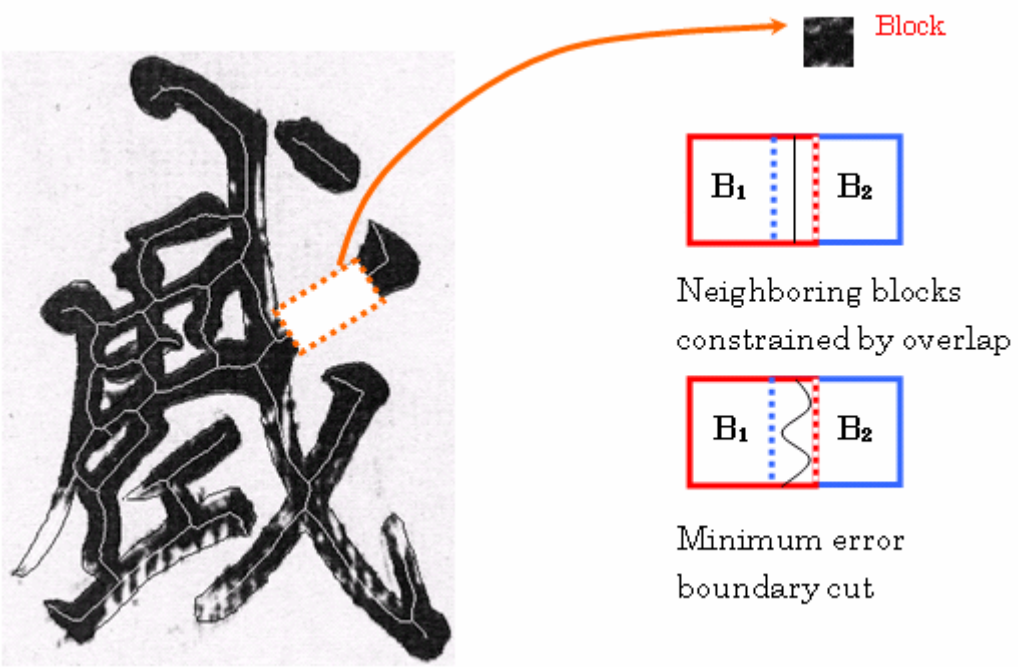


Fig. 10 Quilting patches

- First cut the patch from the reference font.
- Then place it on the target font.
- The next patch is constrained by overlap.
- The minimum error boundary cuts the overlap.

- Repeat the steps until the texture is transferred to all fonts.

I found why the transfer could not remove whole characters from the reference font by quilting the patches. This was because the next patch was constrained by overlap. In certain cases, it is possible for transfer to be successful, e.g., if the target and reference strokes have the same length, orientation, and breadth. In fact, those cases are very limited. In most cases, the length, orientation, and breadth are not the same. Therefore, I have to develop a new algorithm to deal with this problem.

4.6. CONCLUSION

I introduced patch-quilting in this chapter, which is a method of transferring texture to generate a new calligraphic font by stitching together small patches of a reference font. Despite the unsatisfactory results, I developed a new idea for generating scratched and blurred calligraphic fonts. Although there were no problems with seamless patch quilting, the finished look I expected could not be achieved. I also discussed why the system could not generate the finished look of transferred textures. I intend to improve this algorithm to deal with this problem and extend it to generate satisfactory results.

4.7. REFERENCES

1. Mano, J., Nakamura, T., Seki, H., and Itoh, H.: A Scratched-look Expression of Calligraphy Characters Using Renormalization Group, International Fuzzy System and Intelligent Control Conference, pp. 93–102, 1996.

2. 真野淳治、中村剛士、世木博久、伊藤英則：毛筆書体におけるくりこみ群を用いたかすれ、滲み表現、情報処理学会論文誌、Vol. 38, No. 4, pp. 806–814 1997.

3. Nakamura, T., Itoh, H., Seki, H., and Law, T.: A Writing System for Brush Characters Using Neural Recognition and Fussy Interpretation, International Joint Conference on Neural Networks, pp. 2901–2904, 1997.

4. Nakamura, T., Kuroda, T., Itoh, H., and Seki, H.: A Calligraphy System Based on Analyzing User Writing Speed, 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing, pp. 189–190, 1994.

5. Nakamura, T., Kuroda, T., Itoh, H., and Seki, H.: Fuzzy-based Writing System for Acquiring Good Writing Skill of Brush Characters Based on the Analysis of Writing Speed, 3rd Pacific Rim International Conference on Artificial Intelligence, Vol. 2, pp. 822–827, 1994.

6. Nakamura, T., Nozaki, K., H., and Itoh, H.: A Fuzzy-based Calligraphy System Using Fractals, Third European Congress on Intelligent Techniques and Soft Computing, pp. 1440–1444, 1995.

7. A. Efros and W. Freeman. Image Quilting for Texture Synthesis and Transfer. In Proceedings of ACM Siggraph 01, pp. 341–346, 2001.

8. Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In Siggraph 00, pp. 479–488, 2000.

9. S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields, and maximum entropy (frame). International Journal of Computer Vision, Journal of Computer Vision, 27 (2): 1–20, March/April 1998.

10. Y. Xu, B. Guo, and H.-Y. Shum. Chaos mosaic: Fast and memory efficient texture synthesis. Technical Report MSR-TR-2000-32, Microsoft Research, April 2000.

11. Hilditch, C.: Linear Skeleton from Square Cupboards, Machine Intelligence, Vol. 6, pp. 403–420, 1969.

12. Y. Chang, S. Saito, and M. Nakajima: A Framework for Transfer Colors Based on the Basic Color Categories. Proceedings of the Computer Graphics International (CGI' 03) 1530–1052, 2003 IEEE.

13. E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley: Color Transfer Between Images. IEEE Computer Graphics and Applications, Vol. 21(5): 34–41, September 2001.

14. B. Wang, W. Wang, H. Yang, and J. Sun: Efficient Example-based Painting and Synthesis of 2D Directional Texture. IEEE Transactions on Visualization and Computer Graphics, Vol. 10, No. 3, May/June 2004.

15. T. Welsh, M. Ashikhmin, and K. Mueller: Transferring Color to Greyscale Images. Proceedings of Siggraph, San Antonio, USA, 277–280, 2002.

16. A. Hertzmann, C. E. Jacobs, N. Olover, B. Curless, and D.H. Salesin. Image analogies. In Siggraph 01, 2001.

17. L. Liang, C. Liu, Y. Xu, B. Gou, and H. Y. Shum. Real-time texture synthesis by patch-based sampling. Technical Report MSR-TR-2001-40, Microsoft Research, March 2001.

18. J. Portilla and E.P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. International Journal of Computer Vision, 40 (1): 49–71, December 2000.

19. P. Harrison. A non-hierarchical procedure for re-synthesis of complex textures. In WSCG' 2001 Conference proceedings, pages 190–197. Also see http://www.csse.monash.edu.au/~pfh/resynthesizer/, 2001.

20. V. Ostromoukhov and R. D. Hersch. Multi-color and artistic dithering. In Siggraph 99, pp. 425–532, 1999.

21. E. Praun A. Finkelstein, and H. Hoppe. Lapped textures. In Siggraph 00, pp. 465–470, 2000.

List of Publications

Journal Articles

1. Q. Li, T. Nakamura, Y. Chao, L. He, and H. Itoh.
   Color Transfer between Images with IEC *Forma*, Vol. 19, No. 3, pp. 207−222, 2004.

2. Q. Li, T. Nakamura, Y. Chao, L. He, and H. Itoh.
   Color Transfer Based on Hierarchical Image−region Matching With Interactive Evolutionary Computation
   Kansei Engineering International. Vol. 5, No. 4, pp. 1−10, 2006.

International Conference Proceedings

1. Q. Li, T. Nakamura, Y. Chao, L. He, and H. Itoh.
   Image Color Transfer with Region Matching Based on IEC
   IEEE CEC, 2005 IEEE Congress On Evolutionary Computation. pp. 1501−1508, 2005.

Domestic Conference Proceedings

1. Q. Li, T. Nakamura, Y. Chao, L. He, and H. Itoh.
   Color Transfer Based on Image−Region Matching with IEC
   日本感性工学会春季大会, Spring Conference 2005 of Japan Society of Kansei Engineering, 信州大学繊維学部キャンパス, pp. 156−159, 2005. 3. 23−24.

2. Q. Li, T. Nakamura, Y. Chao, L. He, and H. Itoh.
   Color Transfer between Images with IEC
   第１９回日本東海ファジィ研究会, 6−1, 2005.