Algorithmic Painter: a NPR method to generate various styles of painting

Atsushi Kasao[1], and Kazunori Miyata[2]

[1] *Tokyo Polytechnic University, Department of Design, Faculty of Arts,*

2-19-3 Honchou, Nakanoku, Tokyo, 164-8678, Japan

[2] *Japan Advanced Institute of Science and Technology, Center for Knowledge Science*

1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan

[1]kasao@dsn.t-kougei.ac.jp, +81-3-5371-2728 (Phone & Fax)

[2]miyatak@acm.org, +81-761-51-1810 (Phone), +81-761-51-1804 (Fax)

Authors' biographies

Atsushi Kasao has been a Associate Professor in the Faculty of Arts at the Tokyo Polytechnic University since 1998. Prior to joining the Tokyo Polytechnic University, he was a researcher of Fuji Xerox Research Center. He received a B.S. degree from Gakushuin University in 1982, and a M.S. and Ph.D. from the Tokyo Institute of Technology in 1985 and 1999. His research mainly focuses on computer graphics, algorithmic art, painterly software, and multimedia applications. He is a member of ACM and IEEE.

Kazunori Miyata has been a Professor in the Center for Knowledge Science at the Japan Advanced Institute of Science and Technology (JAIST) since 2002. Prior to joining JAIST, he was an Associate Professor in the Department of Imaging Art at the Tokyo Institute of Polytechnics. He received a B.S. degree from Tohoku University in 1984, and a M.S. and Ph.D. from the Tokyo Institute of Technology in 1986 and 1997. His research mainly focuses on fractals, rendering & modeling natural objects, texture generation, and multimedia applications. He is a member of ACM and IEEE.

Keywords

non-photorealistic rendering, painting, brush stroke, image segmentation

Abstract

This paper proposes “Algorithmic Painter”, an algorithm which can produce various styles of painting from source photos. Algorithmic Painter is created by enhancing Synergistic Image Creator, is a painterly rendering method, to be highly expressive but to still reserve the essential characteristics of the source photo. To achieve this, our method extracts three types of image segments automatically from a source photo by newly proposed classification method: edge areas, homogeneous areas, and highly contrastive areas. Next, each obtained image segment is converted into a brushstroke. Finally, the target picture is rendered by assigning a color to each pixel. Furthermore, this method can control the curvy shapes of brushstrokes, so that the obtained image can not only incorporate various artistic touches but also natural touches. As an example, the paper describes the composition technique for three considerably different painting styles.

1 Introduction

Computer-generated imagery methods are progressing in two directions which differ in expressional quality: photorealism and non-photorealism.

The investigation of photorealism in computer graphics is evolving within the paradigm of the photograph. Images are rendered by means of physical simulations which emulate the interaction between lights and materials.

On the other hands, NPR methods pursue communication and stylization. Recent image processing studies also have been concerned with methods for generating artistic impressions, which involve using specific processing techniques such as the use of special image filters. These studies can be applied to promote novel artistic expression, and also to implement a tool which helps people create artistically expressive images without actual painting skill.

This paper proposes a novel NPR method called “Algorithmic Painter”, which can generate various styles of painting.

The final purpose of this study is to propose a method to create a computer generated image as artwork, using a photo as source material. Our goal is not just painter-like expression to the source photo but to create an artistically viable piece. Therefore, we start by thinking what information is needed for this kind of artwork creation framework. Important information in the source photo varies according to the creator. Actually, a painting style deeply depends on information which each painter picks up from the scene. Therefore, the important thing for *Algorithmic Painter* is segmenting and classifying information in the source photo. After classification, CG creators can choose the information they want and arrange it to create their artworks. As for a method to segment information, we need to realize the importance of the special rhythm created by a segmenting-information method. The segmented information should be treated as a minimum unit for the artwork. So, if the segmented information does not have appropriate special rhythms, it is hard to create them.

We had proposed *Synergistic Image Creator*, which segments information in source photos with a K-means algorithm. The reason to use a K-means algorithm is that this algorithm can combine the original special frequency from the source photo with that of the creator. However, *Synergistic Image Creator* can not classify the minimum unit information, create large segments by combining adjacent segments nor create expressive curvy brush strokes. That is why a new painting system is needed to solve these problems. Lastly, *Algorithmic Painter* is developing a good record: artworks created by *Algorithmic Painter* have been recognized at international 2D-CG contests like SIGGRAPH 2002 [21], 2003 [22], the Media Art Festival 2004 in Japan [23] and so on.

This paper is organized as follows. Section 2 gives an overview of related works in the field and indicates the differences between previous efforts and our method; it also describes the "Synergistic Image Creator" [10], which is the basic scheme underlying the proposed method. Section 3 describes our image expression algorithm. Section 4 uses examples to demonstrate some results, and finally, Section 5 gives our conclusions.

2 Background

In this section we describe other work in the field and indicate its relationship to our method. We also give an overview of *Synergistic Image Creator*, which is the basic scheme underlying *Algorithmic Painter*.

2.1 Related Works

Most NPR research has been focused on artistic media simulation, which provides tools for virtual artists – oil paint, watercolor, brush, paper, canvas, and so on. Examples include methods for creating pen-and-ink images [13, 14, 15, 18], oil paintings [1, 11], water paintings [2], colored pencil drawings [17], and oriental water ink paintings [19]. Some methods have been proposed for creating images automatically according to a specific artistic form. Examples include: technical illustrations [4], oil paintings [5, 20], painterly rendering [16], and hatching smooth surfaces [6]. Gooch [20] and Hertzmann [5] showed techniques for making paintings by means of different-sized brushstrokes. With these two methods artworks are basically obtained by tracing the source photo, so these methods can not synergistically rearrange the shapes of brushstrokes according to adjacent ones; the ability to modify brushstrokes according to their surroundings is important for creating a rhythm or a stream of strokes. Hertzmann's algorithms basically use solid-shaped brushstrokes, and for Gooch's method a changeable element is only the width of one brushstroke. Therefore they give us artificial touches and it is hard to feel a natural rhythm. However, the rhythm of brushstrokes is an important element to make us feel comfortable. To create a rhythm in painting, periodically placed brushstrokes are effective. As shown by these examples, in the past basically one special technique has been totally devised to create each new expression. Therefore, the expression of each new style of painting requires a lot of time and energy in order to create yet another algorithm. Our method, on the other hand, codes an image, and applies the algorithms of various expressions to those codes in order to create an artistic picture. Consequently, the user can easily customize an algorithm for various types of expression within our framework.

Hertzmann et al. have reported a framework for image processing by example, which generalizes texture synthesis for the case of two corresponding image pairs [7]. However, this method only treats the features at pixel level, which makes it difficult to express painterly characteristics, such as artistic touches, in large homogenous areas and in highly contrastive areas. With *Algorithmic Painter*, these areas are extracted from a source image, and expressed in appropriate painting styles.

The painter Harold Cohen has developed a computer program called Aaron, which autonomously decides the image contents and draws them [12]. Aaron represents Cohen's knowledge and style of drawing; each drawing produced by the program is unique, but the style of drawing is fixed.

DeCarlo and Santella have devised an abstracted image stylization method based on human processing of visual information [3]. Their method represents images as hierarchical structures, relating meaningful image contents across image scales. Data obtained from an eye-tracker is used to predict the important parts of the image. The output image is then abstracted by pruning the structure with the obtained prediction data. *Algorithmic Painter* also represents an image as a hierarchical structure. But while the contents of the hierarchical structure in DeCarlo’s method are mainly used for abstracting, *Algorithmic Painter* uses the hierarchical structure to find candidate areas to which it can add distinctive expressions, and allows the CG creator to choose appropriate areas to add expressions. As for the algorithm, merged areas are created by connecting adjacent similar color segments like a chain, so that areas with even gradations are merged appropriately into one area. With our method, the appearance of the image can be widely varied by changing the characteristics of the brushstrokes in that area.

2.2 Overview of *Synergistic Image Creator*

We had been studying *Synergistic Image Creator* because it takes the painting process into consideration [8]. We adopted *Synergistic Image Creator* as the framework for *Algorithmic Painter* because *Synergistic Image Creator* codes the whole source image into vector style data before creating the artwork. The coded image data are useful for a visual design analysis of the image structure and can create any size of artwork, free of the source photo. The size of an artwork is very important for the artist, therefore a method in which the artwork size depends on source photo would not be suitable for our purpose.

Synergistic Image Creator consists of three main procedures.

(1) **Image segmentation**: This process divides the source image into segments, using color and edge information from the source image.

(2) **Vectorization**: Each segment's shape, size, position and color is calculated and defined as vector data.

(3) **Brush Stroke Creation and Rendering Process**: Each vectorized segment is transferred into a brushstroke. An image is rendered by painting all pixels in brushstrokes in an appropriate color.

Figure 1 shows the result of *Synergistic Image Creator*. Figure 1(a) is the source photo, and Figure 1(b) is the generated image obtained by varying the color of the segments by 5 %. The obtained brushstrokes hold the features of the source image in detail. Therefore, an observer viewing the resultant painting from a distance sees it as a real photo, not a painting. There is no remarkable difference between the source and the generated images as long as the colors of segments are not widely varied. On the other hand, if you need an artwork which is conspicuously different from the source image, you must drastically change the shapes and colors of the brushstrokes. Consequently, many human facial characteristics may be lost.

To solve these problems *Synergistic Image Creator* has, we need a system to analyze the context of source image to determine what kind of process should be assigned to each image segment.

Our new method offers a solution to these problems by obtaining and classifying the structure of the source image, and creating various brushstrokes with that data.

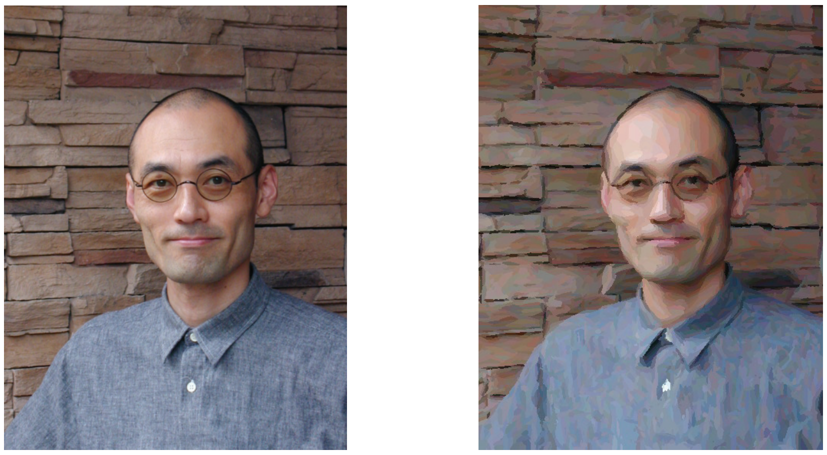

(a) Source Image (b) Generated Image with *Synergistic Image Creator*

Figure 1. Example of original *Synergistic Image Creator*

2.3 Objectives

Our final goal is to develop a system that paints an artwork from a captured image, the same way a picture might be painted from an image projected on the retina. The main objectives of our method are as follows.

1. To support more varied styles of artistic expression than *Synergistic Image Creator* by adopting curvy brush strokes.
2. To create expressive and attractive images without losing the essential characteristics of the source image
3. To allow creators to easily design new expressions by changing several lines of programs, as with *Synergistic Image Creator*

Since almost conventional image expression tools require a new algorithm for every style of painting, considerable effort is needed to create a tool for each new style. *Synergistic Image Creator* codes an image, and applies the algorithms of various expressions to those codes to create a target picture, enabling the user to easily customize an algorithm for various types of expression within our framework. We needed to include these characteristics in *Algorithmic Painter.*

A portrait is one of the most difficult subjects, because of the necessity to retain recognizable facial features. To make the advantages of our method clear, we chose a portrait as our subject, and ascertained *Algorithmic Painter*'s potential by examining how well it expresses the diverse painting styles of several artists.

3 Algorithm

This section gives a process overview and describes the algorithm we devised to generate artistic paintings from images.

3.1 Process Overview

Algorithmic Painter processes images in L*a*b* color space, and consists of eight steps, also shown in Figure 2:

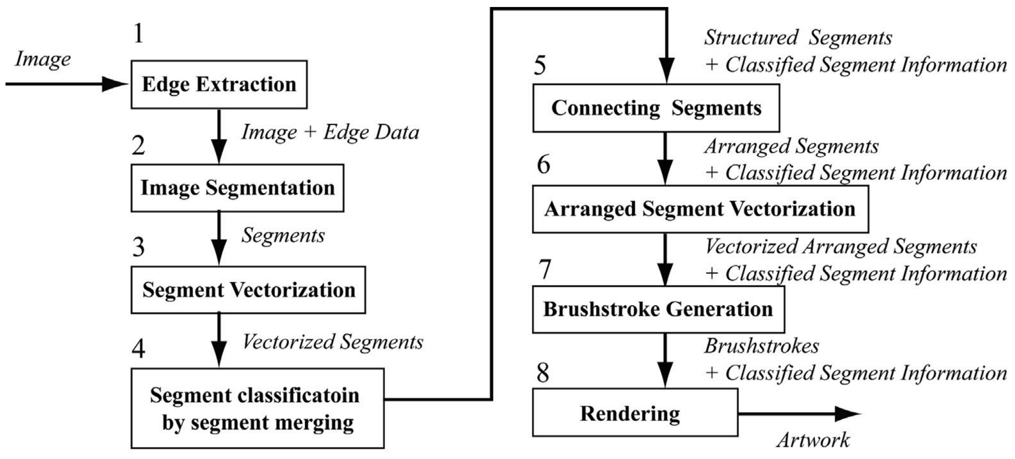


Image
1
Edge Extraction
Image + Edge Data
2
Image Segmentation
Segments
3
Segment Vectorization
Vectorized Segments
4
Segment classificatoin by segment merging
Structured Segments + Classified Segment Information
5
Connecting Segments
Arranged Segments + Classified Segment Information
6
Arranged Segment Vectorization
Vectorized Arranged Segments + Classified Segment Information
7
Brushstroke Generation
Brushstrokes + Classified Segment Information
8
Rendering
Artwork

Figure 2. Process Overview: The italic text shows the flow of data.

The generated brushstrokes consist of seventeen elements, as given in Table 1. Here, D_j and F_j which mean the direction and thinness of the segment j, are important to give each brushstroke a natural and different appearance in the Brushstroke Generation Process. They are given by Equation(1) and (2), respectively.

$$F_j = (\lambda_{1j} - \lambda_{2j}) / (\lambda_{1j} + \lambda_{2j}) \qquad (1)$$

$$D_j = \tan^{-1}(-\omega_{2j} / \omega_{1j}) / \pi + 1/2 \qquad (2)$$

where,

ω_{1j}: weighting value for primary component

ω_{2j}: weighting value for secondary component

3.2 Image segmentation

Prior to the image segmentation process, the continuities of pixels in source photo are extracted [9]. The source image is segmented using color and edge data. To reduce the long processing time, a divided K-means Method [9] was adopted as the segmentation algorithm. In this process, an edge area is also represented by grouping corresponding narrow segments.

3.3 Segment vectorization

The original *Synergistic Image Creator* calculates the characteristics of segments. These component characteristics are used to create brushstrokes [8]. In addition to these characteristics, the curvature of the segment is newly defined with the *Algorithmic Painter* method, which are listed in Table 1.

Serial number (ID)	: j
Color elements	: L^*_j, a^*_j, b^*_j
Number of pixels included in segment j	: N_j
Centroid of segment j	: X_j, Y_j
Deviations of primary and secondary component of segment j	: λ_{1j}, λ_{2j}
Direction and Thinness of segment j	: D_j, F_j
The centroid of part A, B, C	: (x_{aj},y_{aj}), (x_{bj},y_{bj}), (x_{cj},y_{cj})
The number of pixels in part A, B, C	: n_{aj}, n_{bj}, n_{cj}

Table 1. Data structure of j-th brushstroke by *Algorithmic Painter*

Figure 3 shows the result of image segmentation; the red segments found around the subject's ear and glasses are curved areas. Most of the brushstrokes are long and narrow. These curved segments are not conspicuous, but they are important for artistic expression.

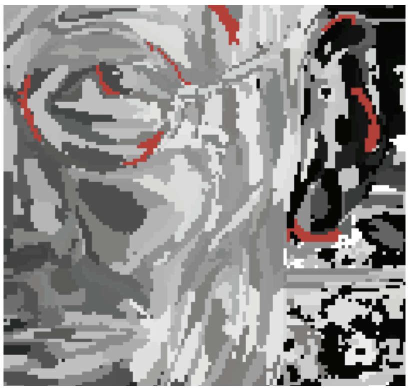


Figure 3. Curved segments

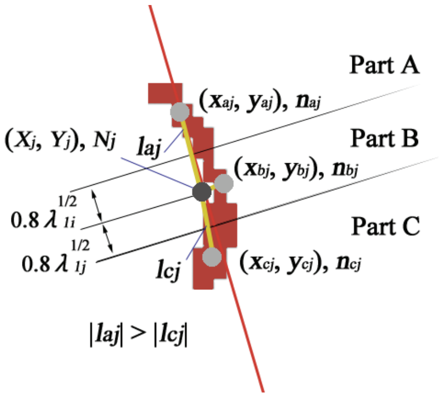


Figure 4. Three parts of an image segment

The curvature of the segment is calculated using the results of the principal component analysis for that segment. Figure 4 illustrates the magnified segment. The red line shows the direction of the segment's primary component. The two black lines are perpendicular to the red line, and intersect at the points distant from the centroid of the segment by $0.8\lambda_{1j}^{1/2}$. The three parts divided by these two black lines are named Part A, Part B, and Part C, respectively. Here, we name Part A so as to satisfy the condition $l_a > l_b$. The centroid of each part, (x_{aj},y_{aj}), (x_{bj},y_{bj}), and (x_{cj},y_{cj}), and the number of pixels in each part, n_{aj}, n_{bj}, and n_{cj} are then calculated. These data are used in the rendering process.

3.4 Segment classification by segment merging

It is necessary to determine what kind of process should be assigned to each segment. For this purpose, all segments are classified by merging the segments obtained through the image segmentation process. This classified segment information is required to create eloquent and natural regions for painting. An artwork is generated by applying a drawing algorithm for each merged segment. Therefore, the way the segments are merged for classification affects the style of painting.

3.4.1 A method of segment merging

One of the main objectives of the proposed method is to create artistic portraits. Creating an appropriate portrait requires not only retaining the facial features but also expressing a painter's style. Accordingly, a merged image segment is classified into the following three types:

type 1: Contours and feature lines

type 2: Large and homogeneous areas

type 3: Highly contrastive areas

Here, *type 1* and *type 3* segments are important for representing human's facial features; *type 2* segments are mainly used to express a style of painting.

type 1: Contours and feature lines

A narrow segment, whose thinness F_j is greater than 0.95, is not merged and remains as it is.

type 2: Large and homogeneous areas

First, neighbors for each image segment are listed. Next, a segment (dark green region in Figure 5 is identified, and the color differences, *ΔC*, between that region and its neighbors are calculated. The segment whose *ΔC* is within the color threshold value, *Tc*, is marked. After that, each marked segment (light green region in Figure 5 is identified, and the same process is performed again.

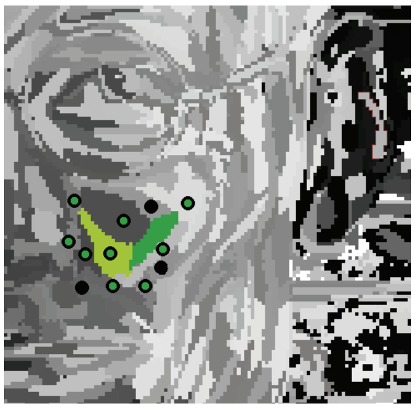


Figure 5. Segment merging: The regions marked with green dots are the segments whose color difference is within *Tc*, and the segments marked with black dots are not.

This procedure is repeated recursively for each segment until there is no region left to be marked, and all the marked segments are merged into one.

This procedure is performed for *n*-th segment merging, by setting the threshold value to *nTc*. The greater the color threshold value, the larger the merged segment.

A merged segment is fixed and classified as a *type 2* area, if the following two conditions are satisfied: where $N_{j,n}$ is the number of segments to be merged into *j*-

th merged segment in the *n*-th merging process, and N_{all} is the number of all segments.

$$\begin{cases} (N_{j,n} - N_{j,n-1}) / N_{j,n-1} < 0.1 & \text{(c1)} \\ N_{j,n} / N_{all} > 1/(60n) & \text{(c2)} \end{cases}$$

The condition (c1) says that the difference between the number of segments in merged segments j and the number of segments in merged segments j in the previous iteration less than 10 % . This condition controls a termination condition for segment merging, therefore stable segments merging can be performed for various condition of the photograph.

In general, the merged segment that has a large area and that was merged under a lower *Tc* value is most valuable in creating an artistic painting. To let a user control this value, the value of *Tc* is defined by Equation (3).

$$Tc = (C_{max} - C_{min}) / Ta \qquad (3)$$

where,

C_{max} : Color of pixel having the highest intensity in all segments

C_{min} : Color of pixel having the lowest intensity in all segments

Ta : User defined constant (default: 80)

type 3: Highly contrastive areas

The highly contrastive area is conspicuous if the number of segments to be merged is small. The merged segment is registered as *type 3* segment if the following conditions are satisfied.

(a) The number of segments to be merged is less than 5, and will not be changed.

(b) The color difference with neighbors is greater than $10Tc$

Figure 6 visualizes an example of classified segments: the black segments are edges (*type 1*), white ones are highly contrastive areas (*type 3*), and those colored in another unique color, such as light blue and dark red, are homogeneous areas (*type 2*). This classified segment information will be used in the following processes.

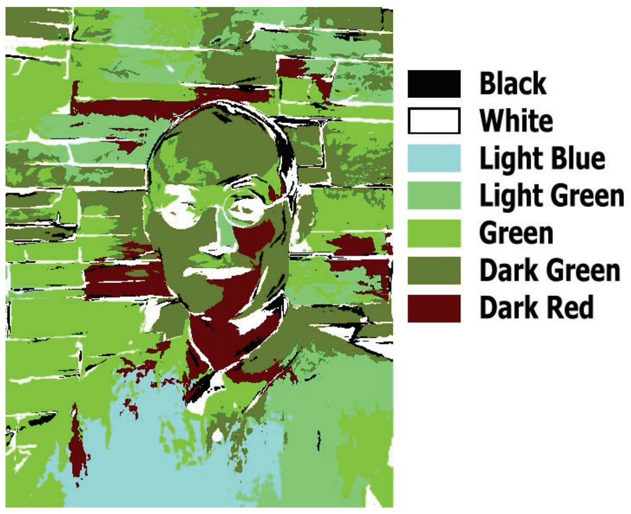
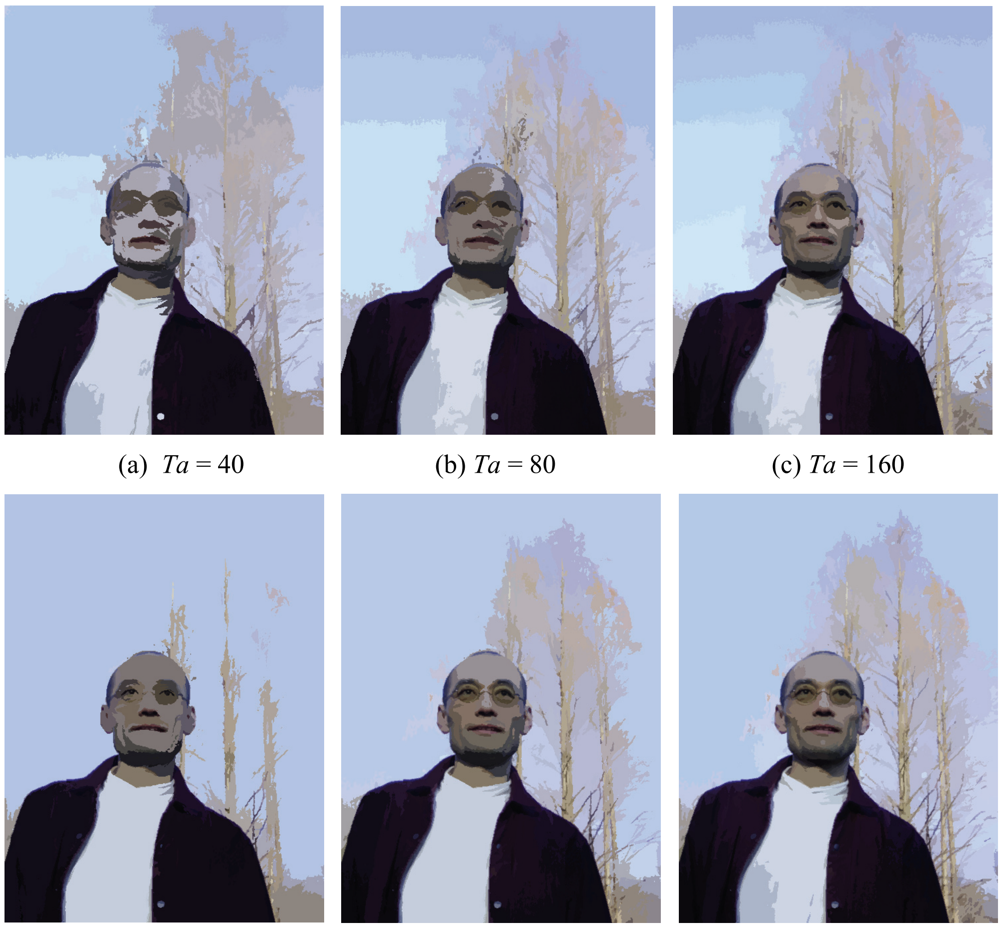


Figure 6. Classified image segments of Figure 1(a): Light blue areas are classified into *type 2* at the first iteration; light green, green, dark green, and dark red areas are classified at the 2nd, 3rd, 4th, and 5th iterations, respectively. Black areas are classified as *type 1* and white areas are *type 3*.

3.4.2 Results of segment merging

In the process of segment merging, condition c1 is important for acquiring an appropriately merged segment. In general, it is hard to merge gradational areas into one. However, some gradational areas, such as a blue sky, are better classified into almost one area because sky is usually painted with the same touch. From now on, condition c1 will be called the adaptive condition. Figure 7 shows the effect of the adaptive condition. The figures in the upper row, Figure 7(a), (b), and (c), show results satisfying only condition c2, where *Ta* is the control parameter for merging in Equation (3). The figures in the lower row, (d), (e), and (f), show results satisfying both conditions c1 and c2.

In Figure 7(a), (b), and (c), the areas in the sky are merged incorrectly, but in (d), (e), and (f), the segments in the blue sky are merged into one. This is because condition c1 does not terminate the merging process while there are segments that should be merged. On the other hand, target segments in (a), (b), and (c) are forced to merge with an incorrect region whose color does not match for that, under only condition c2. This fault also creates inappropriately merged segments in the facial area in (a), (b), and (c).



(a) *Ta* = 40 (b) *Ta* = 80 (c) *Ta* = 160

(d) *Ta* = 80 (e) *Ta* = 160 (f) *Ta* = 240

Figure 7. Results of segment merging: All images are created by painting every segment with its average color.

3.5 Connecting Segments

In general, if the brushstrokes are coarse, a painting does not have detailed features but does have painterly characteristics; if the brushstrokes are fine, the painting does not have painterly characteristics, but does have details. Accordingly, a painter chooses an appropriate brush size depending on the part of the painting.

This section proposes a process to choose an appropriate brushstroke. It is called the connecting process, which merges segments adaptively from fine segments to coarse ones.

The class attribute of an image segment is used to hold the facial features in a painting. *Type 1* and *type 3* segments should not be connected because facial features need to be drawn in fine brushstrokes. The following connecting process is only applied to *type 2* segments.

Figure 8(a) shows the relationship between Segments *M* and *T*, In this process, Segment *M* is the current segment, and Segment *T* is the target segment which is to be connected to Segment *M*. W_m and W_t are the centroids of Segments *M* and *T*. L_m is the straight line in the direction of Segment *M*'s primary component, which passes along W_m. L_t is the straight line in the direction of Segment *T*'s primary component, which passes along W_t. The angle φ is measured counterclockwise between L_m and L_t.

The connecting process selects the neighboring segments that satisfy the following conditions:

(i) The angle φ is positive, if W_t is located in the left side of L_m.
(ii) The angle φ is negative, if W_t is located in the right side of L_m.

Thus, in the case of Figure 8(b), Segment *a* is merged, but Segment *b* is not.

Next, the process chooses the nearest segment among those selected, and connects them. This connecting process is restricted from connecting more than two segments. All connected segments consist of two segments and these are used for expressing several different painterly characteristics.

Finally, the connected segments are vectorized, using the procedure described in Section 4.3.

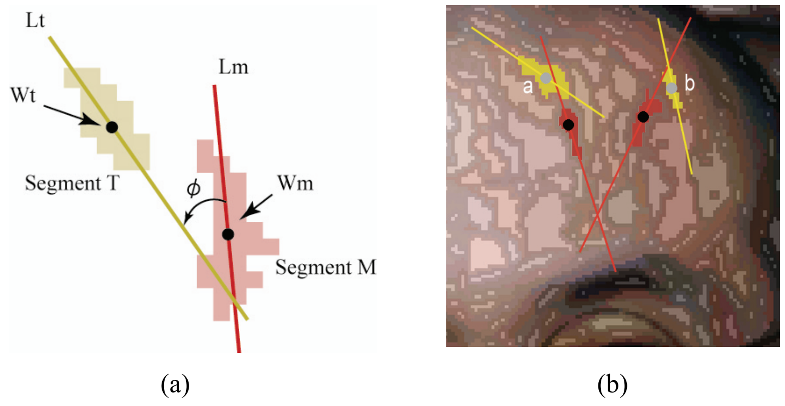


Figure 8. Selection of connecting target: (a) Relationship of Segment *M* and Segment *T*, (b) An example

3.6 Brushstroke generation

Brushstrokes are created from the vectorized arranged segments and classified segment information. First, the method used to create brushstrokes in the original *Synergistic Image Creator* is described for reference. Basically, brushstrokes are created by deciding which pixel belongs to which segment, as shown in Figure 9(a).

The distance function affects the shape of the brushstroke. Euclidean distance is not suitable for making a natural brushstroke, because it defines circular strokes. It is necessary to define another distance function to make ellipsoidal brushstrokes. The way to create such brushstrokes is as follows;

When pixel *n* takes the position shown in Figure 9(a), to create a ellipsoidal brushstroke, pixel *n* should be included in brushstroke *j* instead of brushstroke *j'*, which is nearer pixel n in Euclidean space. To realize this, new distance $L_{j,n}$ is given by Equation (4). Consequently, the shape of brushstroke *j* will be stretched along the long axis because $L_{j,n} < L_{j',n}$. The distance $L_{j,n}$ is given by Equation (4) to make a natural brushstroke.

$$\left.\begin{array}{l}
D_{j,n} = |(\tan^{-1}(x_{j,n} / y_{j,n})) / \pi - D_{j,n}| \\
\textit{if } D_{j,n} \geq 0.5 \text{ then } D_{j,n} = 1 - D_{j,n} \\
x_{j,n} = (X_n - X_j) D_{j,n} F_j / N_j^{1/2} \\
y_{j,n} = (Y_n - Y_j) D_{j,n} F_j / N_j^{1/2} \\
L_{j,n}^2 = x_{j,n}^2 + y_{j,n}^2
\end{array}\right\} \quad (4)$$

where,

(X_n, Y_n): Coordinate of current pixel n

(X_j, Y_j) : Centroid of segment j

Modifying Equation (4) changes the shape of the brushstrokes. Suppose that every segment is an ellipse. Note that this assumption does not mean that actually created brushstrokes will be ellipses or symmetrical shapes [8]. The brushstrokes created by these steps are called legacy brushstrokes.

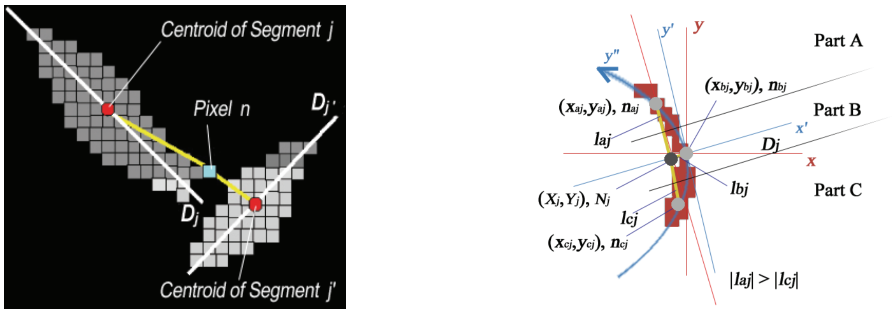


Figure 9. Brushstroke creation: (a) The current processing pixel *n*, belongs to the brushstroke that has the shortest distance function in the brushstroke creation process. (b) Notation of curved segment.

Every segment is curved to some extent. Accordingly, defining the brushstroke with curvature parameters makes it more natural. A curly brushstroke is created by introducing curved coordinates. In this method, only the *y*-coordinate is deformed by a quadratic curve, as shown in Figure 9(b). It is not necessary to use a rectangular coordinate system, because we only need to define the distance from the origin of each arranged segment to the current pixel.

In the case of a legacy brushstroke, the origin of the coordinate is the centroid of the segment. On the other hand, in the case of a curly brushstroke, it is reasonable that the centroid of Part B is chosen as the origin of the coordinate system. Here, we also name Part A so as to satisfy the condition $l_a > l_b$. Accordingly, the center of a curved arranged segment is shifted from the centroid of the whole segment to the centroid of Part B. Then, the axis of the coordinate is rotated by D_j. The new coordinate (x_j', y_j') is given by Equations (5) and (6).

$$x_j' = (x - x_{bj})\cos D_j + (y - y_{bj})\sin D_j \tag{5}$$

$$y_j' = -(x - x_{bj})\sin D_j + (y - y_{bj})\cos D_j \tag{6}$$

The final coordinate (x_i'', y_i'') is given by Equations (7) and (8).

$$x_j'' = x_j' + A_j y_j'^2 \tag{7}$$

$$y_j'' = y_j' \tag{8}$$

where,

$$A_j = A_{user}(l_{bj} / l_{aj}) \ \ when\ 0 \le D_j < \pi, x_{aj}\, y_{bj} \times x_{bj}\, y_{aj} > 0$$
$$A_j = - A_{user}(l_{bj} / l_{aj}) \ \ when\ 0 \le D_j < \pi, x_{aj}\, y_{bj} \times x_{bj}\, y_{aj} \le 0$$
$$A_j = A_{user}(l_{cj} / l_{aj}) \ \ when\ \pi < D_j \le 2\pi, x_{aj}\, y_{bj} \times x_{bj}\, y_{aj} < 0$$
$$A_j = - A_{user} (l_{cj} / l_{aj}) \ \ when\ \pi < D_j \le 2\pi, x_{aj}\, y_{bj} \times x_{bj}\, y_{aj} \ge 0$$

A_{user} is a positive constant controlling the curvature of the brushstroke. The generated artwork is faithful to the source image when A_{user} is set to 1. A variety of expressions can be given to the artwork by controlling this constant. The distance $L_{j,n}''$ is given by Equation (9).

$$L_{j,n}'' = (x_j''^2 + y_j''^2)^{1/2} \qquad (9)$$

A final curved brushstroke is created by choosing the region that has the minimum $L_{j,n}''$ for all pixels.
The final image is rendered by establishing the brushstroke color for each pixel. The tone of the painting is varied by changing colors for each created brushstroke.

4 Results

This section shows various examples that highlight the advantages of this proposed NPR method. The merged segments for segment classification, unconnected segments, and connected segments are all vectorized, and used to draw a picture by applying individual algorithms for each segment.

4.1 Optimization of segment merging

It is important to set appropriate parameters for the process of segment merging, because the quality of the output image as an artwork depends on those parameters.

Figure 10 shows the effect of changing *Ta,* which controls the number of segments. All figures in Figure 10 are drawn by using unconnected segments for the white and black areas shown in Figure 6 and merged segments for the others. In Figure 10(a) the important facial parts such as eyes, are not drawn satisfactorily. In Figure 10(b), those parts are well drawn. However, the facial contours in Figure 10(b) are drawn less well than in Figure 10(c), because the number of segments in (b) is smaller than in (c). Therefore, the best value for *Ta* is 150 among these examples. We have checked the effect of changing *Ta* in other images, and found that the best value for it is almost always 150.

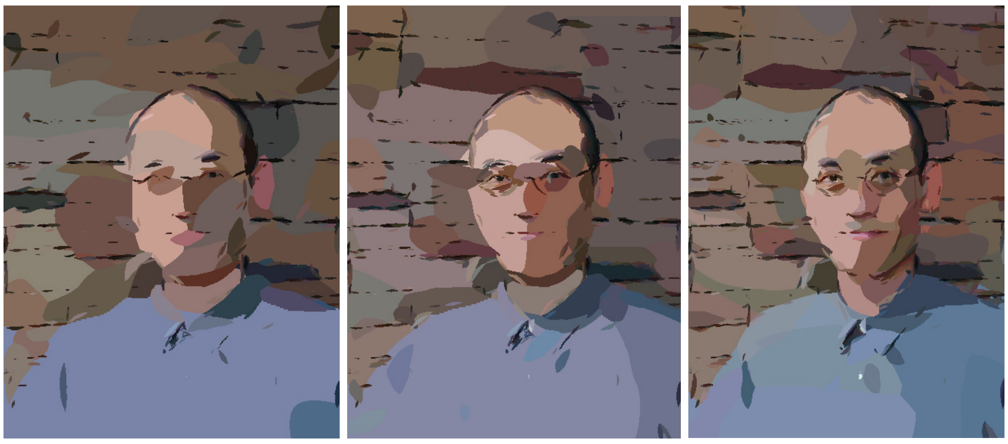

(a) *Ta* = 80 (b) *Ta* = 110 (c) *Ta* = 150

Figure 10. Optimization of image region merging

The level of detail in image segmentation does not completely depend on *Ta*. For example, the level of detail around the neck in Figure 10(b) is higher than that in Figure 10(c).

In Figure 10(c), the important details are well drawn, and the large areas are roughly rendered deviating from their original contours. This style of painting is suitable for some kinds of portraits.

The parameter A_{user} is used to intentionally bend a brushstroke. In this example it is not desirable to bend the brushstrokes very strongly, but proper bending is necessary to generate natural brushstrokes. For all the examples in Figure 10 A_{user} is set to 1, which is the default value. The effect of changing A_{user} is discussed further in Section 5.4.

4.2 Controlling the level of detail

Figure 11 shows the effect of changing the level of detail of the picture. Here, all edges and the regions that have high contrast are drawn with small brushstrokes.

In the same way as in Figure 10, Figure 11(a) is generated by applying connected segments for the dark red region shown in Figure 6, and by applying merged segments for the regions that are homogeneous and have a larger area than the dark-red regions. Figure 11(b) is obtained by applying merged segments for the light blue region shown in Figure 6, and applying connected segments for the remaining homogeneous region. Figure 11(c) is produced by applying connected segments, to all homogeneous regions. This example shows the effectiveness of segment connection. Figure 11(d) is the same image as Figure 1(b) for convenience of comparison. Figure 11 (c) ought to look more like a painting than Figure 11 (d). These images also demonstrate the improvements over our previous systems, *Synergistic Image Creator.*

These examples also show another advantage of this method; The image is created from vectorized brushstrokes. Therefore, even though few arranged segments are used for creation of the brushstrokes, the final artworks are rendered without any remaining not-painted-areas.

4.3 Computational time

Computational time depends on the number of segments and arranged segments. For example, Figure 11(c), which has 449×581 pixels, took 393 seconds for image segmentation, 415 seconds for segment merging, 432 seconds for segment connection, and 595 seconds for rendering on a Mobile Intel Pentium III 866MHz processor.

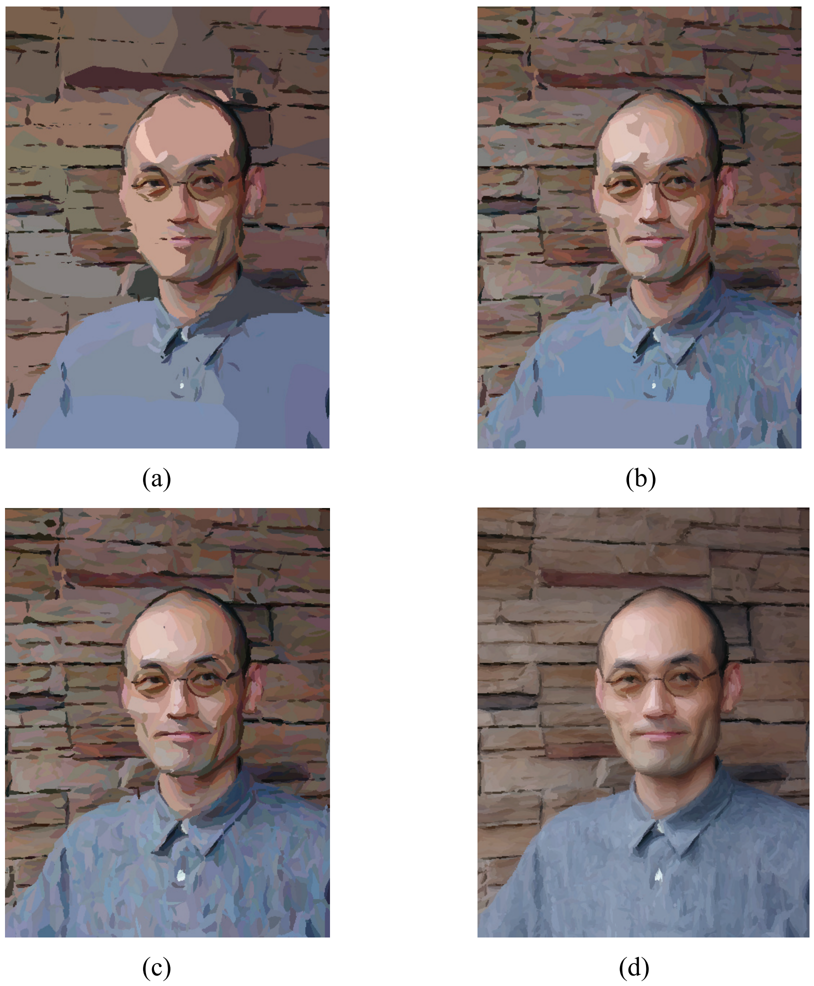


Figure 11. Controlling the level of detail: (a) ,(b) and (c) are created with Ta = 150, A_{user}=8 and (d) is the same image as Figure 1(b).

4.4 Application - various styles of painting

This subsection shows how *Algorithmic Painter* can be applied to the distinctive painting styles inspired by the brushstrokes of Van Gogh, Renoir, and Dufy. Before expression of the artwork, image analysis is executed. Figure 12(a) is a source image, (b) is a segmented image, and (c) shows classified segment information that is the result of analysis. By referring to this classified segment information, various types of expressions can be created.

The following three examples show that our framework broadens the gamut of rendering styles.

4.4.1 Painting style inspired by Van Gogh

In Van Gogh's later years his style of painting changed, with swirling brushstrokes in the background. The curved brushstrokes along the contour of portrait in "*Garden of Saint-Paul Hospital*" and "*Portrait of the Art Dealer Alexander Reid*" are examples of this style.

To express the characteristics of these kinds of brushstrokes, the strokes are transformed by adjusting their directions and thinness to the values of the conspicuous brushstroke, the one that has maximum thinness. This transformation process conveys the curvature of segments from one to another; therefore the process is iterated 100 times, by dropping any segment, with an F_j greater than 0.9 from the list of those to be processed.

In the above process, only segments which are highly homogeneous, as indicated by the greenish and dark red areas in Figure 12(c) are connected and processed; the parameter A_{user} is set to 1.6 for creating curly brushstrokes. Figure 12(d) shows the shapes of the brushstrokes.

To prevent a monotonous touch in an artwork, three types of rendering are applied, as follows;

(1) Pixel brightness is raised gradually as we move toward the boundary of the region, when $0.93 < F_j$.
(2) The upper part of the region is highlighted and the lower part is shaded, when $0.85 < F_j < 0.93$.
(3) The ordinal rendering process is applied for other cases.

The final artwork is shown in Figure 12(f).

Figure 13 shows the comparison between Gogh-like brushstrokes created by the *Synergistic Image Creator*, and those created by the proposed modifications to *Synergistic Image Creator, Algorithmic Painter.* Figure 13 (a) shows that *Synergistic Image Creator* can not create large segments by combining adjacent segments nor create expressive curvy brushstrokes, therefore it also shows too regular arrangements and too uniform-sized brushstrokes.

4.4.2 Painting style inspired by Renoir

There are two levels of detail in Renoir's work, such as "*On the Terrace*"; the important parts are drawn in detail, and the others are drawn roughly.

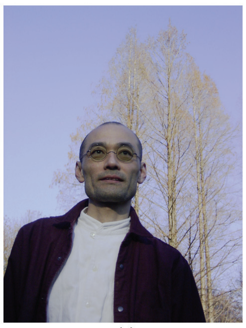
To express the characteristics of this style of painting, the roughly painted areas are rendered with brushstrokes that are obtained by placing duplicated brushstrokes in the direction of the region's secondary principal component.

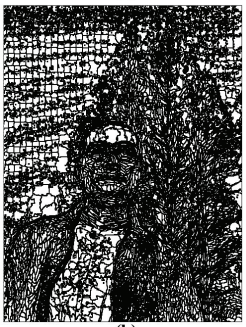
In this example, except for the white and black segments shown in Figure 12(c), almost all of the regions are drawn with duplicated brushstrokes created with connected segments; however, the segments shown by light green in Figure 12(c), such as the sky and the white shirt, are painted with round brushes created with unconnected single segments. The black and white segments in Figure 12(c) are painted with unconnected single strokes. The created brushstrokes are shown in Figure 14(a). To emphasize blurred brushstrokes, a color is faded into white according to the length of the stroke. Figure 14(b) is the final artwork.

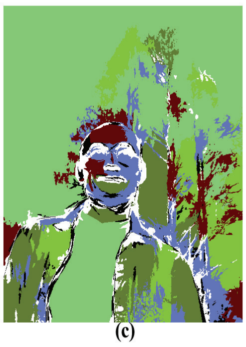
4.4.3 Painting style inspired by Dufy

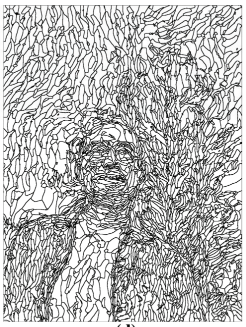
In Dufy's style, such as "*Le Paddock*", contours of objects are expressed with curved lines, and are freely painted with light colors. Figure 10(c) shows Dufy's characteristics to some extent, therefore, brushstrokes are created by the same method used in Figure 10(c). In this example, except for the white and black segments shown in Figure 12(c), almost all of the regions are drawn with single large brushstrokes created with merged segments. The brushstrokes are shown in Figure 14(d). The right shoulder is not painted, and the blue sky area is too large to be drawn with only one brush stroke. To obtain a more finely merged segment, the merging process is performed without the adaptive condition, setting *Ta* to 160. Under this condition, the blue sky is not merged into one segment (as shown in Figure 14(e)), therefore, the right shoulder is clearly delineated (shown in Figure 14(f)).

In the rendering process, the center lines of segments are drawn with deep colors, and their neighbors are painted with light colors (Figure 14(g)).


(a)


(b)


(c)


(d)

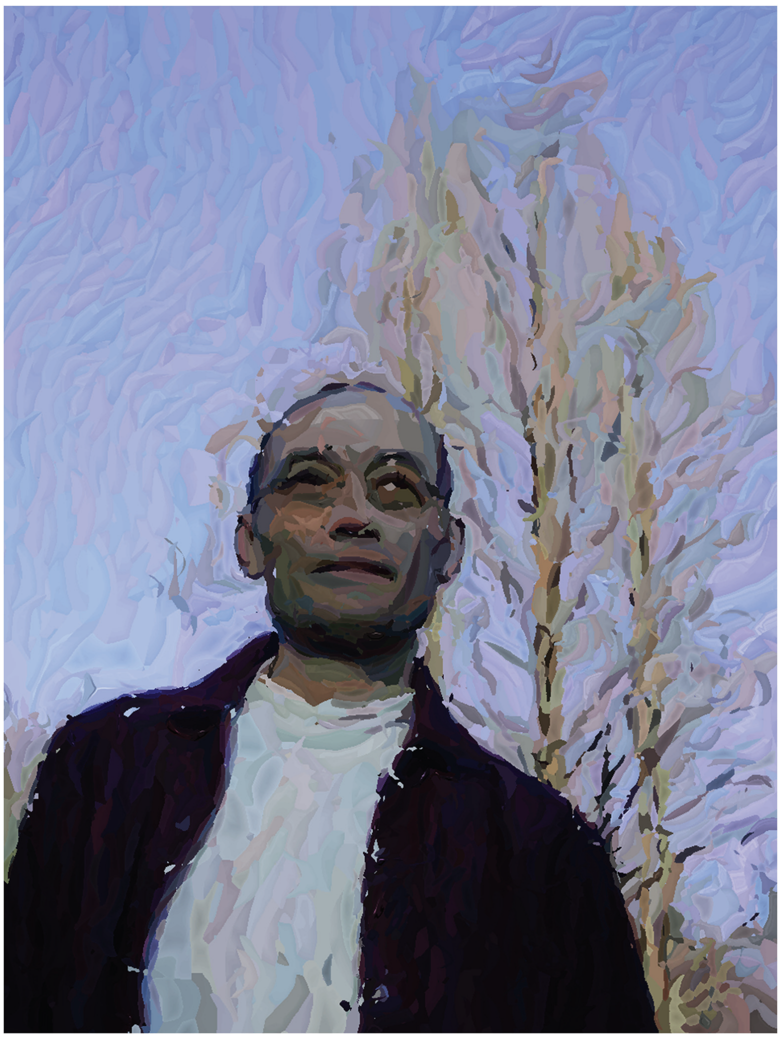
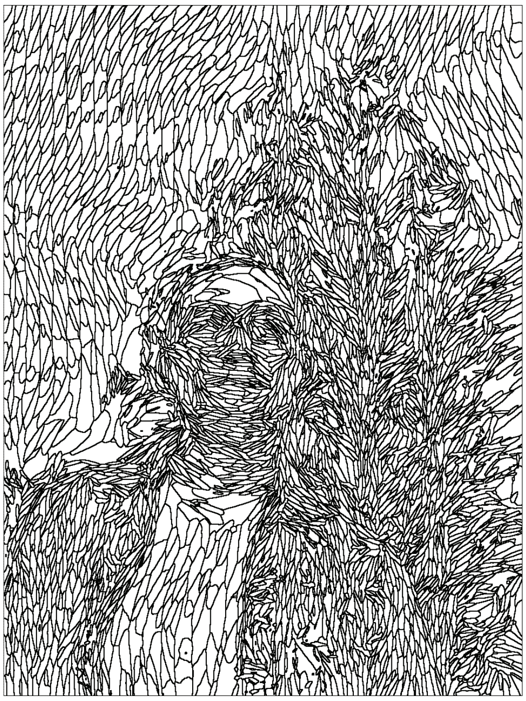

(e)

Figure 12. Painting style inspired by VanGogh:

(a) Source Image
(b) Segments
(c) Classified segment
(d) Brushstroke
(e) Painting style inspired by Van Gogh


(a) Brushstrokes by *Synergistic Image Creator*

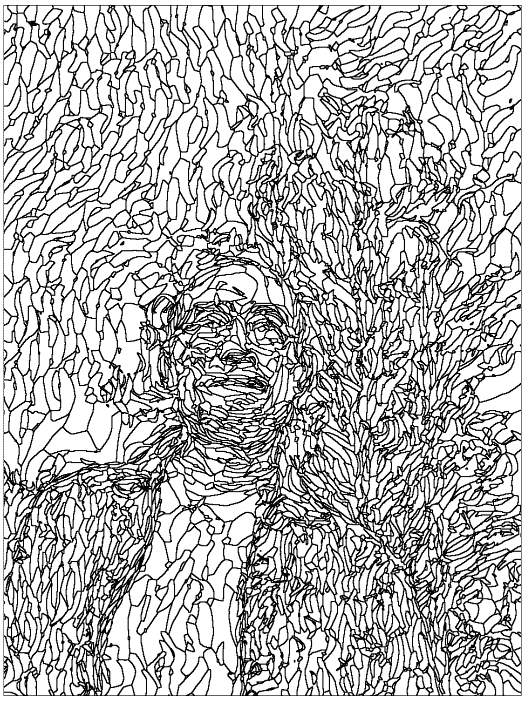

(b) Brushstrokes by *Algorithmic Painter*

Figure 13. Comparison of brushstrokes

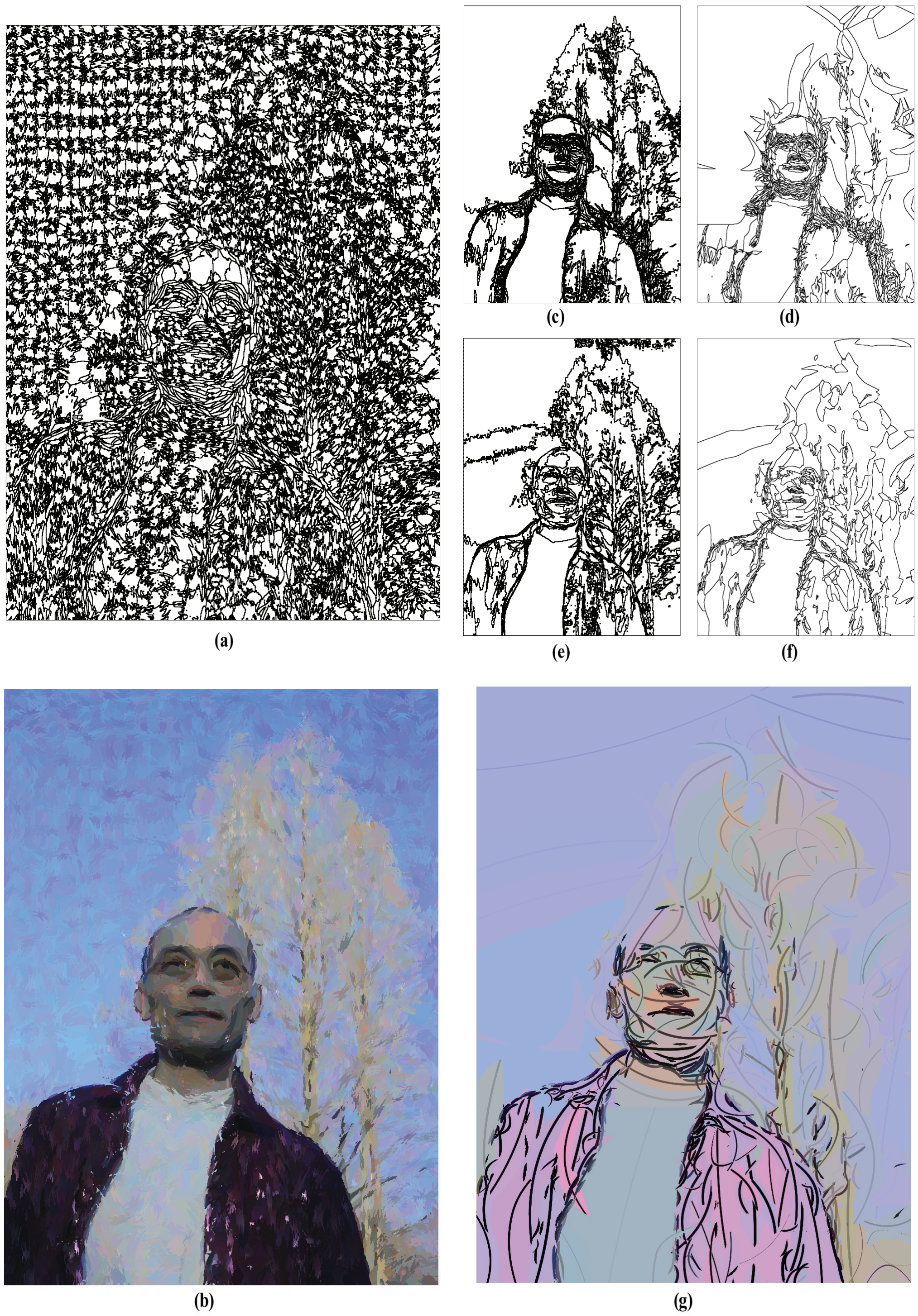


Figure 14. Painting styles inspired by Renoir and Dufy:

(a) Brushstrokes inspired by Renoir
(b) Painting style inspired by Renoir
(c) Merged segments created by adaptive condition
(d) Brushstrokes created with segments in (c)
(e) Merged segments created by adaptive condition
(f) Brushstrokes created with segments in (e)
(g) Painting style inspired by Dufy.

5 Conclusion

We have presented a method for creating an artistic painting from a source photo by enhancing *Synergistic Image Creator*. Because our method can classify the segments of a source photo into three types for the convenience of the CG creator, s/he can choose his/her favorite brushstrokes and apply them to the classified segments, to create an artwork which contains several types of brushstrokes. Moreover, this algorithm can produce large segments by combining adjacent segments and can produce expressive curvy brush strokes. The obtained painting holds the features of the original image, and can also be rendered in various styles. Within this framework a CG creator can easily design a desired painting style by customizing an algorithm to define the shape of the brushstrokes. For image segmentation, we might consider performing some kind of texture analysis to classify different parts of the image.

As a demonstration, we have created a well-expressed portrait in three varied painting styles inspired by Van Gogh, Renoir, and Dufy.

In the future, we would like to develop an interactive image expression tool for creating artworks, based on *Algorithmic Painter*. We would like to provide an environment that the user can study the relation between algorithm and artistic expression, and can be used to improve brash handling skills as a teaching tool.

References

1. Meier, B.J. (1996) Painterly rendering for animation, Proceedings of ACM SIGGRAPH 1996: 477-484

2. Curtis C.J, Anderson S.E, Seims J.E, Fleischery K.W, Salesin D.E (1997) Computer-generated Watercolor, Proceedings of ACM SIGGRAPH 1997: 421-430

3. DeCaRlo D, Santella A (2002) Stylization and Abstraction of Photographs, ACM Transaction on Graphics, 21(3): 769-776

4. Gooch A, et.al. (1998) A non-photorealistic lighting model for automatic technical illustration, Proceedings of ACM SIGGRAPH 1998: 447-452

5. Hertzmann A (1998) Painterly rendering with curved brush strokes of multiple sizes, Proceedings of ACM SIGGRAPH 1998: 453-460

6. Hertzmann A, Zorin D (2000) Illustrating smooth surfaces, Proceedings of ACM SIGGRAPH 2000: 517-526

7. Hertzmann A, et.al. (2001) Image analogies, Proc. of ACM SIGGRAPH 2001: 327-340

8. Kasao A, Nakajima M (1998) A Resolution Independent Nonrealistic Imaging System for Artistic Use, Proceedings of the International Conference on IEEE Multimedia Computing and Systems: 358-367

9. Kasao A, Nakajima M (1998) K-Means Algorithm Using Texture Directionality for Natural Image Segmentation, Proceedings of '98 International Workshop on Advanced Image Technology: 23-28

10. Kasao A, Nakajima M (1999) Synergistic image creator - A picture generation system with consideration of the painting process, Systems and Computers in Japan 30(10): 13-21

11. Litwinowicz P (1997) Processing Images and Video for an Impressionist Effect, Proceedings of ACM SIGGRAPH 1997: 407-414

12. McCorduck P (1990) AARON'S CODE: Meta-Art, Artificial Intelligence, and the Work of Halod Cohen, W.H.Freeman and Company

13. Salisbury M.P, Anderson, S.E, Barzel R, Salesin D.H (1994) Interactive pen-and-ink illustration, Proceedings of ACM SIGGRAPH 1994: 101-108

14. Salisbury M.P, Anderson C, Lischinski D, Salesin D.H (1996) Scale-Dependent Reproduction of Pen and Ink Illustrations, Proceedings of ACM SIGGRAPH 1996: 461-468

15. Salisbury M.P, Wong M.T, Hughes J.F, Salesin D.H (1997) Orientable Textures for Image-Based Pen-and-Ink Illustration, Proceedings of ACM SIGGRAPH 1997: 401-406

16. Shiraishi M, Yamaguchi Y (2000) An algorithm for automatic painterly rendering based on local source image approximation, Proceedings of the First International Symposium on Non-photorealistic Animation and Rendering: 53-58

17. Takagi S, Fujishiro I, Nakajima M (1999) Volumetric modeling of colored pencil drawing, Proceedings of Pacific Graphics '99: 250-258

18. Winkenbach G, Salesin D.H (1994) Rendering Parametric Surfaces in Pen and Ink, Proceedings of ACM SIGGRAPH 1994: 91-100

19. Zhang Q (1999) Simple Cellular-Automaton-Based Simulation of Ink Behavior and Its Application to Suibokuga-like 3D Rendering of Trees, The Journal of Visualization and Computer Animation 10: 27-37

20. Gooch, B., G. Coombe, and P. Shirley (2002). Artistic vision: Painterly rendering using computer vision techniques. In proc. NPAR 2002: 83-90 & 160

21. http://www.siggraph.org/artdesign/gallery/S02/onwall/kasao/1bigimage.html

22. http://www.siggraph.org/artdesign/gallery/S03/2d/0263.html

23. http://plaza.bunka.go.jp/museum/jury_15_art.html