芸術科学会論文誌 Vol.3 No.4 pp.224―234

Stroke-Based *Suibokuga*-Like Rendering for Three-Dimensional Geometric Models

- *Ten* and *Shun* Touches -

Youetsu SATO†, Tadahiro FUJIMOTO†, Kazunobu MURAOKA‡ and Norishige CHIBA†

† Department of Computer and Information Science, Faculty of Engineering, Iwate University
‡ Department of Communications Engineering, Tohoku Institute of Technology

Abstract

Non-photorealistic rendering has become an important research topics in computer graphics in recent years. We have previously proposed a non-photorealistic rendering method to generate *Suibokuga*-like images of trees. This method was suitable only for representing trees in *Mokkotsuho* paintings because the images were generated from three-dimensional skeleton data. In this paper, we propose a method to generate *Suibokuga*-like images of arbitrary objects from three-dimensional geometric models, such as polygonal models. The proposed method realizes *Kou*, *Ten,* and *Shun* brush stroke techniques for creating *Sensenbyoho* paintings, which are a typical *Suibokuga* style for representing landscapes. Moreover, the images can be generated from arbitrary viewpoints and light source information. The ability of this method is demonstrated by showing various example.

Keywords: non-photorealistic rendering (NPR); *Suibokuga*-like rendering; three-dimensional geometric model; brush stroke; simulation of ink diffusion; cellular automaton;

1. Introduction

There has been a significant amount of research activity in recent years centered on the area of computer graphics dealing with non-photorealistic rendering [1,2]. In contrast with photorealistic rendering, the aim of non-photorealistic rendering is to create images identical to those drawn by humans such as paintings, cell animations, and technical illustrations. The following approaches have been reported.

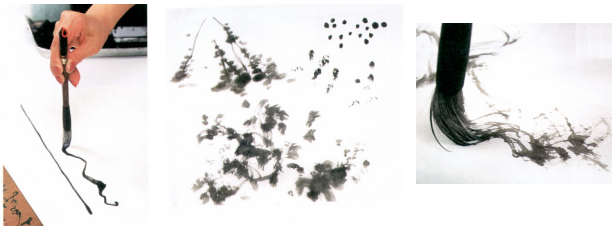
(1) Methods in which an input image such as a photograph is used, and the output image is generated automatically [3,4,5]
(2) Methods in which a mouse or drawing tablet is used to interactively create the image [6,7,8]
(3) Methods in which the image is created automatically from a three-dimensional geometric model [9,10,11,12,13]

Suibokuga, or *Sumi-e*, is an attractive and traditional painting style in which strokes are drawn in Chinese black ink, called *Sumi*, by a brush on paper. We have previously proposed a method, which is categorized into type (3) above, for generating *Suibokuga*-like images of trees from three-dimensional skeleton data [11]. In this method, brush stroke information was obtained from the skeleton data, and cellular automata were used to simulate the transfer and diffusion of water and ink (*Sumi*) on paper and in a brush.

In this paper, we propose a method that is extended from the previous one. The proposed method generates *Suibokuga*-like images of arbitrary objects other than trees, so long as the objects are defined as three-dimensional geometric models, such as polygonal models, from which necessary data can be obtained. This method creates *Sensenbyoho* paintings, which are typically found in landscape paintings (*Sansuiga*), by *Kou*, *Ten,* and *Shun* brush stroke techniques.

Some methods for generating *Suibokuga*-like images and other related methods have been reported. Strassmann proposed a technique for generating very impressive *Suibokuga*-like images by modeling brush strokes using cubic splines, which were computed from the position and pressure data which the user inputted by a keyboard [14]. Small's technique realized the effect of the diffusion of pigments inside paper. In this technique, the interaction between the paper fibers and the pigments is simulated on a cellular automaton using the parallel computer architecture [15]. Although his paper treated watercolors, not *Suibokuga* paintings, the technique is considered to be greatly relevant to the ink blur effect, called *Nijimi*, on paper in *Suibokuga* paintings. Guo and Kunii proposed a technique that also achieved the *Nijimi* effect by taking into consideration the absorption of paper and the flow and density change of water and ink [16]. This technique has realized the *Nijimi* effect peculiar to *Suibokuga* paintings based on a different approach from Small's technique. Curtis et al. proposed a technique for reproducing various phenomena by watercolors and paper [6]. However, this technique is not suitable for realizing the effects peculiar to *Suibokuga* paintings by itself. As another theme using *Sumi*, techniques for reproducing Chinese or Japanese calligraphy, called *Shodo,* in which characters are written using *Sumi* and a brush on paper, are also proposed. Moreover, research on brush models for interactively

(a) Vertical motion (*Choku-Hitsu*) (b) Inclined motion (*Soku-Hitsu*)
Fig .1 Basic brush motion techniques [25]



(a)*Kou* (b) *Ten* (c) *Shun*
Fig.2 Basic brush stroke techniques [27]

creating *Suibokuga* paintings and characters in calligraphy on a computer has been done[17,18,19,20].

Some methods have been proposed for creating *Suibokuga*-like images from three-dimensional geometric models. They are categorized into the following two types:

(i) Outlines are drawn with strokes and shaded areas are rendered using toon shading or texture mapping of stroke patterns [21,22,23].

(ii) Both of outlines and shaded areas are drawn with strokes [24].

The method proposed in this paper uses the approach (ii). This method is different from other methods in that it aims to realize *Sensenbyoho* technique, which is often used for landscape paintings, by appropriately drawing Kou, Ten, and Shun brush strokes and changing stroke thickness according to depth values obtained from three-dimensional geometric models.

In Section 2, the features of actual *Suibokuga* paintings and basic representation techniques are described. In Section 3, the *Suibokuga*-like rendering method proposed in [11] is summarized. In Section 4, a *Suibokuga*-like rendering method for three-dimensional geometric models is proposed. In Section 5, the effectiveness of the proposed method is demonstrated by showing various examples. Finally, section 6 provides conclusions and themes for future research.

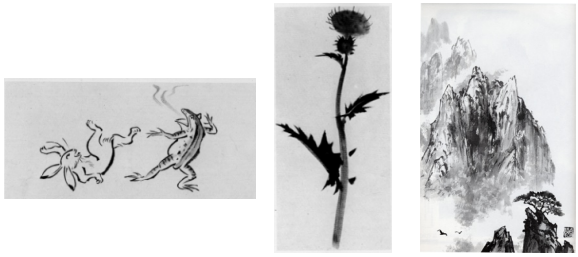
2. Features of *Suibokuga*

Suibokuga are paintings in which only light and dark shades of ink are used to represent all the colors and shapes of objects. In this section, basic brush and drawing techniques distinctive of actual *Suibokuga* are briefly summarized[25,26,27,28,29].

2.1 Basic *Suibokuga* brush techniques

In *Suibokuga,* the following brush motion and stroke techniques, which mean how to move a brush and how to generate strokes, are used.

There are two basic brush motion techniques: vertical motion (*Choku-hitsu*) and inclined motion (*Soku-hitsu*), as shown in Fig.1. When a brush is moved in a vertical direction, the axis of the brush is inclined in the same direction as the direction of motion, and the brush tip passes through the center of the drawn line,



(a) *Hakubyoho* (b) *Mokkotsuho* (c) *Sensenbyoho*
Fig.3 Basic drawing techniques [26,28]

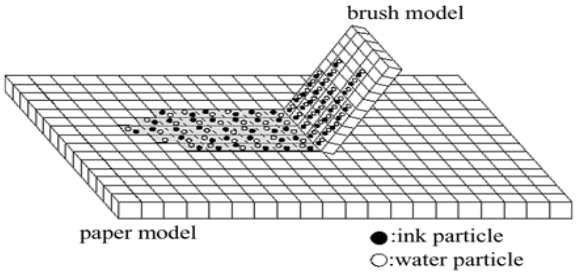


Fig.4 Fundamental models of *Suibokuga*-like rendering method[11]

as shown in Fig.1 (a). On the other hand, in an inclined motion, the brush axis is inclined and the body of the brush is used to draw a stroke, as shown in Fig.1 (b). In general, vertical brush motions are used to draw powerful stroke lines; the thickness of the strokes can be varied by adjusting the pressure applied to the brush and the manner in which the brush is lowered. In contrast, inclined brush motions are used to draw gentle stroke lines; the gradation of ink can be produced by varying the quantity of ink applied. In particular, inclined brush motions can be used to produce scratchiness effects, which are called *Kasure*. In the proposed method, we treat only vertical brush motions.

Basic brush stroke techniques include *Kou*, *Ten*, and *Shun,* as shown in Fig.2.

Kou: The most basic line strokes in *Suibokuga.* This is used mainly when drawing outlines.

Ten: Points or dots produced by dabbing the paper with the tip of a brush. This is used to draw shaded areas, such as backgrounds.

Shun: Strokes used for representing the texture and roughness of the surface of an object. This is used to draw shaded areas enclosed with outlines drawn by using *Kou*.

2.2 Basic *Suibokuga* drawing techniques

Basic drawing techniques used in *Suibokuga* are mainly *Mokkotsuho*, *Hakubyoho*, and *Sensenbyoho* (cf. Fig.3).

Hakubyoho: Only the outlines of an object are drawn by using *Kou*.

Mokkotsuho: The whole shape of an object is drawn with a single brush stroke, without drawing the outlines of the object.

Sensenbyoho: The outlines of an object are drawn by using *Kou*, and shaded areas are drawn by using *Ten* and *Shun,* etc. This drawing technique is typically found in landscape paintings.

(a) Vertical motion (*Choku-hitsu*) (b) Inclined motion (*Soku-hitsu*)

Fig.5 Examples of generated ink strokes representing basic brush motion techniques

Fig.6 An example of generated ink stroke representing the effect of *Kasure*

The previously proposed method [11] could represent a certain kind of objects, such as trees and grass, and generate *Mokkotsuho*-like images. On the other hand, the method proposed in this paper is extended to enhance the ability to represent various kinds of objects and realize *Sensenbyoho*-like images.

3. Fundamental Models of *Suibokuga*-like Rendering Method

In the proposed method, in order to generate ink strokes on paper, the models from the previous work [11] are utilized. The method composed of a "transfer/diffusion model of water and ink particles", "paper model", and "brush model", which are shown in Fig.4. In this section, we explain these models briefly; the details are described in [11].

The transfer/diffusion model of water and ink particles is used for simulating the transfer and diffusion of water and ink on paper or in a brush. This model is realized using a two-dimensional cellular automaton. First, paper or a brush is modeled as a two-dimensional grid array. Then, water and ink particles move on the array according to a certain cellular automaton rule. Each cell on the array has a 'tank', which is connected to the tanks of the four nearest-neighbor cells by 'pipes'. The tank is given a water capacity, a bottom height, and a pipe height, which characterize the paper or brush model. These values control the quantity and directionality of the transfer and diffusion of water and ink, and this results in creating various characteristic impressions of ink blurring and scratching.

The paper model is defined as a two-dimensional grid array, as described above. Paper texture formed by fibers is realized by putting line segments on the array at random position and direction. These line segments determine the water capacity and the bottom height of the tanks on the array.

The brush model is also defined as a two-dimensional grid array. As shown in Fig.4, water and ink particles are transferred and diffused in the brush model and move onto the paper model. Then, these particles are transferred and diffused on the paper

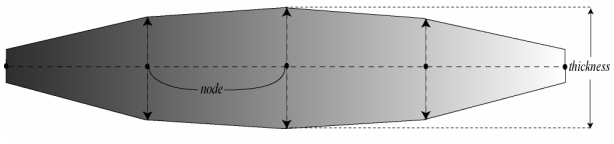


Fig.7 Geometric representation of an ink stroke shape

model. The width of an ink stroke drawn on the paper is changed according to the pressure applied to the brush. This is achieved by controlling the number of brush cells in contact with the paper according to the brush pressure. The stroke width B_w is obtained from the brush pressure P as follows.

$$B_w = \gamma\,(1 - exp(-P)) \qquad (1)$$

Here, γ is a predefined constant.

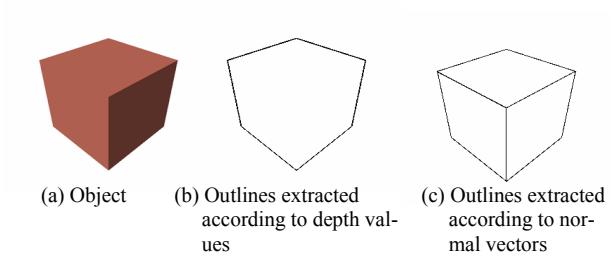
Figures 5 and 6 show examples of ink strokes generated using the above models. Figure 5 shows examples that realize two basic brush motion techniques described in Section 2.1 (cf. Fig.1). (a) shows strokes by vertical motion (*Choku-hitsu*), and (b) shows strokes by inclined motion (*Soku-hitsu*). Figure 6 shows the effect of *Kasure,* which is generated when the brush velocity is high and the supply of ink to the paper is insufficient.

4. *Suibokuga*-like Rendering Method for Three-Dimensional Geometric Models

The ability of the previously proposed method [11] was limited to generating only *Mokkotsuho*-like images of trees by giving brush strokes obtained from three-dimensional skeleton data. We extend this method to generate *Sensenbyoho*-like images of arbitrary objects using three-dimensional geometric models such as polygonal models. In the proposed method, any geometric model is available as long as necessary data can be obtained from the model (cf. Section 4.1). As described in Section 2.2 and shown in Fig.3, *Sensenbyoho* paintings are drawn with *Kou*, *Ten*, and *Shun* strokes; *Kou* strokes are used to draw the outlines of an object, while *Ten* and *Shun* strokes are used to draw shaded areas. The proposed method creates these strokes properly from given geometric models.

4.1 Pixel data

First, three-dimensional geometric models are processed to obtain three kinds of 'pixel data': depth values, intensity values, and normal vectors on all the pixels of the screen on which a *Sensenbyoho*-like image of the models is generated. On each pixel $P_{i,j}$, the depth value $Z_{i,j}$ is the distance from the viewpoint to the intersection point on the surface of the model that is first hit by the ray from the viewpoint through the pixel. The intensity value $I_{i,j}$ of the intersection point is obtained by shading using light source information as brightness in monochrome. The normal vector $\mathbf{N}_{i,j}$ of the intersection point is calculated from the geometric data of the model and normalized to make its length 1. When the ray does not hit any surfaces, the pixel is



(a) Object (b) Outlines extracted according to depth values (c) Outlines extracted according to normal vectors

Fig.8 Examples of outline extraction

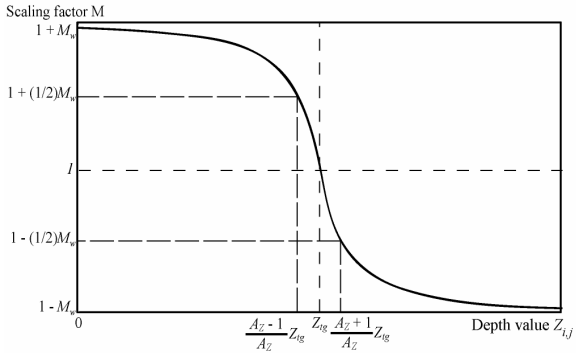


Fig.9 Relationship between depth value and scaling factor

given infinity as its depth value, a background color value as its intensity value, and nothing as its normal vector. These three kinds of pixel data are registered with the pixels and used for generating *Kou*, *Ten*, and *Shun* strokes, as described in the following sections. Any kind of geometric model is available as long as the above pixel data can be obtained. In this paper, we mainly use polygonal models as examples because of simplicity.

4.2 Stroke data

In order to represent the shape of an ink stroke on paper, we adopt the representation form shown in Fig.7. In this representation, some nodes are linked in a line to represent a stroke. A series of the branches linking the nodes is called 'stroke axis'. Each node is given a position and a thickness. A trapezoid is determined on each branch between two nodes, and all the trapezoids constitute a stroke shape, as shown in Fig.7. Each node is put exactly on the position of a pixel on the screen to use the correspondent pixel data registered with the pixel (cf. Section 4.1).

This geometric representation is transformed to a natural-looking ink stroke by the rendering method described in Section 3. In a stroke, the brush model moves along its stroke axis and traces its nodes and branches. In this tracing, the brush pressure P in Eq.(1) at each position is determined as the thickness value obtained by linear interpolation of the thicknesses of the nodes. Water and ink particles are put along the stroke axis, and are transferred and diffused on the paper using a cellular automaton model, as shown in Fig.4. This process generates a smooth and natural-looking ink stroke on the paper, although the original geometric representation of Fig.7 has a simple polygonal shape.

4.3 Generation of outline strokes

To generate strokes for outlines (*Kou* strokes), first, the screen is scanned to extract all the pixels constituting outlines by using the pixel data (cf. Section 4.1). The extracted pixels are called 'outline pixels'. Next, the method of thinning is applied to the outline pixels. Then, the screen is scanned again to find outline pixels to be starting pixels. When a starting pixel is found, a series of adjacent outline pixels are traced from the starting pixel until a terminal condition is satisfied; these outline pixels are registered as an 'outline stroke'. This process is repeated until all the outline pixels are registered as outline strokes. The details of this algorithm are described below. This helps readers to easily implement it. Besides, for the outline stroke detection algorithm of Step 3, other methods, such as those proposed in [12,13], are also available.

Step 1. To extract outline pixels, all the pixels $P_{i,j}$ on the screen are tested whether the condition of Step 1-1 or 1-2 is satisfied. Every outline pixel is extracted by either Step 1-1 or 1-2, as shown in Fig.8.

Step 1-1. For the depth value $Z_{i,j}$ of pixel $P_{i,j}$, if $Z_{i,j}$-$Z_{i-1,j} > D_z$ or $Z_{i,j}$-$Z_{i,j-1} > D_z$, then $P_{i,j}$ is registered as an outline pixel. Here, D_z is a threshold value. This step detects an outline pixel in the case that the difference in depth values between adjacent pixels is large. This case is caused by the boundaries of objects, as shown in Fig.8 (b).

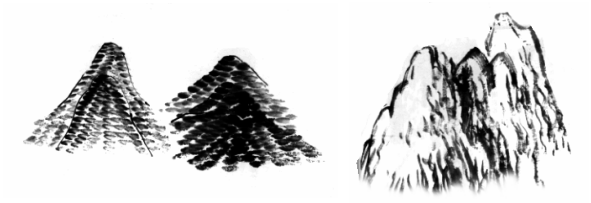
Step 1-2. If pixel $P_{i,j}$ is given normal vector $\mathbf{N}_{i,j}$, then the inner product $IP_{i,j} = \mathbf{N}_{i,j} \cdot \mathbf{L}$ is calculated, where $\mathbf{L}$ is the light source vector. If $IP_{i,j}$-$IP_{i-1,j} > D_N$ or $IP_{i,j}$-$IP_{i,j-1} > D_N$, then $P_{i,j}$ is registered as an outline pixel. Here, D_N is a threshold value. This step detects an outline pixel in the case that the directions of the normal vectors in adjacent pixels are significantly different. This case is related to the curvature of the surface, as shown in Fig.8 (c).

Step 2. The method of thinning is applied to the outline pixels extracted in Step 1 to thin and clear outline strokes obtained as follows.

Step 3. Steps 3-1 to 3-3 detect a series of outline pixels to be registered as an outline stroke. These three steps are repeated until all the outline pixels are added to outline strokes.

Step 3-1. Among the outline pixels that have not been added to any outline strokes yet, if there are outline pixels that are endpoints of outlines, then the most upper-left outline pixel is selected as a starting pixel of an outline stroke. If there are not such outline pixels, then an arbitrary outline pixel is selected as a starting pixel.

Step 3-2. A series of adjacent outline pixels are traced from the starting pixel. If the trace reaches a branch point that has more than two adjacent outline pixels to trace next, then one adjacent outline pixel with a depth value closer to that of the current outline pixel is chosen. If there are more than two adjacent outline pixels that have the same depth value, then one



(a) *Ten* strokes (b) *Shun* strokes

Fig.10 Examples of strokes in shaded areas in actual *Suibokuga* paintings[28,29]

(a) Basic *Ten* stroke (b) Basic *Shun* stroke

Fig.11 Basic forms of *Ten* and *Shun* strokes

adjacent outline pixel is chosen so that the trace can proceed in the direction closest to the current tracing direction.

Step 3-3. If one of the following conditions is satisfied, the trace is finished. Then, all the outline pixels from the starting pixel to the current outline pixel are registered as a single outline stroke.

Condition 1: The difference between the depth values of the current outline pixel and the next outline pixel is greater than a predefined value.
Condition 2: The tracing direction changes significantly.
Condition 3: The trace reaches an endpoint.
Condition 4: The trace forms a loop.

Each outline stroke obtained above is transformed to the representation form described in Section 4.2 (cf. Fig.7). Some outline pixels are thinned out, and the remaining outline pixels become nodes. Specifically, outline pixels lining in the same direction are considered to constitute one branch; only the outline pixels positioned on the endpoints of such branches are selected as nodes (node pixels), and other outline pixels are removed. Then, each node pixel $P_{i,j}$ is given a thickness determined by using the depth value $Z_{i,j}$ registered with the pixel. First, a 'basic thickness' for all the node pixels is predetermined. Then, the scaling factor M for the node pixel $P_{i,j}$ is calculated by the following equation (cf. Fig.9).

$$M = 1 - 2M_w \left(\tan^{-1} A_Z \left(\frac{Z_{i,j}}{Z_{tg}} - 1 \right) \Big/ \pi \right) \qquad 0 \le M_w \le 1 \qquad (2)$$

As shown in Fig.9, the range of M is from 1- M_w to 1+ M_w for a given constant M_w. The constant Z_{tg} is a predefined depth value; if $Z_{i,j}$ is equal to Z_{tg}, then M is equal to 1. The constant A_Z is used for controlling the gradient around (Z_{tg}, 1). By positioning a target point in the three-dimensional space, the value of Z_{tg} is determined as the distance from the viewpoint to the target point. When $Z_{i,j}$ becomes small (that is, the intersection point on the object surface is near the viewpoint), M becomes large and approaches 1+ M_W. When $Z_{i,j}$ becomes large (that is, the intersection point is far from the viewpoint), M becomes small and approaches 1 - M_W. Using Eq.(2), the thickness of the node pixel $P_{i,j}$ is determined by multiplying the basic thickness by the scaling factor M. If $Z_{i,j}$ is equal to Z_{tg}, the thickness is the same as the basic thickness. If $Z_{i,j}$ is smaller, the thickness becomes larger, and vice versa. Equation (2) gives the effect that the thicknesses of node pixels having depth values near Z_{tg} are given greater difference.

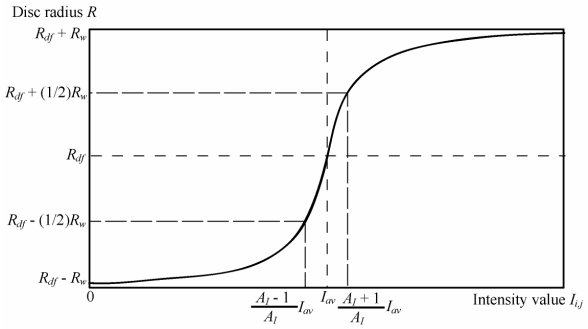


Fig.12 Relationship between intensity value and disk radius in Poisson disk sampling

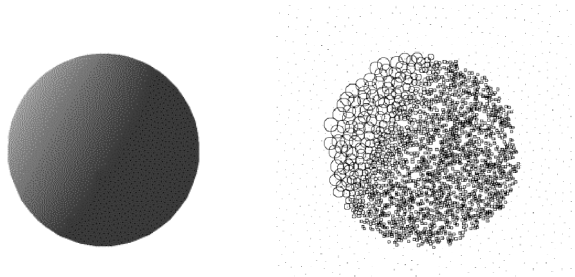
4.4 Generation of strokes in shaded areas

After drawing outlines, the texture and roughness of the surface of an object are represented by drawing *Ten* and *Shun* strokes. *Ten* strokes are represented by drawing dots, as shown in Fig.10 (a). *Shun* strokes are represented by guiding the brush in directions so as to trace the slope of the surface, as shown in Fig.10 (b). These strokes are generated by the methods explained below.

4.4.1 Method for generating *Ten* strokes

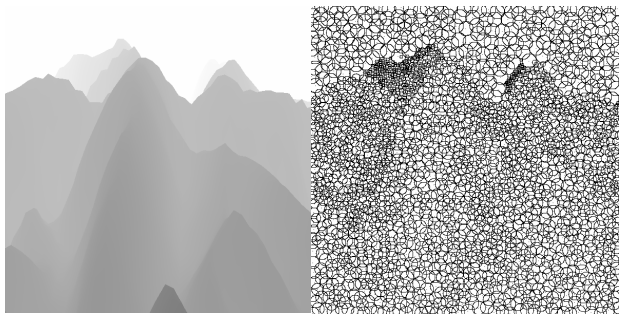
Actual *Suibokuga* paintings have various types of *Ten* strokes [27,28,29]. Among these types, the proposed method realizes the most basic type; a *Ten* stroke has the shape like a grain of rice. In the proposed method, all *Ten* strokes are produced based on a 'basic *Ten* stroke', as shown in Fig.11 (a). The basic *Ten* stroke is defined using the representation form described in Section 4.2; all the nodes are arranged on adjacent pixels straight in horizontal direction and given proper thicknesses so as to form the shape of a typical *Ten* stroke. Here, the node centered in a *Ten* stroke is called 'center node'. A *Ten* stroke actually used in a generated image is produced by determining the position of its center node on the screen and the thickness of each node properly. A proper number of *Ten* strokes are generated on the screen by the following procedure.

Step 1. First of all, the positions of the center nodes of all *Ten* strokes are determined using the Poisson disc sampling method with variable disc radii. Generally, the Poisson disc sampling method is a technique



(a) A rendered image having intensity values (b) Disc radii of sampling points

Fig.13 An example of the relationship between intensity values and disc radii



(a) An image based on depth values (b) Scaling factors

Fig.14 An example of the relationship between depth values and scaling factors

that distributes sampling points on the plane as uniformly as possible. Giving variable disc radii to sampling points results in changing the density of sampling points on the place; the position of a sampling point given a disc radius is determined such that other sampling points do not fall within the disc that centers the sampling point and has the given disc radius. Using this method, each sampling point is put on a proper pixel position on the screen as the center node of a *Ten* stroke, which results in producing the proper density and distribution of *Ten* strokes for an objective image. Technically, sampling points are put on the screen sequentially one by one. After a new sampling point is put on a random pixel position $P_{i,j}$ that is not inside the discs of the sampling points that have already been put, the disc radius R of the new sampling point is determined using the intensity value $I_{i,j}$ registered with the pixel as follows (cf. Fig.12).

$$R = R_{df} + 2R_w\left(\tan^{-1} A_I\left(\frac{I_{i,j}}{I_{av}} - 1\right)\Big/\pi\right) \qquad 0 \le R_w \le R_{df} \quad (3)$$

Equation (3) is similar to Eq.(2). This results in emphasizing the change of intensity values around the average intensity value, and is different from a halftoning method that exactly reproduces the intensity values. As shown in Fig.12, the range of R is from R_{df} - R_w to R_{df} + R_w for the constant R_w. The constant I_{av} is the average intensity value; if $I_{i,j}$ is equal to I_{av}, then R is equal to R_{df}. The constant A_I is used for controlling the gradient around (I_{av}, R_{df}). When $I_{i,j}$ decreases (that is, the intensity is dark), R decreases and approaches R_{df} - R_w. When $I_{i,j}$ increases (that is, the intensity is bright), R increases and approaches R_{df} + R_w. Equation (3) controls the density of *Ten* strokes properly according to the intensity values on the screen. Figure 13 is an example showing the relationship between intensity values and disc radii. *Ten* strokes are generated densely in dark areas having smaller intensity values and sparsely in bright areas having greater intensity values. The parameters above are determined by a user by trial and error.

Step 2. The *Ten* strokes above are placed on the screen without rotation. This means that the stroke axes of the *Ten* strokes are directed horizontally. Thus, not only the center nodes but also all the other nodes are put exactly on pixel positions on the screen. Then, the thickness of each node is determined. This is achieved in the same way as described in Section 4.3. That is, first, the scaling factor M of the node pixel $P_{i,j}$ is calculated using the depth value $Z_{i,j}$ by Eq.(2). In this case, the constants M_w and A_Z are properly controlled, and given different values from those in the case of outline strokes (*Kou* strokes). Then, the thickness of the node pixel is determined by multiplying the thickness of the correspondent node of the basic *Ten* stroke by the scaling factor M. Figure 14 is an example showing the relationship between depth values and scaling factors. (a) is an image based on depth values; a pixel having greater depth value is given brighter intensity. (b) is obtained from (a); the radius of each circle indicates the scaling factor calculated at the center of the circle. In this example, the background pixels are given a certain depth value instead of infinity for convenience, thus the background in (b) is given circles having a radius given by the depth value.

4.4.2 Method for generating *Shun* strokes

There are also various types of *Shun* strokes in actual *Suibokuga* paintings[27,28,29]. The proposed method realizes *Shun* strokes that are drawn by moving the brush so as to trace the slope of the surface of an object. Similarly to *Ten* strokes, all *Shun* strokes are produced based on a 'basic *Shun* stroke' that has a typical shape of a *Shun* stroke, as shown in Fig.11 (b). Actually, the number of the nodes of a *Shun* stroke used in a generated image is not fixed, because the length of a *Shun* stroke is variable. Thus the basic *Shun* stroke is given a predefined number of nodes, and each node is given a proper thickness. The number of the nodes of a *Shun* stroke drawn on an image and their positions are determined according to the slope of the surface of an object in the way described in the procedure below. Here, one end node at which a stroke flow starts is called 'starting node'. A *Shun* stroke actually used in a generated image is produced by determining the positions of its starting node and following other nodes on the screen and giving proper thicknesses to these nodes. The procedure is as follows.

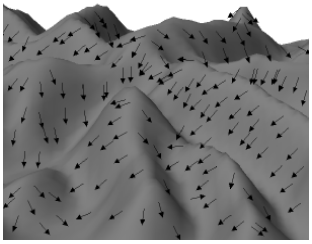


Fig.15 An example of vector field showing slope directions

Step 1. For each pixel $P_{i,j}$ with which a normal vector $\mathbf{N}_{i,j}$ of the surface of an object is registered, the slope direction of the surface at the pixel is determined. This slope direction is the tangential direction of the intersection line of the surface with the plane containing the upward vector **U** and the normal vector $\mathbf{N}_{i,j}$. Here, the upward vector **U** is the unit vector directed upward in the world coordinate system. The vector of the slope direction is obtained by $\mathbf{U}\times\mathbf{N}_{i,j}\times\mathbf{N}_{i,j}$, where × denotes the vector product. This vector is projected onto the screen, normalized, and registered with the pixel as 'slope direction vector'. All the slope direction vectors on the screen constitute a vector field, as shown in Fig.15. This field is named 'slope direction field'.

Step 2. In a similar way to Step 1 of Section 4.4.1, the positions of the starting nodes, which are called 'starting position', of all *Shun* strokes on the screen are determined using the Poisson disc sampling method based on the intensity values. In this case, the constants R_{df}, R_w, and A_1 in Eq.(3) are given proper values that are different from those in the case of *Ten* strokes.

Step 3. In order to generate *Shun* strokes, Steps 3-1 to 3-3 are repeated until all the starting positions determined above are processed.

Step 3-1. A starting position that has not been processed is chosen to generate a *Shun* stroke. From the starting position, a series of nodes are positioned sequentially on the screen along the flow of the slope direction field at intervals of a predefined distance. In this case, each node is actually placed on the pixel nearest the above position on the screen. The placement of nodes is finished when one of the following conditions is satisfied.

Condition 1: The path linking the nodes forms a loop.

Condition 2: The difference between the depth values registered with the current pixel and the next pixel is greater than a predefined value.

Condition 3: The path crosses over an outline stroke.

Condition 4: The length of the path exceeds a predefined value.

Step 3-2. In a similar way to Step 2 of Section 4.4.1, the scaling factor of each node obtained above is calculated. The constants M_w and A_Z in Eq.(2) are given proper values for *Shun* strokes. Then, the thickness of the node is determined by multiplying the thickness of the correspondent node of the basic *Shun* stroke by the scaling factor. In this case, if the number of the nodes in the generated *Shun* stroke is greater than that in the basic *Shun* stroke, the thickness of the node centered in the basic *Shun* stroke is used for the extra nodes in the middle part of the generated *Shun* stroke. If the number of the nodes in the generated *Shun* stroke is smaller, the nodes in the middle part of the basic *Shun* stroke are disregarded such that the numbers of the nodes in both the strokes are the same.

In Step 1, when a polygonal model is given, by using interpolated normal vectors obtained by the Phong smooth shading technique and so forth instead of normal vectors directly obtained from the polygonal model, smoother slope direction vectors are generated on the slope direction field. This results in generating natural smooth *Shun* strokes.

4.5 Generation of *Suibokuga*-like image

In order to create *Sensenbyoho*-like images, the three kinds of strokes proposed above are appropriately used. These strokes are rendered using the transfer/diffusion model of water and ink particles, the paper model, and the brush model, as described in Section 4.2. In actual *Suibokuga* paintings, near objects are drawn using thicker ink and far objects are drawn using thinner ink to give perspective effect. In order to realize this effect, in the proposed method, the quantity of water and ink contained in the brush are adjusted; near objects are drawn by reducing the number of water particles and increasing that of ink particles, and far objects are drawn by increasing the number of water particles and reducing that of ink particles.

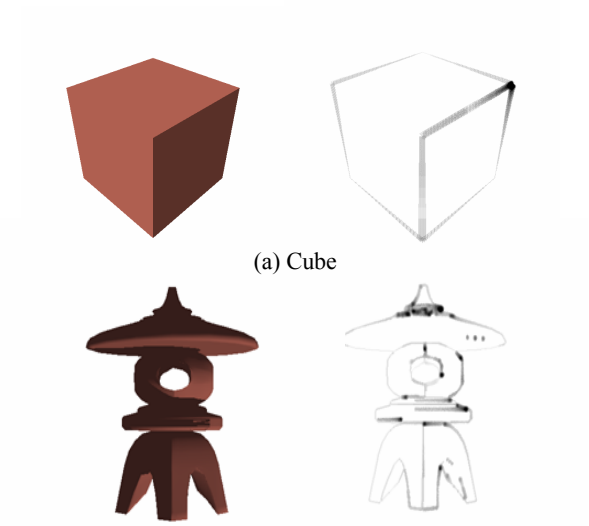
5. Examples

Several example images generated by the proposed method are shown in this section.

Figures 16, 17, and 18 show examples of rendered outline strokes and strokes in shaded areas. In these figures, the left images were generated by usual rendering method; the right images were generated using the method for outline strokes described in Section 4.3, that for *Ten* strokes in Section 4.4.1, and that for *Shun* strokes in Section 4.4.2, respectively.

Figures 19 and 20 show examples generated by merging outline strokes and shaded area strokes. In each of Figures 19 and 20, (a) shows an image including *Ten* strokes and (b) shows an image including *Shun* strokes. In Fig.20, the left images are generated by usual rendering method, whereas the right images are *Suibokuga*-like images generated by the proposed method. Comparing these images shows us that the light-source information of the scene affects the generated *Suibokuga*-like images.

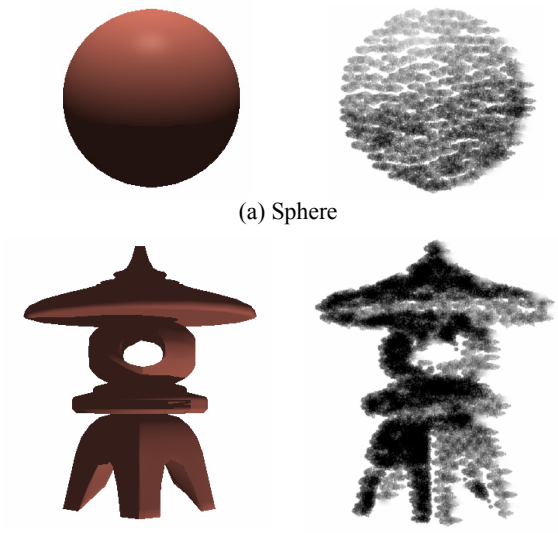
Figure 21 shows images generated from a polygonal model of tree. The left image was generated by usual rendering method, and the right image was given outline strokes and *Shun* strokes by the proposed



(a) Cube

(b) Garden lantern

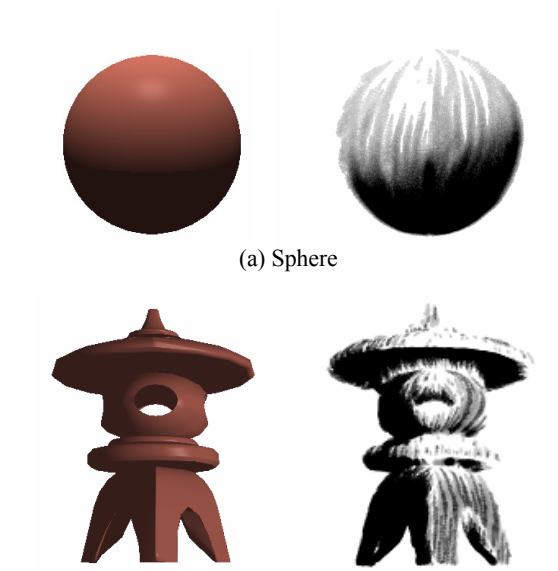
Fig.16 Examples of rendered outline strokes



(a) Sphere

(b) Garden lantern

Fig.17 Examples of rendered *Ten* strokes in shaded areas



(a) Sphere

(b) Garden lantern

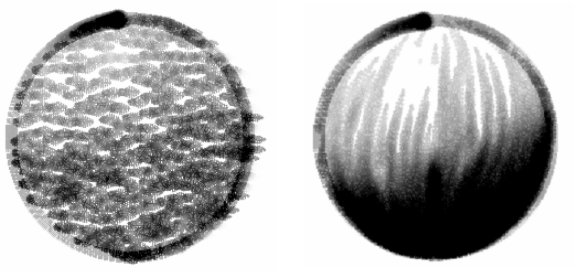
Fig.18 Examples of rendered *Shun* strokes in shaded areas

method. Unfortunately, this right image looks like a *Hakubyoho*-like image with clear outlines, and the *Shun* strokes drawn on the tree surface give almost no effect. This is caused by the reason that the areas surrounded with the outline strokes are too small to shade with *Shun* strokes.

Figure 22 shows images generated from a polygonal model of mountain in the same way as Fig.21. (a) and (b) show images of the same model rendered from different viewpoints. In each case, the *Shun* strokes flowing appropriately along the slope of each mountain side result in creating a *Suibokuga*-like impression. This means that the proposed method can create an appropriate *Suibokuga*-like image of a given geometric model from an arbitrary viewpoint by generating appropriate strokes from the model. Besides, comparing the left and right images shows us that the light-source information is effective.

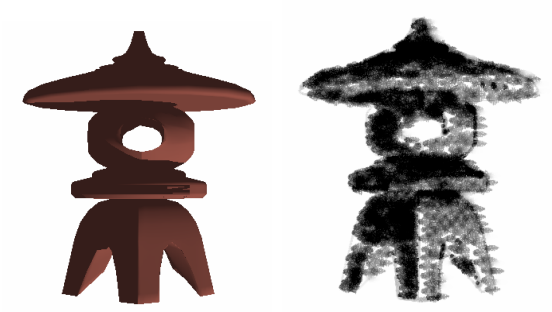
Figure 23 shows images of a scene containing two objects: a mountain and a tree. An actual *Suibokuga* painting drawn by *Sensenbyoho* is sometimes drawn by the way that the overall image is first drawn in thinner ink and then thicker ink is overlaid to emphasize objects and perspective effect. The image (c) in Fig. 23 is generated in such a way. First, an initial image was generated by the usual way. Then, the initial image was processed again by reducing water particles, increasing ink particles, and adjusting parameters M_w and A_Z in Eq.(2) for *Shun* strokes so that scaling factors M did not differ so much according to depth values Z_{ij}. This process added denser and finer *Shun* strokes on dark areas of the mountain surfaces and resulted in increasing light and dark contrasts. Here, we find that the image (c) is not given the same brightness distribution as (a). For example, the mountain surface part on the right of the tree is not so dark in (a), but is dark in (c). This was caused by the reason that surface parts near the viewpoint were given thicker and denser strokes (cf. Section 4.5). In this example, the effect of these strokes was so dominant that the original brightness distribution in (a) was not reflected on (c) so well. On the other hand, the top part of the mountain drawn in the middle of the image is so thin in (c), while the left side of this top part has dark intensity in (a). This was caused by Condition 2 of Step 3-1 described in Section 4.4.2. When positioning the next node following the current node on the screen in Step 3-1, the depth values registered with the pixels on which these nodes are placed tend to have a great difference if the object part is far from the viewpoint. In this case, Condition 2 is easily satisfied, and the growth of the stroke is finished. In Fig.23, the top part above is so far from the viewpoint. Therefore, for most starting nodes in this part, even the second nodes following them satisfied Condition 2, which resulted in generating no strokes.

All the images presented in this paper were generated using an SGI Origin2000 (R10000, 250 MHz, 4



(a) Outline strokes and *Ten* strokes (b) Outline strokes and *Shun* strokes

Fig.19 Examples of rendered images by merging outline strokes and shaded area strokes (sphere)

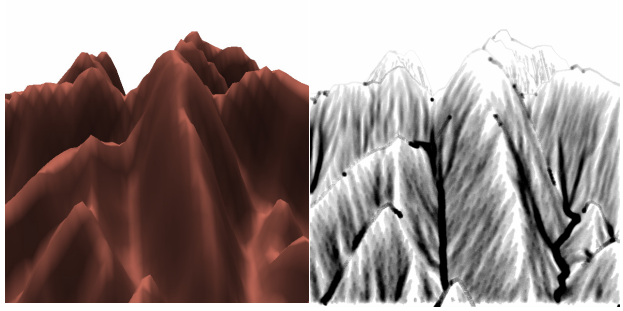


(a) Outline strokes and *Ten* strokes

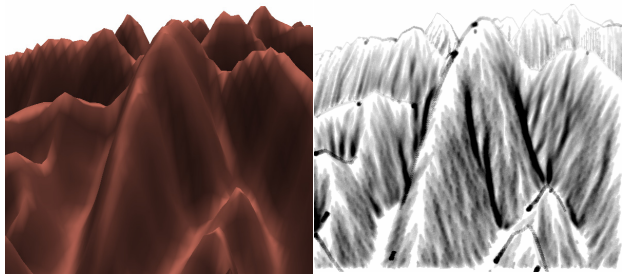
(b) Outline strokes and *Shun* strokes

Fig.20 Examples of rendered images by merging outline strokes and shaded area strokes (garden lantern)

Fig.21 Images generated from a polygonal model of tree



(a) A mountain scene



(b) Another mountain scene rendered from a different viewpoint from that of (a)

Fig.22 Images generated from a polygonal model of mountain

CPUs) system. Table 1 shows the number of strokes and calculation time required for generating the images of Fig.19 (a) and (b). This result of these simple example cases helps readers easily understand the performance of the proposed method.

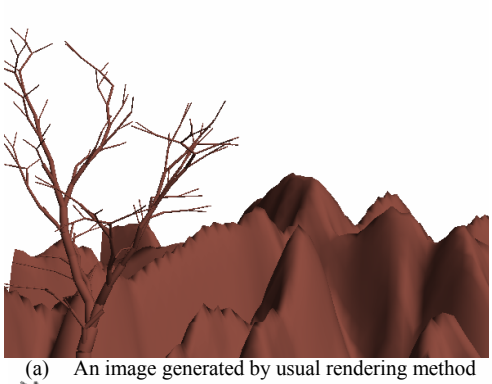
6. Conclusions

In this paper, we have proposed a method for generating *Suibokuga*-like images from three-dimensional geometric models such as polygonal models. This method can generate a *Suibokuga*-like image of arbitrary objects automatically as long as pixel data (depth values, intensity values, and normal vectors) can be obtained. The image can be generated from arbitrary viewpoints and affected by the light-source information properly. The proposed methods for generating outline strokes (*Kou* strokes) and shaded area strokes (*Ten* and *Shun* strokes) make it possible to create *Sensenbyoho*-like images, which are often used in landscape paintings. The effectiveness of the proposed method has been demonstrated through several examples of generated images.

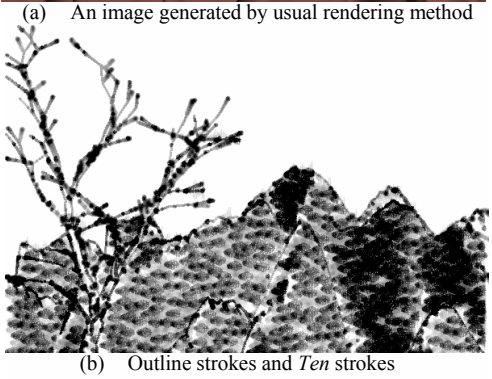
Future areas for study include the following.

- We should improve the stroke generation methods. For example, *Ten* and *Shun* strokes should be given appropriate length according to depth values.
- A method capable of selecting appropriate stroke techniques according to the features of different objects and realizing various drawing techniques is desired. For example, this method can generate an image in which trees are drawn in *Mokkotsuho* style and mountains are drawn in *Sensenbyoho* style.
- A method that creates deformed stroke shapes representing the feature of an object appropriately is useful.
- A method to generate strokes for more sophisticated representation with characteristic structural distribution patterns of strokes and usage of brushes according to artists should be realized.

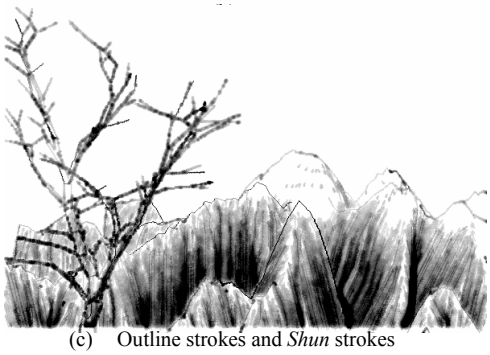
In addition, we are planning to develop a technique for reducing calculation time and a system with user interface.



(a) An image generated by usual rendering method



(b) Outline strokes and *Ten* strokes



(c) Outline strokes and *Shun* strokes

Fig.23 Images of a scene consisting of two objects, tree and mountain

Table 1 Number of strokes and calculation time of Fig.19

	Number of strokes	Calculation time (sec)
Fig.19 (a)	387	7300.5
Fig.19 (b)	306	2688.9

Acknowledgement

This work was supported in part by "A Support System for Region-specific R&D Activities" of National Institute of Information and Communications Technology of Japan.

References

[1] Bruce Gooch and Amy Gooch, Non-photorealistic Rendering, A K peters, Ltd, 2001.

[2] Stylized Depiction in Computer Graphics, http://www.red3d.com/cwr/npr/

[3] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin, Orientable Textures for Image-Based Pen-and-Ink Illustration, In proceedings of SIGGRAPH 97, pp. 401-406, 1997.

[4] Aaron Hertzmann, Painterly rendering with curved brush strokes of multiple sizes, In proceedings of SIGGRAPH 98, pp. 453-460, 1998.

[5] Doug DeCarlo, Anthony Santella, Stylization and Abstraction of Photographs, In proceedings of SIGGRAPH 2002, pp. 769-776, 2002.

[6] Cassidy J. Curtis, Sean E.Anderson Joshua E. Seims, Kurt W. Fleischer, David H. Salesin, computer-Generated Watercolor, In proceedings of SIGGRAPH 97, pp. 401-406, 1997.

[7] Piranesi, http://www.informatix.co.uk/piranesi/index.shtml

[8] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb John F. Hughes Adam Finkelstein, WYSIWYG NPR: Drawing Strokes Directly on 3D Models, In proceedings of SIGGRAPH 2002, pp. 755-762, 2002.

[9] Oliver Deussen, Thomas Strothotte, Computer-Generated Pen-and-Ink Illustration of Trees, In proceedings of SIGGRAPH 2000, pp. 13-18, 2000.

[10] Aaron Hertzmann, Denis Zorin, Illustrating smooth surfaces, In proceedings of SIGGRAPH 2000, pp. 517-526, 2000.

[11] Qing Zhang, Y. Sato, J. Takahashi, K. Muraoka and N. Chiba, Simple Cellular-Automaton-based Simulation of Ink Behavior and Its Application to *Suibokuga*-like Rendering of Trees, The Journal of Visualization and Animation, pp. 27-37, 1999.

[12] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein and John F. Hughes, Real-Time Nonphotorealistic Rendering, In proceedings of SIGGRAPH 97, pp. 415-420, 1997.

[13] Atsushi Kobayashi, Toshiyuki Oyama, Koji Nakamaru, Yoshio Ohno, Various Expressions of Silhouette Strokes for 3D CG, The 19th NICOGRAPH, pp. 27-32, 2003, in Japanese.

[14] Steven Strassmann, Hairy brushes, In proceedings of SIGGRAPH 86, pp. 225-232, 1986.

[15] David Small, Simulating watercolor by modeling diffusion, pigment, and paper fibers, In proceedings of SPIE, pp. 140-146, 1991.

[16] Qinglian Guo and Toshiyasu L. Kunii, Modeling the diffuse painting of Sumi-e, In IFIP Modeling in Computer-Graphics, pp. 329-338, 1991.

[17] Qinglian Guo, Generating Realistic Calligraphy Words, IEICE, E78A, pp. 1556-1558, November 1996.

[18] Helena T. F. Wong, Horace HS Ip, "Virtual Brush: A Model-Based Synthesis of Chinese Calligraphy", Computers & Graphics, Vol. 24, pp. 99-113, February 2000.

[19] Suguru Saito and Masayuki Nakajima, 3D physically based brush model for painting, SIGGRAPH99 Conference Abstracts and Applications, page 226, 1999.

[20] Jintae Lee, Diffusion rendering of black ink paintings using new paper and ink models, Computers & Graphics, pp. 295-308, April 2001.

[21] Ching (Clara) Chen, Ergun Akleman, Jianer Chen, Two Methods for Creating Chinese Painting, In Proceedings of 10 th Pacific Conference on Computer Graphics and Applications (PG'02), pp. 403-412, 2002.

[22] Jun-Wei Yeh and Ming Ouhyoung, Non-Photorealistic Rendering in Chinese Painting of Animals, In Proceedings of ChinaGraph2002, 2002.
[23] Zhen-Chung Shih, The Synthesis of Chinese Ink Paintings, In Proceedings of NICOGRAPH International 2002, pp. 11-20, 2002.
[24] Keiji Kawasaki and Yoshio Ohno, “Suiboku-ga”-Style Rendering for Polygon Data Sets, The 18th NICOGRAPH, pp. 121-126, 2002, in Japanese.
[25] Ryoukuu Shiba, Beginner’s *Suibokuga*, publishing company Seibi Shuppan, 1993, the Textbook of the *Suibokuga* in Japanese.
[26] Ko Ito, How to Draw *Suibokuga*, publishing company Goto Shoin,1993, the Textbook of the *Suibokuga* in Japanese.
[27] Invitation to the *Suibokuga*, Japan Broadcast Publishing, 1995, the Textbook of the *Suibokuga* in Japanese.
[28] Sumi-e No.8, japan publications, 1984, the Textbook of the *Suibokuga* in Japanese.
[29] Sumi-e No.31, japan publications, 1989, the Textbook of the *Suibokuga* in Japanese.