An Algorithm For Automatic Painterly Rendering Based On Local Source Image Approximation

Michio Shiraishi Yasushi Yamaguchi
{shira,yama}@graco.c.u-tokyo.ac.jp

University of Tokyo

Abstract

This paper presents a new method for the automatic painterly rendering. This method synthesizes an impressive image with a hand-crafted look from a source image such as a photograph. This method generates rectangular brush strokes approximating the local regions of the source image with suitable locations, orientations, and sizes. These properties are calculated with the image moments of the color difference images, obtained by taking the differences between the local source images and the stroke colors. The method explicitly deals with not only intensity but also chromaticity of the source image. The resulting image is composited with smaller strokes at the details while its flat regions are painted with larger ones. The method is also able to control the density of strokes as well as their painting order based on their sizes. The density is controlled by a dithering method with space-filling curves. Painting process starts from the larger strokes and finishes with the finer ones. Because of this density control and the painting order, the final image preserves the details of the source image.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation; I.4.0 [Image Processing and Computer Vision]: General

Additional Keywords: Non-photorealistic Rendering, Painterly Rendering, Image Moment, Brush Stroke

1 Introduction

Non-photorealistic rendering has become one of the essential research topics in the computer graphics society. The wide range of targets for non-photorealistic rendering includes edge-enhanced images[13], pen-and-ink illustrations[14], cell animations[4], and even floral ornaments[17]. This paper focuses on the paintings such as oil-paintings. There are mainly two approaches to the painterly rendering: the physical simulation and the brush stroke composition. The former approach simulates the physical behavior of oil-color [1], water-color[5], etc. The latter generates images with a hand-crafted look by placing brush stroke textures on a canvas. This paper takes the latter approach, which is further discussed in section 2.

The contribution of our work is a new method for automatically placing, sizing, orienting and scheduling brush strokes in a conventional painterly rendering framework. The proposed method is based on the approximation of the local source image with a rectangular brush stroke. The merits of this method can be summarized as follows:

1. The stroke size varies over the canvas. It means that the details are represented by smaller strokes while the flat area in the source image is painted with larger ones. This diversity of stroke size enhances the richness of the resulting image. In order to allow this kind of diversity, the stroke density is controlled by the dithering method , so that smaller strokes are located more densely than larger ones.

2. The location, orientation, width, and length of each brush stroke are properly determined, so that the brush stroke approximates the local region of the source image. This property is essential to represent the details in the source image, because any irrelevant deviation of the stroke attributes may spoil the result.

3. The painting order of strokes can be determined by their sizes. The smaller strokes are painted later than the larger ones, so that the resulting image preserves the details of the source image.

This paper is organized by six sections including this introduction. In section 2, the painterly rendering algorithm in general is described and the previous works are discussed. Section 3 explains how the proposed method approximates the local source image with a stroke, using the image moments of the color difference image. The overall algorithm is described in section 4, and the results are shown in section 5. Finally, section 6 concludes the paper.

2 Painterly Rendering

The framework for stroke composition used in this paper is identical to Haeberli's one[8], although the interactivity is not exploited in our work. In this section, the automatic procedures for the painterly rendering is explained. Section 2.1 describes the overall process of the image generation. Section 2.2 explains the model of the brush stroke in detail. Section 2.3 reviews the previous works.

The method to be discussed takes a two-dimensional color image as its input and generates a painterly rendered two-dimensional color image as its output. In this sense, it can be regarded as an image processing method. Thus, it does not use any three-dimensional information, *e.g.*, depth information.

There are many commercial painting software packages [3, 15] with which the users may create painterly rendered images by placing the brush stroke textures. However, such interaction with the user is not of interest in this paper. The aim of this study is to create the painterly rendered image automatically using the minimum user-defined parameters.

2.1 Process

The process of the non-interactive painterly rendering consists of two steps, *i.e.*, preparation and composition, as shown in Figure 1.

Preparation The system first determines the distribution of strokes, their attributes, and painting order. This step determines the following aspects:

Stroke distribution A painterly rendered image is composited by many strokes. The total number of strokes and the distribution over the canvas should be determined.

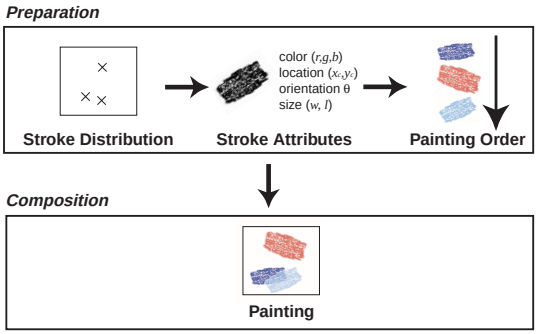


Figure 1: A process of painterly rendering.

Stroke attributes A brush stroke has some attributes such as color. For each stroke, its attributes should be determined to approximate the input source image. These attributes are further described in section 2.2.

Painting order The order of painting brush strokes also influences the final image. As the strokes are painted successively on the canvas, the strokes painted later may cover the strokes painted before. Therefore, the brush strokes should be sorted in order of painting.

Composition Strokes are painted on a white canvas one after the other. This process usually performed by the traditional alpha blending.

This paper mainly aims at determining stroke distribution, stroke attributes, and painting order.

2.2 Brush Stroke

There are many schemes to represent a brush stroke, such as a simple primitive like a rectangle[8, 11, 12] and a model using spline curves[9]. In this paper, the rectangular strokes are used for its simplicity. Each brush stroke has the following four attributes as shown in Figure 2:

Color The color attribute $\mathbf{C} \in \mathbf{R}^3$ is specified in the RGB color space.

Location The location attribute is represented by the coordinates (x_c, y_c) of the center of the stroke.

Orientation The orientation attribute is specified by the angle θ between the main axis of the stroke and the x-axis of the canvas.

Size The size attribute is specified by two dimensions, the width w and length l of the stroke.

The brush stroke texture is also influential in achieving the hand-crafted look. The texture image used in this paper (Figure 3(a)) is taken from Meier's paper[12]. Our method is applicable to an arbitrary texture image which is shaped like a rectangle or an ellipse.

Each brush stroke is processed and painted as shown in Figure 3:

1. scale the texture image to fit the width w and length l of the brush stroke (Figure 3(b)),
2. rotate the scaled image by θ (Figure 3(c)),
3. translate the scaled and rotated image by (x_c, y_c) (Figure 3(d)),
4. paint the stroke with the color **C** by the alpha blending (Figure 3(e)).

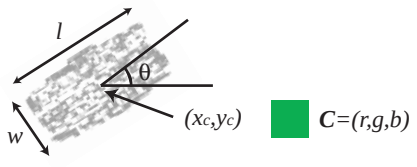


Figure 2: Stroke attributes.

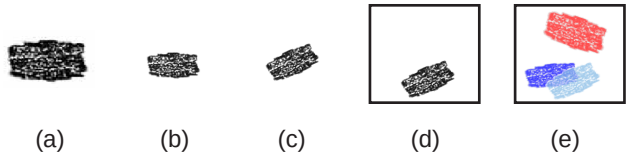


Figure 3: A composition process of a stroke.

2.3 Previous Works

Haeberli[8] developed an interactive system, which enables a user to place strokes on a canvas. The stroke distribution is given with an input device such as a mouse. A stroke color is chosen from the colors in the corresponding region of the source image. A stroke orientation as well as size can be controlled by the user. In addition to that, he proposed a method to control the stroke orientation automatically using the intensity gradient of the source image. The method orients the stroke vertically to the intensity gradient at the location.

Meier[12] introduced a method with a 3D model input. The information of 3D models, such as surface normals, yields to determine their orientations and sizes. Painting order is also favorably defined by the depth information. However, these techniques are not applicable to handle the 2D image input.

The method by Litwinowicz[11] takes a video sequence as its input. This work concentrates on exploiting the frame coherence for a painterly animation sequence. The jittered grid is used as the initial stroke distribution. A stroke color is determined by taking from the source image just same as Haeberli's work. The stroke size may vary, because a stroke may be clipped by the edges detected in the source image to preserve the contours of objects. However, the stroke width is constant over the canvas. The orientation is determined by the intensity gradient, which is similar to Haeberli's method[8] as well. The painting order is randomized to obtain a hand-crafted touch.

Hertzmann[9] used the brush strokes represented by spline curves. The method composes an image with several layers. Each layer is painted by the strokes with its own constant width. The upper layer is painted with narrower strokes, where the significant difference remains between the source image and the composited image with the lower layers. As a result, the canvas image gradually gets close to the source image. The resulting image contains strokes with several widths. The orientation of spline curves are controlled by the intensity gradient.

The proposed method is unique in respect of the intensive use of the local region of the source image. Basically, the previous methods consider the stroke location as a point, and do not care about the surrounding region. For example, the orientation is determined by the intensity gradient at the stroke location. This method may lead to poor results becuase the gradient is sensitive to the noises around the pixel. Accordingly, it is significant to consider the region around the stroke location. The stroke placement should be controlled to fit the region of the source image, because the stroke itself has a finite area. The previous methods sometimes adopt blur-

ring to incorporate the local region. Our interest is to exploit more information of the pixels around the stroke location.

Another important aspect is the use of the color information. The previous methods sometimes think little of the color, which is one of the most important features in the painterly rendered images. As stated above, they only uses the intensity values for the orientation determination. The intensity is merely one aspect of the colour. We must consider the chromaticity information in order to approximate the local region of the source image.

3 Local Source Image Approximation

This section discusses a method for approximating a local source image with a single stroke. The complete painting algorithm will be discussed in Section 4.

The goal of this section is to determine the stroke size, orientation, and location based on the stroke color and on the local region of the source image. This can be formalized as follows:

> Given a portion of the source image $\mathbf{C}_w$ and the stroke color $\mathbf{C}$, determine the other attributes of the stroke: location (x_c, y_c), orientation θ, and size (w, l), to approximate $\mathbf{C}_w$ with the stroke.

This 'color-first' strategy resembles the painting process of artists. An artist first prepares a color on a palette, then places the brush on the canvas at a suitable location.

This is achieved by a two-step algorithm. Firstly, the color difference image of $\mathbf{C}_w$ from $\mathbf{C}$, is generated. This step is explained in section 3.1. Then, a rectangular shape is calculated which approximates the color difference image. This is done by computing the image moments of the color difference image. This step is described in section 3.2.

3.1 Color Difference Image

A **color difference image** is a gray-scale image, $I : R^2 \rightarrow R$, obtained from a portion of the source image $\mathbf{C}_w$ and a stroke color $\mathbf{C}$. Its pixel value $I(x,y)$ is larger if the corresponding $\mathbf{C}_w(x,y)$ is closer to the stroke color $\mathbf{C}$. $I(x,y)$ is given by the following equation,

$$I(x,y) = f(d(\mathbf{C}, \mathbf{C}_w(x,y))).$$

Here, $d(\mathbf{C}_1, \mathbf{C}_2) \in R$ is the distance of two colors, $\mathbf{C}_1$ and $\mathbf{C}_2$, in the CIE-$L^*u^*v^*$ color space. In this perceptually uniform color space, the color distance between $\mathbf{C}_1$ and $\mathbf{C}_2$ is given by the Euclidean distance[7].

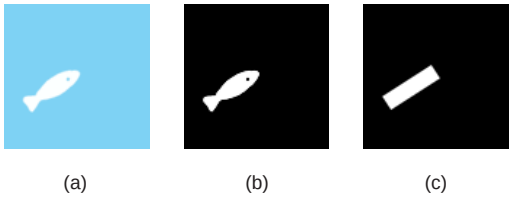
$f(d) \in [0,1]$ is the function that maps the color distance d to the intensity of the color difference image. In the current implementation,

$$f(d) = \begin{cases} (1-(d/d_0)^2)^2 & \text{if} \quad d \in [0, d_0] \\ 0 & \text{if} \quad d \in [d_0, \infty) \end{cases},$$

is used to avoid the square-root operation in the distance calculation for the computational efficiency. d_0 is a constant value which reflects the color distance range of 'similar color.' In the current implementation, $d_0 = 150$ is used as a default value. Figure 4(b) is a color difference image generated from Figure 4(a). In this case, $\mathbf{C}$ is white.

3.2 Image Moment

The **image moment**[10] is defined on the gray-scale image, $I(x,y) \in [0,1]$. The image moment of lth degree about the x-axis and mth degree about the y-axis is defined as,



(a) (b) (c)

Figure 4: The approximation based on image moments: (a) A local source image. (b) A color difference image. (c) A binary image of the equivalent rectangle.

$$M_{lm} = \sum_x \sum_y x^l y^m I(x,y).$$

For $n = l + m$, M_{lm} is called the image moment of nth degree. The image moment of zeroth degree, M_{00} is simply the sum of all the intensity values of the gray-scale image. Assuming that the image contains an object, M_{00} stands for the area of the object. The center of mass of the object can be calculated, combining M_{00} with the image moments of first degree, M_{10} and M_{01}. Using the moments of the second degree, M_{20}, M_{02} and M_{11}, together, the orientation of the object in the image can be calculated.

The binary image of the **equivalent rectangle** has the same zeroth, first and second moments as the given gray-scale image. Figure 4(c) is the image of the equivalent rectangle of Figure 4(b), such that these two images have the same image moments of up to second degree.

The coordinates of the center (x_c, y_c), the angle θ between the longer edge and the x-axis, and the length of edges, (w, l), are calculated as follows[6]:

$$\begin{aligned} x_c &= \frac{M_{10}}{M_{00}}, \\ y_c &= \frac{M_{01}}{M_{00}}, \\ \theta &= \frac{\tan^{-1}\left(\frac{b}{a-c}\right)}{2}, \\ w &= \sqrt{6\left(a+c-\sqrt{b^2+(a-c)^2}\right)}, \\ l &= \sqrt{6\left(a+c+\sqrt{b^2+(a-c)^2}\right)}, \end{aligned} \tag{1}$$

where a, b, and c are defined as,

$$\begin{aligned} a &= \frac{M_{20}}{M_{00}} - x_c^2, \\ b &= 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right), \\ c &= \frac{M_{02}}{M_{00}} - y_c^2. \end{aligned}$$

4 Algorithm

The discussion in the previous section enables to solve the issues in the preparation step of the painterly rendering shown in Figure 1. The image moments of the color difference image gives a powerful

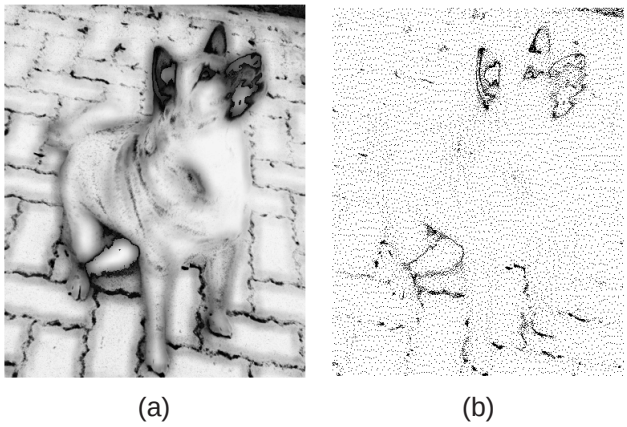


Figure 5: (a) A stroke area image. (b) The stroke distribution.

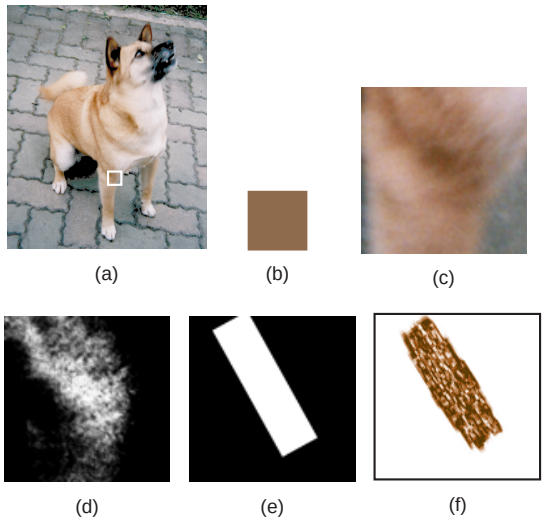


Figure 6: A process of stroke attributes determination: (a) A source image. (b) A stroke color. (c) A local source image. (d) A color difference image. (e) An image of the equivalent rectangle. (f) A rendered brush stroke.

tool to determine stroke distribution, size attributes, and painting order. Sections 4.1 through 4.3 explain the algorithms for determining them, respectively.

4.1 Stroke Distribution

The stroke density should be adapted according to the stroke area. Namely, the stroke density should be inversely proportional to the area of the stroke placed in the region. Therefore, the stroke area should be estimated first, then the density is determined based on the estimation.

The algorithm consists of the following two steps, stroke area estimation and sampling points generation.

1. **Stroke Area Estimation** The first step estimates the stroke area over the canvas. This step yields a **stroke area image** of the source image. The stroke area image is a gray-scale image whose intensity value represents the estimated stroke area to be used at the point of the image. The stroke area is given by the image moment of zeroth degree, M_{00}, as explained in section 3.2, once its color and the local region are fixed. The extent of this local region is specified by a user with a parameter s which stands for an edge length of a square. This parameter s will be further discussed in section 4.2. The stroke area image is obtained by calculating M_{00} for each pixel with its color and its surrounding square region. Figure 5(a) shows an example of the stroke area image, where the darker region indicates finer strokes.

2. **Sampling Points Generation** Based on the stroke area image, sampling points are determined. A sampling point indicates an tentative stroke location. Each stroke will be shifted in the next step of determining the stroke attributes. The sampling points should be distributed densely where the stroke area image is dark, while sparsely where the stroke area image is light.

   This is exactly a kind of dithering process that distributes dots over the image. Therefore the clustered dithering algorithms are not suitable. Furthermore, the dots should be uniformly assigned without any lack. A matrix dithering that may induce such lacks is also inappropriate.

   This study uses a modified version of a space-filling curve algorithm[16] without clustering. For each pixel along the space-filling curve, $1/M_{00}$ is summed up. When the sum reaches a certain threshold, the pixel is adopted as a sampling point and the summation restarts with the remainder. Particularly, the space-filling curve used here is the Murray polygon[2], because it can deal with a non-square image. Figure 5(b) shows the sampling points generated by this method.

4.2 Stroke Attributes

The stroke distribution described above gives an tentative location of each stroke. All stroke attributes are fixed with the following four steps.

1. **Color Sampling** The color attribute **C** is defined as the color at the tentative location of the source image. The color shown in Figure 6(b) is the result of sampling at the center of the white rectangle in Figure 6(a).

2. **Source Image Cropping** In this step, the source image is cropped with an $s \times s$ square, whose center is located at the tentative location. This cropped image is the local region to be approximated by the stroke. Figure 6(c) is the result of this step. The cropping size s is specified by a user. The area of the resulting stroke do not exceed that of the cropping square, s^2. This means that the parameter allows the user to control the maximum stroke size.

3. **Color Difference Image Generation** The color difference image is calculated with **C** and the cropped image as explained in section 3.1. Figure 6(d) is the color difference image after this step.

4. **Equivalent Rectangle Calculation** This step computes the equivalent rectangle of the color difference image based on its image moments. The parameters of the equivalent rectangle, x_c, y_c, θ, w, and l, are calculated with equations (2). The parameters of the stroke are set to those of the equivalent rectangle as described in section 3. Figure 6(e) shows the equivalent rectangle to Figure 6(d). The actual stroke location is shifted to the center of the equivalent rectangle. In the case of Figure 6, the stroke is moved to the upper-left. Figure 6(f) shows the textured brush stroke, which will be composited on the canvas.

4.3 Painting Order

After the attributes of all strokes are determined, the strokes are sorted by the area of the equivalent rectangle, *wl*. Larger brush strokes are painted before the smaller ones. The resulting image preserves the details of the source image, because the smaller strokes are not hidden by the larger ones.

The example shown in Figure 7 demonstrates the composition process with sorted brush strokes. The figure shows the canvases after painting larger 2500 strokes, 5000 strokes, 7500 strokes, and 10000 strokes. The details in the source image emerge gradually as smaller strokes are painted.

5 Results

Figure 8 shows some results. The input source image is shown in Figure 8(a), whose size is 400×490 pixels. Figure 8(b) and Figure 8(c) are generated with different values of the parameter *s*. Figure 8(b) was obtained with $s = 25$. The body of the dog is painted with larger strokes, while the details such as the face are drawn well. Figure 8(c) is painted with $s = 15$. Compared with Figure 8(b), the image is rendered by smaller brush strokes. These results demonstrate that the user can control the coarseness by the parameter *s*. Other examples are shown in Figure 9.

6 Conclusion and Future Work

This paper proposes the techniques for the preparation step in painterly rendering. The method varies the stroke size over the canvas, which enriches the expressiveness. The details are painted with finer strokes, while the flat area with larger ones. The stroke density is properly controlled by applying the space-filling-curve dithering technique to the stroke area image. This method takes into account of the chromaticity as well as intensity to determine the stroke attributes. Therefore the resulting stroke attributes are suitable for approximating the local region of the source image. Furthermore, the painting order is determined based on the stroke sizes. The smaller strokes are painted later than larger strokes. In this way, the resulting image preserves and the details of the source image.

There is much work to be done for the image moment-based painterly rendering. For instance, some strategy for determining parameter *s* would be profitable. In our current implementation, a user-specified value *s* is fixed all over the canvas. Since this parameter explicitly limits the stroke size, the coarseness of the result can be adaptively controlled by varying the parameter *s*. This parameter *s* affects the separation of close objects. For example, in Figure 9, three toes are painted with a single stroke because they are covered by a cropping square.

Another problem is finding the minimum number of strokes to cover the overall canvas area. The examples in this paper are composited with many strokes to avoid the 'cracks', which could happen when the stroke density is not large enough. A porous texture image makes the problem more difficult.

References

[1] T. Cockshott, J. Patterson, and D. England. Modelling the texture of paint. *Computer Graphics Forum*, 11(3):C217–C226, C476, 1992.

[2] A. J. Cole. Halftoning without dither or edge enhancement. *The Visual Computer*, 7:232–246, 1991.

[3] Fractal Design Corporation. Fractal design painter.

[4] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture mapping for cel animation. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 435–446, July 1998.

[5] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 421–430, August 1997.

[6] William T. Freeman, David B. Anderson, Paul A. Beardsley, Chris N. Dodge, Michal Roth, Craig D. Weissman, William S. Yerazunis, Hiroshi Kage, Kazuo Kyuma, Yasunari Miyake, and Ken ichi Tanaka. Computer vision for interactive computer graphics. *IEEE Computer Graphics and Applications*, 18(3):42–53, May/June 1998.

[7] Andrew S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, CA, 1995.

[8] Paul Haeberli. Paint by numbers: Abstract image representations. *Computer Graphics*, 24(4):207–214, August 1990.

[9] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In Michael F. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 453–460, July 1998.

[10] Berthold K. P Horn. *Robot Vision*. MIT Press, Cambridge 1986, 1986.

[11] Peter Litwinowicz. Processing images and video for an impressionist effect. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 407–414, August 1997.

[12] Barbara J. Meier. Painterly rendering for animation. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 477–484, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

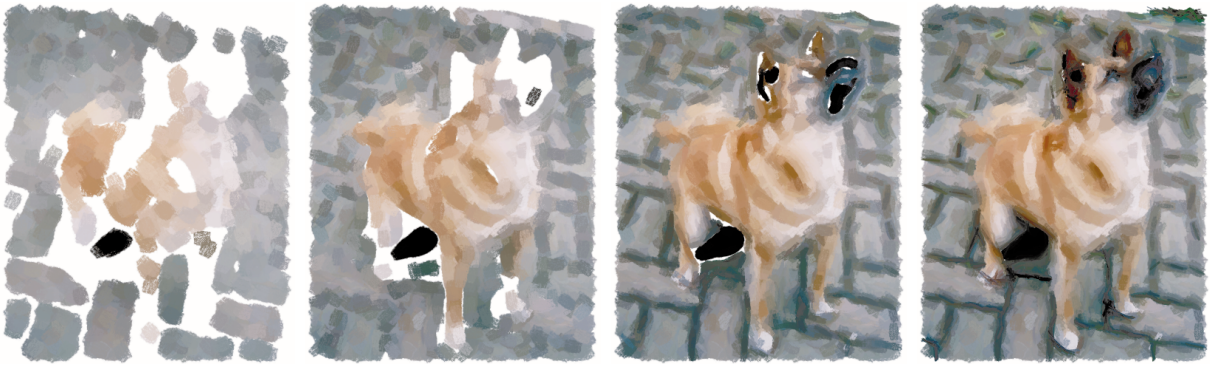
[13] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *Computer Graphics*, 24(4):197–206, August 1990.

[14] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 401–406, August 1997.

[15] Adobe Systems. Photoshop 5.0.

[16] Luiz Velho and Jonas de Miranda Gomes. Digital halftoning with space filling curves. *Computer Graphics*, 25(4):81–90, July 1991.

[17] Michael T. Wong, Douglas E. Zongker, and David H. Salesin. Computer-generated floral ornament. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 423–434, July 1998.

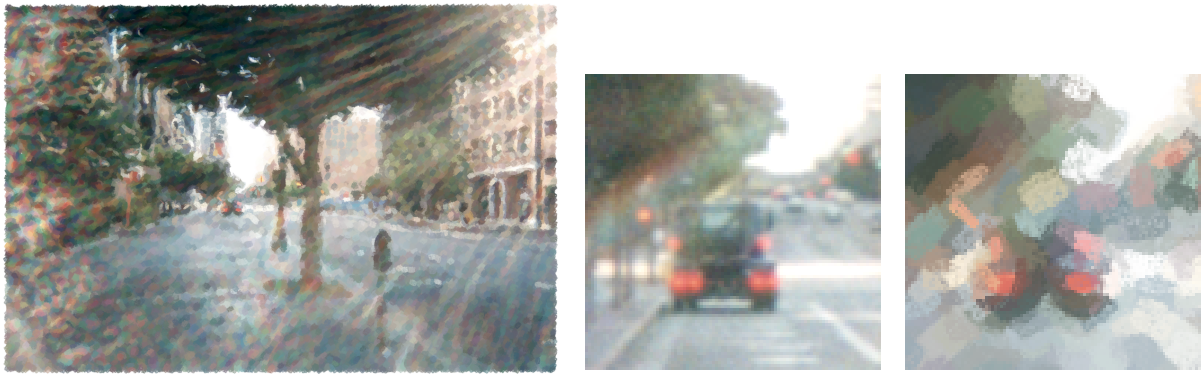


(a) (b) (c) (d)

Figure 7: Canvas images after larger (a)2500, (b)5000, (c)7500, and (d)10000 strokes. The final image is Figure 8(b), 11048 strokes.

(a) (b) (c)

Figure 8: Influence of the parameter s. (a) A source image. (b) $s = 25$, 11048 strokes. (c) $s = 15$, 9673 strokes.



(a) (b) (c)

Figure 9: (a) *A quiet street in the afternoon.* (1200×821 pixels, $s = 15$, 37640 strokes.), (b) Close-up of the car at the center of the image. (c) Corresponding local region of the source image.