A Display Algorithm of Brush Strokes Using Bézier Functions

Tomoyuki Nishita, Shinichi Takita, and Eihachiro Nakamae

ABSTRACT

Graphics editors have recently come into wide use. But for displaying high quality images, a more powerful tool has been desired. This paper proposes a useful display method for Chinese calligraphy, traditional Japanese ink painting called *sumie*, and watercolor painting. The method comprises techniques to express the outlines of a brush stroke and to vary shades of color. That is, the outlines of a brush stroke are modeled using piecewise Bézier curves, and the variation of gray shade inside of the outline are defined by Bézier functions. This method provides effective characteristics of a brush stroke such as shade variation, the scratchiness produced by dry brush, and blotchiness caused by the diffusion of ink.

Keywords: Outline fonts, Bézier curves, Brush strokes, Chinese calligraphy, Scan conversion

1 INTRODUCTION

Graphics editors (painting systems) useful in graphic art have recently come into wide use. Due to the increase in users of such systems, a tool able to render such expressions as oil painting and airbrush work has been desired, in order to represent better quality images. This paper proposes a useful method of express the Chinese calligraphy, traditional Japanese ink painting called *sumie*, and watercolor painting. This method provides better quality compared with previous methods which attempt to render such effects as shade variation, scratchiness (produced by a brush with too little ink remaining), blotchiness caused by diffusion of ink, or a stroke with texture; It is made up of the techniques of scan converting an outline of a brush and shade variation; the outline of a brush stroke is described by Bézier curves, the shade variation in the inside of the outline is expressed as Bézier functions, and these regions are scan converted with high accuracy.

In the following sections, the effects of brush strokes, previous work, and the scan conversion method of brush strokes are described. Finally examples of results which prove the usefulness of the proposed method are presented.

2 EFFECTS OF BRUSH STROKES AND PREVIOUS WORK

2.1 Effects of Brush Strokes

Two dimensional painting systems are divided into two types; the painting type giving color information to each pixel, and the drawing type specifying a primitive such as a circle, or a straight line. The representation method discussed here belongs to the latter; both outlines of a stroke and shade variation are described by functions.

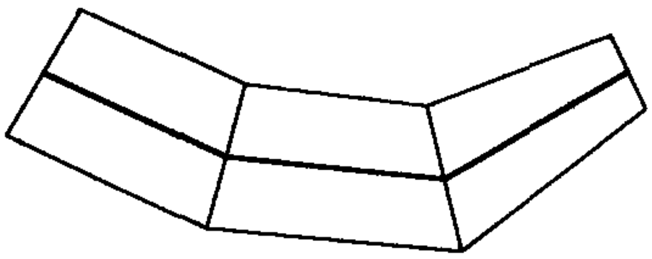


Figure 1: Expression of a brush stroke by quadrilateral approximation.

Sumie can produce not only the shape of curves but also perspective images by using subtle shade variation of ink. The ink density on the brush is uneven. Therefore, the ink density on each part of the brush affects the distribution of shade in a stroke, and the ink quantity varies as the brush moves through the stroke. This produces the effect of fuzzy shade gradation.

When a brush with watery ink moves on absorbent paper, particles of ink diffuse into the paper. This phenomenon is called blotchiness. Meanwhile, when the ink quantity is too little, a part of a brush stroke does not come out clearly. This phenomenon is called scratchiness (dry brush effect). With traditional Chinese calligraphy, depending on the particular character, the brush must be swept up abruptly at the end of the stroke. Such a movement of a brush also gives rise to scratchiness. Change of brush pressure varies the shade and width of a stroke. When drawing *sumie*, the variation of shade has to be taken into consideration. When rendering "Bokusai-painting" (traditional Chinese color painting), variation of color must be considered. Furthermore, outlines of almost all fonts resembling a writing brush style consist of curves. Therefore, a precise scan-conversion method of the closed areas bounded by curves is required.

2.2 Previous Methods for Brush Strokes

Some methods for displaying brush strokes have been developed. Strassmann [11] first attempted to simulate *sumie*. His method uses spline curves to express the trajectory of a stroke, and the area covered by the stroke is approximated by a set of quadrilaterals (see Figure 1). Chua [1] developed the method expressing the outline of a stroke by Bézier curves. This method uses a PostScript equipped printer. As stripes bounded by Bézier curves of a small width with different intensities are arranged to represent shade variation, shade varies discretely. Therefore, the quality of the image produced is not high enough. The effect of blotchiness is not considered in his system. Pham [9] used a B-spline curve to express the trajectory of a brush stroke. The area covered by the stroke is approximated by a set of small quadrilaterals to be filled with ink. Recently a method to simulate the blotchiness caused by the diffusion of ink using microscopic property of paper was proposed by Guo [4], but the area is also approximated by a set of quadrilaterals. As most methods use a set of quadrilaterals as an approximation of the area bounded by curves, the image quality is still open to improvement. The proposed method describing outlines of strokes and shade variation by Bézier functions yields precise and smooth outlines and variation of shade.

The method proposes a scan-conversion method for curved outlines without any polygonal approximation. As direct scan-conversion algorithms of curves, both of the scan-conversion of quadratic splines [8] and the quadratic rational Bézier curve [10] have been developed. In these methods intersections of a scanline and curves are calculated by solving a quadratic equation; intersection points can be obtained analytically. As these methods use quadratic curves, image quality is unsatisfactory. On the other hand, it is difficult to obtain intersections

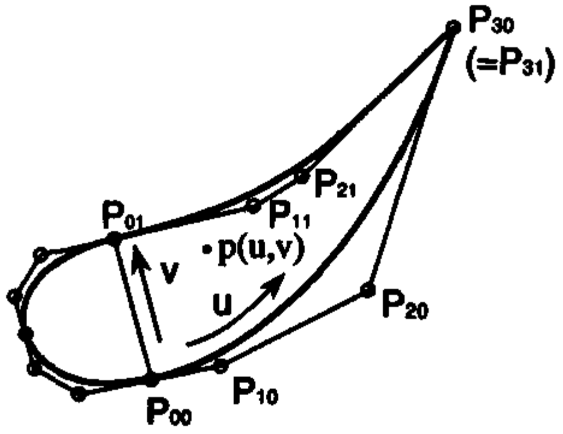


Figure 2: Expression of a brush stroke by Bézie curves.

analytically for curves with a degree of more than two. The Newton method is often used as one of the numerical analysis methods; but it requires a suitable initial guess, and is not robust. It is difficult to guarantee finding all solutions. As far as the authors know, no direct scan-conversion algorithm of curves, which have a degree of more than two, has been published. The advantages of the proposed method are that the intersections of a scanline and a curve can be obtained robustly by employing the convex hull property of Bézier curves, and the high degree Bézier curves are scan converted by iterations using linear equations.

The system which controls the movement of an actual writing brush by a computer like a plotter was proposed [12], though this subject is out of our discussion.

3 BASIC CONCEPT AND OUTLINE OF PROCEDURE

A couple of cubic Bézier curves make up the outline of a stroke. Let's consider Bézier planar patches bounded by curves (see Figure 2). The degree of the Bézier patch is 3 × 1, and each patch is expressed by two parameters, u and v; u is the parameter along it, and v the parameter across it. A shade value is defined as Bézier functions of u and v. Variation of shade with respect to v is defined as a cubic Bézier function because the distribution of ink across a stroke is uneven.

The outline of a brush stroke is given as Bézier curves of degree three or degree one (i.e., a straight line). A point in the the Bézier patch is denoted by $P(u,v)$, and the shade value at that point is described by function of (u,v).

As a brush stroke is represented by a planar Bézier patch (3 × 1, or 1 × 1 degree Bézier patch), point P in the patch is expressed by

$$P(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{1} P_{ij} B_i^n(u) B_j^1(v) \tag{1}$$

where $P_{ij}(x_{ij}, y_{ij})$ $(i = 0, 1, 2, \cdots, n; n = 3 \text{ or } 1, j = 1, 2)$ are the coordinates of the control points, and both curves on $v = 0$ and $v = 1$ give the boundary curves of a stroke. B is the Bernstein polynomial, and is given by $B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$.

The outline of the procedure is as follows:

(1) A scanline moves from the top to the bottom, and every intersection of the scanline and outline curves is calculated.

(2) For every pixel between these intersections, u and v are evaluated.

(3) The shade is computed using u and v, and is displayed.

In Step (2), in the case of 3×1 degree patch, an equation of degree 6 should be solved to calculate parameters u and v from coordinate (x, y) on a screen (when we use Kajiya's raytracing method[5]). Now we are discussing a painting system, and it is not practical, due to computational expense, to solve such a high degree equation for every pixel. Therefore, we propose an approximation method based on linear equations. That is, the values of u and v for sampling points on the scanline within the patch are obtained as the intersections of iso-parametric curves of the patch (iso-parametric curves of v component are lines) and the scanline.

Data input is performed interactively. Two input methods are available in our system. One of them uses points on the outline of brush stroke, some points on the outline are taken into a computer by mouse, and then Bézier curves are constructed using the data. Another method uses a sequence of points on a center line of a stroke; after inputting these points on the center line, Bézier curves expressing the center line are constructed, then widths are given as offset distances from the control points, and the offset curves are constructed by Bézier curves.

4 SCAN CONVERSION OF BÉZIER PATCHES

4.1 Calculation of Intersections of a Scanline and Curves

The scan-conversion algorithm proposed here is the improved version of calculation of intersections of Bézier curves and a straight line, which has been developed for the ray tracing method [6] of Bézier patches to save on computation time.

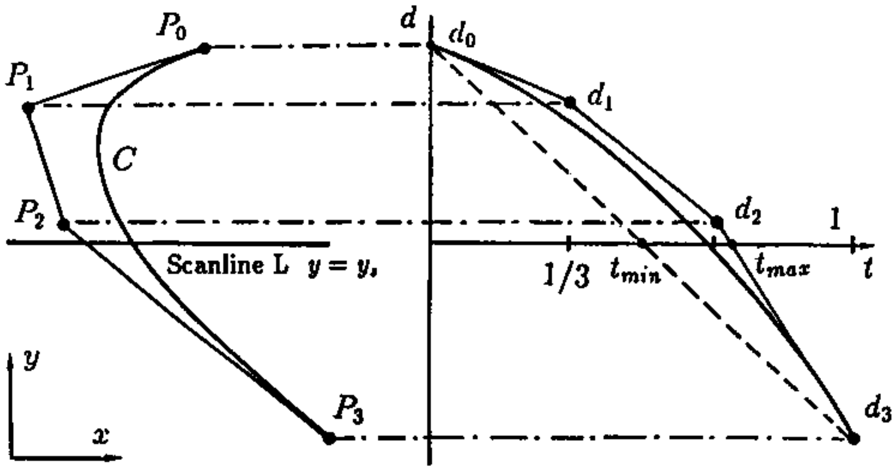
To obtain intersection points efficiently the following premises are made. 1)The scanline moves from the top to the bottom, and 2) the curve is monotone decreasing in y component. That is, the curve intersects with the scanline only once. The preprocessing for setting this condition is performed in advance; an original Bézier curve is subdivided at every point where the first derivative is zero.

The algorithm of calculating an intersection point between Bézier curve C of degree n and scanline L is explained using Figure 3 ($n = 3$ in this figure). Coordinates (x, y) of an arbitrary point on C are expressed by using parameter t as follows(e.g., $v = 0$ in equation (2)):

$$\begin{aligned} x(t) &= \sum_{i=0}^{n} x_i B_i^n(t), \\ y(t) &= \sum_{i=0}^{n} y_i B_i^n(t). \end{aligned} \tag{2}$$

Let's denote y_s as y coordinate of the scanline, and then the equation of scanline L is expressed by $y - y_s = 0$. The intersection between curve C and straight line L is obtained by substituting y-component of equation (2) in this equation,

$$\sum_{i=0}^{n} y_i B_i^n(t) - y_s = 0. \tag{3}$$



(a) Bezier Curve. (b) Distance function.

Figure 3: Intersection between scanline and Bézier curve.

As $\sum_{i=0}^{n} y_s B_i^n(t) = y_s$,

$$\sum_{i=0}^{n} d_i B_i^n(t) = 0. \tag{4}$$

where $d_i = y_i - y_s$.

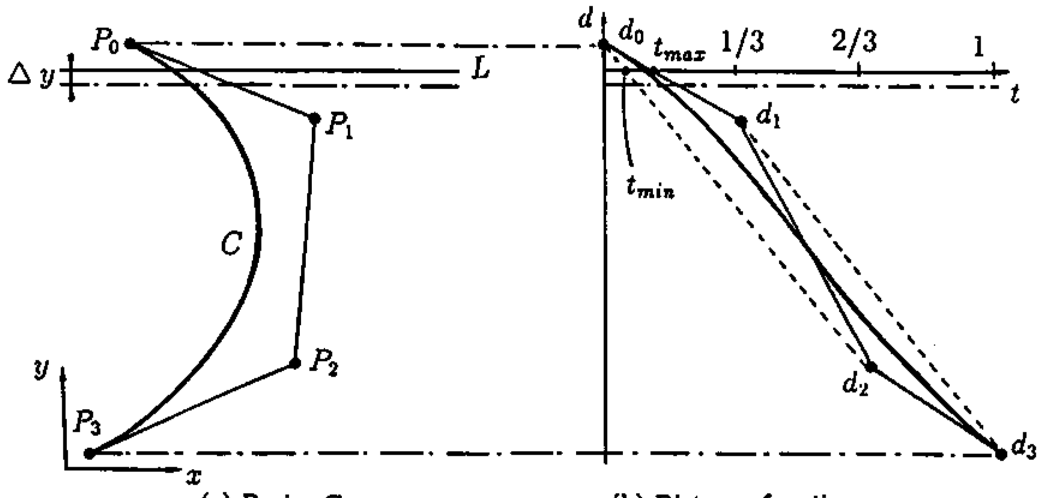
As shown in Figure 3 (b), equation (4) is equivalent to a non-parametric Bézier curve. Assumed that the function composed of control point $(i/n, d_i)$ is $d(t)$, d is the distance from the scanline to the Bézier curve. Therefore we call equation (4) a distant function. The solutions of equation (4) is obtained by an iteration method (*Bézier clipping* method[6]).

4.2 Scan Conversion Using Coherence between Scanlines

For saving computation time on the iterative method described in the previous sub-section, the coherence between scanlines is utilized. Assume that scanline L decreases by Δy (here $\Delta y = 1$; scanline width). The value of parameter t of curve C is nearly equal to zero at the intersection point of the curve C with the first scanline (i.e., the intersection point is near the starting point P_0 on C). In most cases the intersection interval $[t_{min}, t_{max}]$ between a scanline and the convex hull formed by control points d_is of a distance function is narrow enough (see Figure 4(b)); the solution can be obtained by few iterations. After converging to solution t_{min}, the curve is subdivided at t_{min}; the interval $[0, t_{min}]$ is clipped away, and a new interval $[t_{min}, 1]$ is used for the next scanline as new control points. Then the control points for interval $[t_{min}, 1]$ are used for the next scanline as new control points.

By splitting the curve like this, every intersection point between a new interval and the following(next) scanline (a horizontal chain line shown in the figure) is near the point where $t = 0$ on the new curve. Therefore, the solution is obtained by few iterations. If the difference of t_{min} from that of previous scanline is very small, the Bézier curve is not subdivided as mentioned later. The subdivision of Bézier curves is applied only to y component, rather than to both x and y components. Subdivision is accomplished by the well-known de Casteljau method[2].

Let's denote the minimum and maximum y values of the control points as y_{min} and y_{max}, and the values of t_{min} and t_{max} on the original(i.e., before subdivision) Bézier curve as T_{min}



(a) Bezier Curve. (b) Distance function.

Figure 4: Intersection interval on a scanline.

and T_{max}, respectively. Let *tol* be the tolerance. Then the procedure of scan conversion is as follows:

1) Set $y_s = y_{max}$, $T'_{min} = 0$.

2) Extract the interval $[t_{min}, t_{max}]$ using the control points d_i of the distance function, where $d_i = y_i - y_s$ $(i = 1, 2, \cdots, n)$.

3) If $T_{max} - T_{min} > tol$, extract the distance function of the interval $[t_{min}, t_{max}]$ and go back to 2). Otherwise go to the next step.

4) Calculate $t = (T_{min} + T_{max})/2$, and calculate x coordinate from t(see equation (2)).

5) If $T_{min} - T'_{min} > tol$, extract the interval $[t_{min}, 1]$ on y component of the Bézier curve, and set $T'_{min} = T_{min}$. Otherwise go to the next step.

6) If $y_s > y_{min}$, set $y_s = y_s - \Delta y$ and return to 2). Otherwise terminate the algorithm.

As the tolerance is defined in parameter space, the value of the tolerance has to be given depending on the length of the curve; in order to obtain the same degree of accuracy of the x coordinate of an intersection point, the tolerance may be small for a long curve, and large for a short curve. Therefore, in this paper the inverse of the length of the longer side of the bounding box determined by the control points of the curve is used as the tolerance. Subdivision in step 3) is applied twice, since the left side of t_{min} and the right side of t_{max} should be clipped away. Meanwhile, in step 5) to clip the left side of t_{min} the curve is subdivided once.

In the proposed method the smaller the area of the convex hull composed of the control points of the distance function, the fewer the number of iterations. Namely, as the intervals between y components of the control points approach equidistance, the distance function comes close to a straight line and thus the number of iteration becomes few. For our examples, the proposed method requires only an average of 1.3 iterations to obtain one intersection.

4.3 Inverse Mapping

As stated in the previous sub-section, the intersection points of a scanline and boundary curves can be extracted efficiently by the intersection test using the Bézier clipping method.

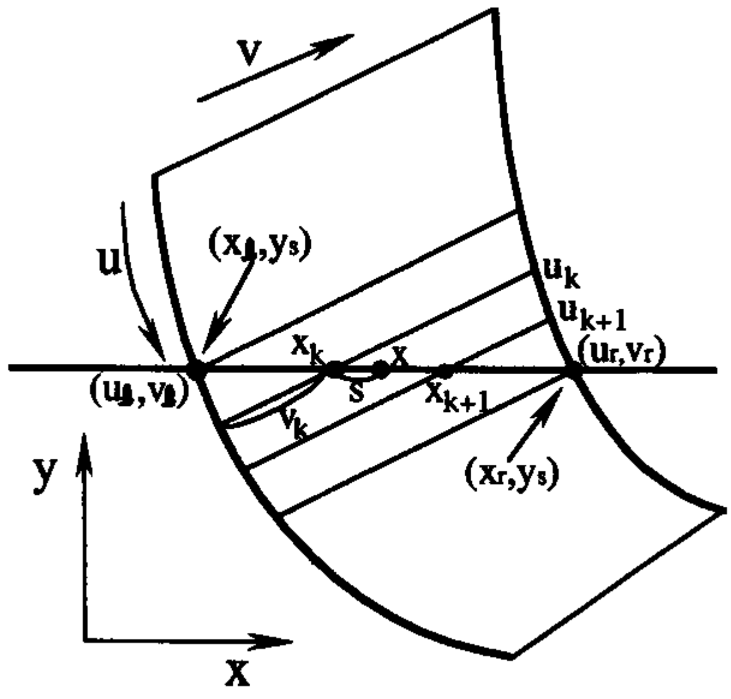


Figure 5: Inverse mapping.

In the next step in order to fill the Bézier patch, inverse mapping calculating values of u and v from coordinate (x, y) is required. For this calculation, as mentioned before, it is necessary to solve equations of degree six. Though the Bézier clipping method .can be used, the following approximation is applied to improve the efficiency.

The intersection points of scanline $y = y_s$ and boundary curves of a Bézier patch are denoted by (x_l, y_s) and (x_r, y_s), and parameters at those points are described by (u_l, v_l) and (u_r, v_r), respectively. After calculation of intersection points (x_l, y_s) and (x_r, y_s) (see Figure 5), (u, v) of every pixel between the intersections are evaluated in the following manner.

As the degree of v component is one, it is guaranteed that an intersection line of the scanline and the Bézier patch is within $[u_l, u_r]$ in the parameter space.

Sampling points are set by dividing the interval $[x_l, x_r]$ by N, where $N = (x_r - x_l)/M$; M is the sampling span (usually it's set to 2 or 3 pixel width). After calculation of (u, v) for every sampling point, the values of the parameters at every pixel between the sampling points are linearly interpolated.

At the k-th sampling point ($k = 1, 2, \cdots, N-1$), let (u_k, v_k) denote parameters and x_k express the x-coordinates at the intersections between the scanline and the outline. Then these values are obtained as follows:

1) If $u_r - u_l \geq \epsilon$ (ϵ : *tolerance*) ,

$$\begin{aligned} u_k &= u_l + (u_r - u_l)k/N, \\ v_k &= (y(u_k, 0) - y_s)/(y(u_k, 0) - y(u_k, 1)), \\ x_k &= x(u_k, v_k), \end{aligned}$$

where functions $x(u, v)$ and $y(u, v)$ are x and y components of the equation, respectively (1). The following equations give the values of (u, v) satisfying the condition $x_k \leq x <$

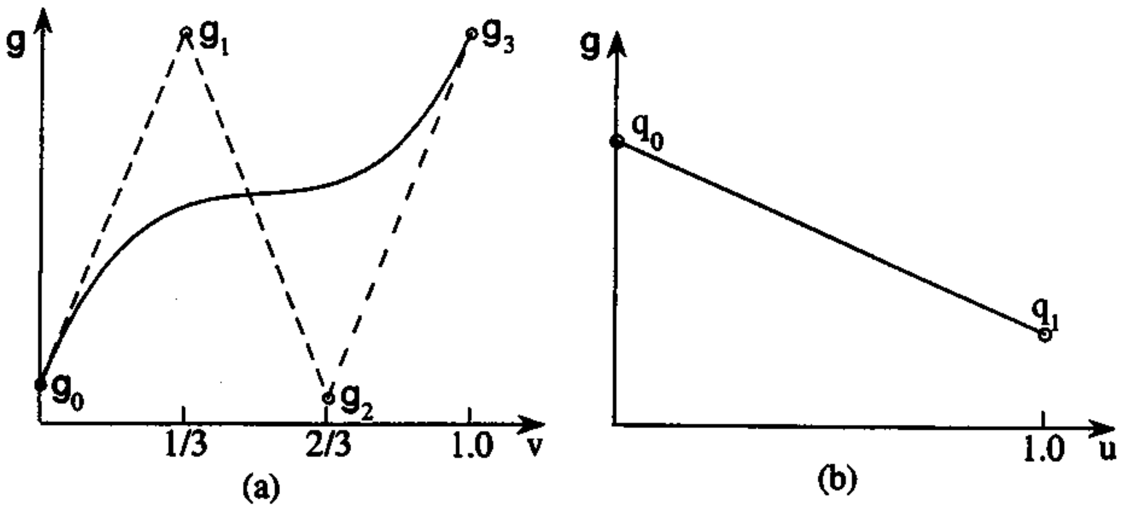


Figure 6: Shade variation by Bézie function.

x_{k+1}.

$$\begin{aligned} s &= (x - x_k)/(x_{k+1} - x_k), \\ u &= u_k + (u_{k+1} - u_k)s, \\ v &= v_k + (v_{k+1} - v_k)s. \end{aligned} \tag{5}$$

2) If $u_r - u_l < \epsilon$,

$$\begin{aligned} s &= (x - x_l)/(x_r - x_l), \\ u &= (u_l + u_r)/2, \\ v &= v_l + (v_r - v_l)s. \end{aligned} \tag{6}$$

5 CALCULATION OF SHADE VARIATIONS

Shade of a brush stroke is given as Bézier functions of u and v (u is the parameter along the stroke and v is the one across the stroke). Shade function g expressed by u and v is shown in Figure 6. This function shows the blending ratio between ink color and paper color. The shade function is defined as

$$g(u,v) = \sum_{j=0}^{3} g_j B_j^3(v) \sum_{i=0}^{1} q_i B_i^1(u) \tag{7}$$

where Bézier functions of u and v are linear and degree 3, respectively, and both g_j and q_i are the control points of Bézier functions for shade variation. It is assumed that the ink decreases linearly along a stroke (with u component), and the distribution of ink across the stroke is not uniform. For example, in case that ink is dense in the center part and thin in the margin, the degree of function should be more than two. Therefore, we use a cubic Bézier function for v component.

Color(or intensity) C at point (x, y) is obtained from ink color C_i and paper color C_p, using shade function g; it is calculated by

$$C(x,y) = g(u,v)C_i + (1 - g(u,v))C_p(x,y). \tag{8}$$

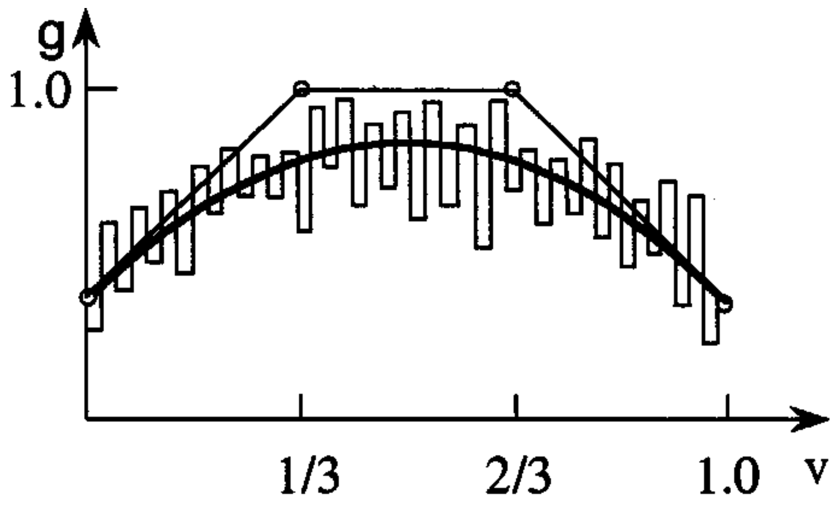


Figure 7: Shade function with dry brush effect.

By treating the color of the colored paper as the color of new paper, it is possible to overlap another stroke.

5.1 Effect of Dry Brush

When the moisture on a brush is too little, a part of a brush stroke does not come out clearly. That is, scratchiness arises; *dry brush* effect. Once this arises, it remains until the ink is replenished. To represent the quantity of each bristle, v component is discretized (e.g., divided into 50 elements). An array is used to memorize the ink quantity of each bristle. Variation of the ink quantity (v component) is given as a cubic Bézier function, and a small amount of variation caused by the application of random numbers is superimposed. Figure 7 shows initial ink quantity at $u = 0$ (User can specify the control points of the Bézier function and the the magnitude of the random numbers). Along the movement of a stroke (in proportion to u), the ink quantity decreases linearly. When the ink quantity of a bristle drops below a given threshold, its locus fades away.

5.2 Effect of Blotchiness

When a sheet of absorbent paper is used, blotchiness (diffusion or *nijimi* in Japanese) arises, and it is noticeable around the boundary of a stroke. The diffusion area is assigned as a function of v component in our system. That is, blotchiness arises in the regions of $0 \leq v < dv$ and $1 - dv < v \leq 1$, where 0.1 or 0.2 are used as dv. In order to vary shade in these regions, our system uses the Fourier function [3] which is proposed for generating cloud patterns.

The nearer the boundary, the greater the effect of blotchiness. The shade value is weighted by the following equation F which is shown in Figure 8.

$$F(u) = k\sum_{i=0}^{m} c_i(\sin(f_i u + p_i) + T) \cdot \sum_{i=0}^{m} c_i(\sin(g_i v_0 + q_i) + T), \qquad (9)$$

where constant k is used to keep the range of F smaller than 1, f_i and g_i are frequencies, c_i is the magnitude for the i-th frequency, and p_i and g_i are phases. Among those constants, there are following relations:

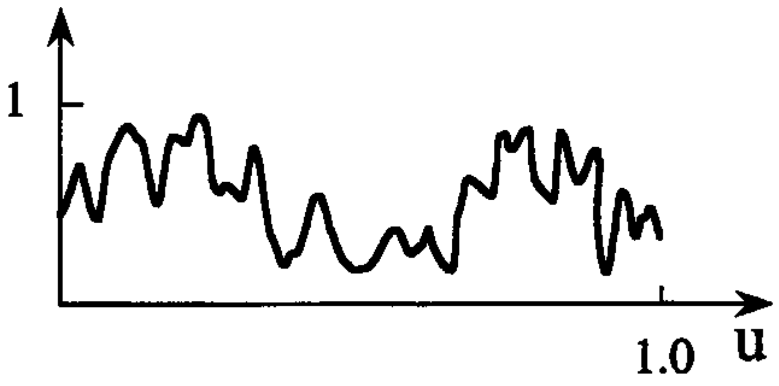


Figure 8: Shade variation by Fourier series.

$f_{i+1} = 2f_i$, $g_{i+1} = 2g_i$, $c_{i+1} = 1/\sqrt{2}c_i, p_i = (\pi/2)\sin(g_i + v_0/2)$ (q_i is given in the similar manner). m ranges from 3 to 6, and T is set to 0.5. v_0 is constant.

For example, when $v < dv$, if $v < Fdv$, the shade function g is set to 0. In the case of $1 - dv < v$, if $v > Fdv/(1 - dv)$, g is set to 0.

Figure 9 shows examples of simple brush strokes; (a) represents shade variation, (b) expresses the effect of dry brush, (c) shows blotchiness, and (d) shows overlapping strokes with different colors.

5.3 Texture Mapping

As (u, v) are calculated for every point in a patch, general texture mapping methods for 3D objects are applicable without any change.

6 EXAMPLES

Figure 10 shows some examples of the proposed method. Figure (a) shows an example of Chinese calligraphy, a character which means heart. Figure (b) demonstrate dry brush effect. Figure (c) represents the effect of blotchiness.

Examples of *sumie* are shown in Figure 11; leaves and branches of a sasanqua camellia branch with a flower is shown in Figure (a), a Japanese nightingale in (b), branches of a tinged Japanese maple tree in (c), and tulips in (d). The number of Bézier curves in these examples are, in order, 232, 212, 633, and 226, respectively. The stamp shown in the bottom left in Figure (a) was the scanned image, and the *sumie* was overlapped. Blotchiness appears in a flower in Figure (a), a part of the leaves in Figure (c), and the flowers of Figure (d). The effect of dry brush is depicted in the leaves in Figure (d). The texture mapped *sumie* is shown in Figure 12; curved surfaces are displayed by the raytracing method [6].

Antialiasing is performed by the multi-scanning method without polygonalization [7].

7 CONCLUSION

We have proposed a powerful display method for Chinese calligraphy, Japanese ink painting, and watercolor painting. The method is composed by the techniques of scan conversion for

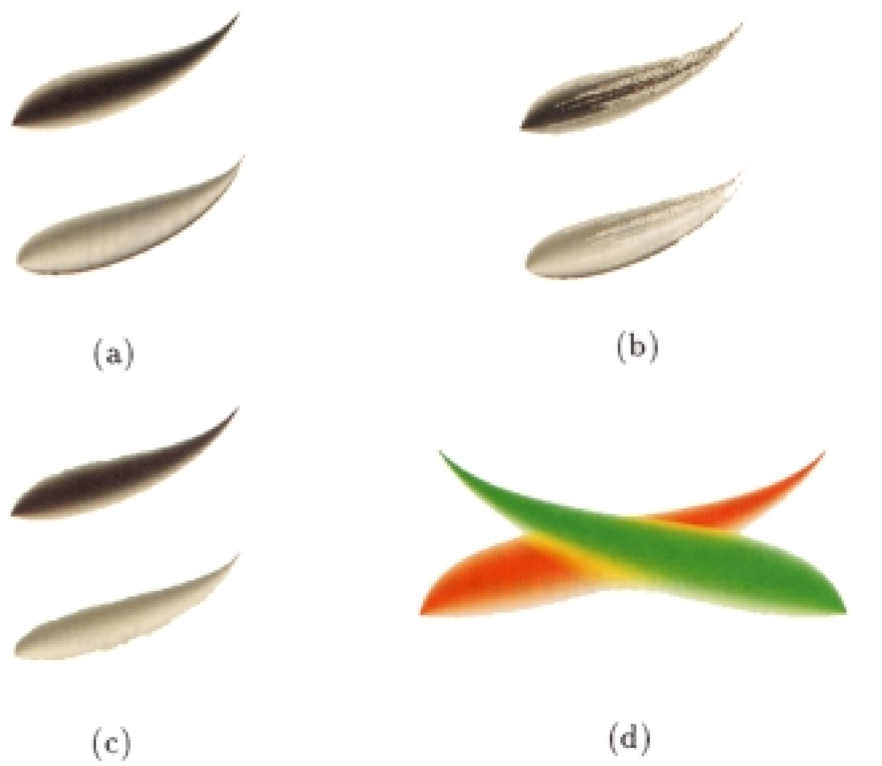


Figure 9: Examples of simple strokes.

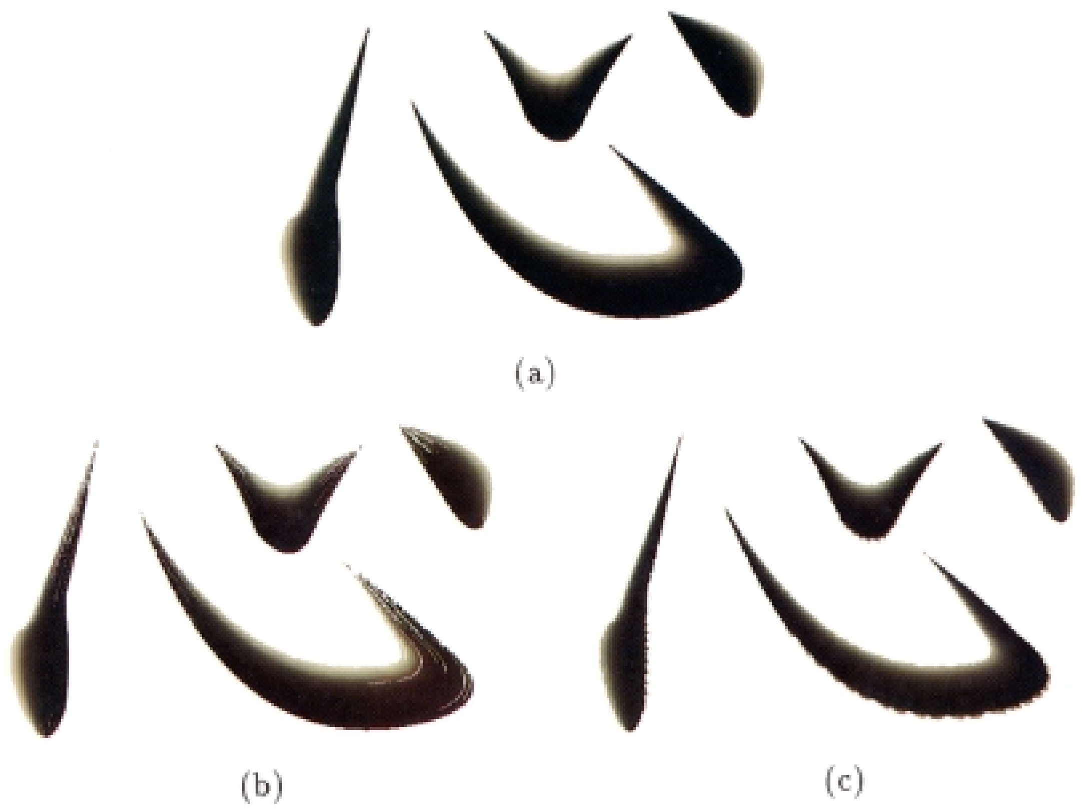


Figure 10: Examples of Chinese calligraphy.

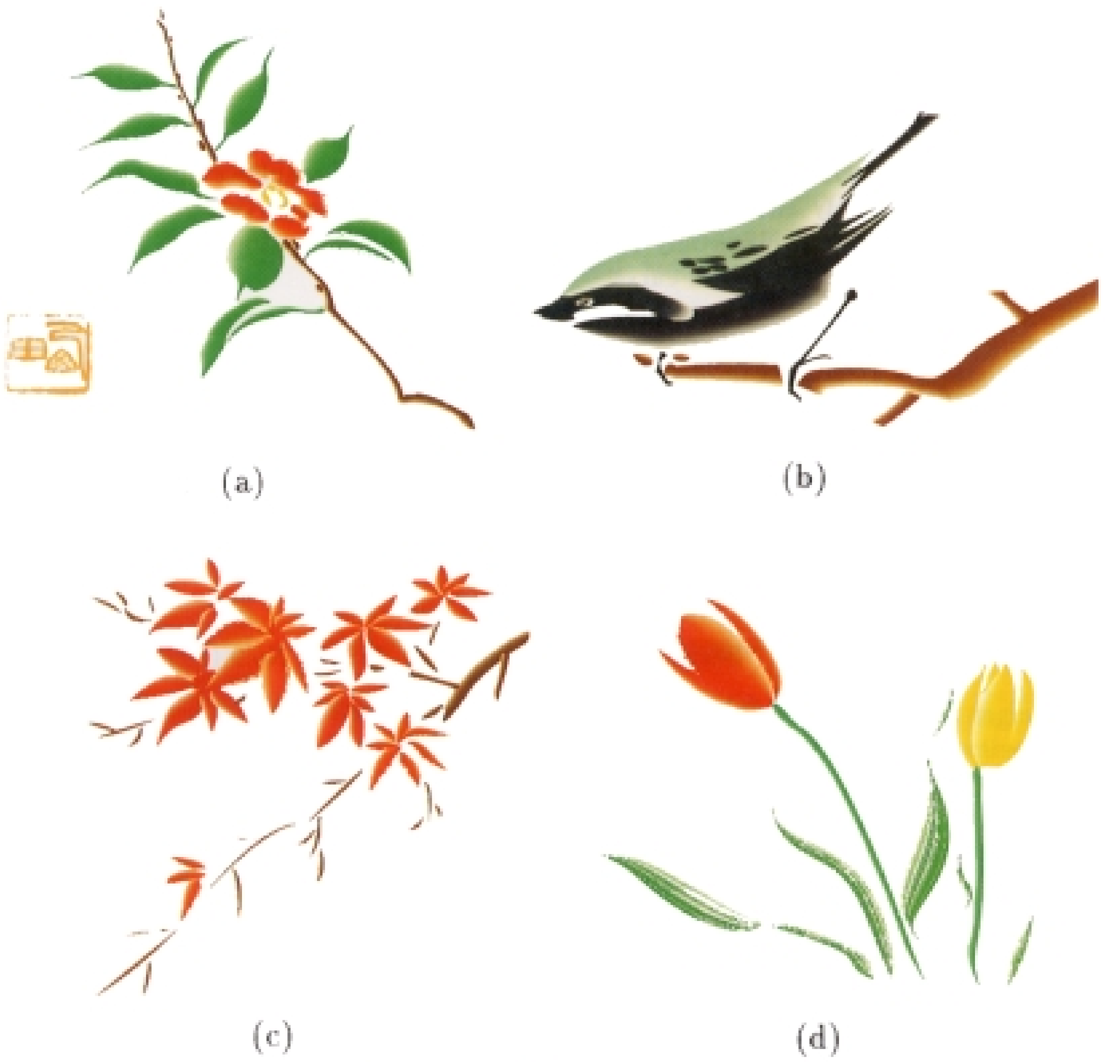


Figure 11: Examples of Japanese ink painting.

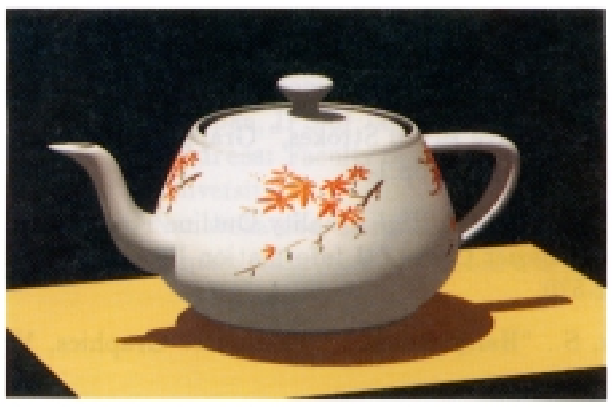


Figure 12: An example of texture mapped *sumie*.

an outline of a brush and varying shade; the outline of a brush stroke is described by Bézier curves, and shade variation inside the outline is defined by the Bézier function.

The advantages of the proposed method are as follows:

(1) Scan conversion is performed precisely because the algorithm is without any polygonal approximation of boundary curves. Intersections of scanlines and outlines can be obtained robustly and precisely using the convex hull property of Bézier curves; the proposed method requires only 1.3 times iteration on average to obtain one intersection of a scanline and the Bézier curve.
(2) Subtle shade variations such as fuzzy shade gradation, dry brush, and blotchiness can be displayed.
(3) High quality images can be displayed by performing antialiasing.

Acknowledgment The authors would like to acknowledge Y. Taniguchi and M. Nomura for their help in data creation.

REFERENCES

[1] Chua, Y.S., "Bézier Brushstrokes," CAD, Vol.22, No.9 (1990), pp.550-555.

[2] Farrin, G., "Curves and Surfaces for Computer Aided Geometric Design," Academic Press Inc., (1988), p.25-31.

[3] Gardner, G.W.,"Visual Simulation of Clouds," Computer Graphics, Vol.19, No.4,(1985), pp.229-303.

[4] Guo, Q., Kunii, T., "Modeling the Diffuse Painting of 'Sumie'," Modeling in Computer Graphics (Proc. of the IFIP WG5.10), Springer-Verlag(1991), pp.329-338.

[5] Kajiya, J., "Ray Tracing Parametric Patches," Computer Graphics, Vol.16, No.3,(1982), pp.245-254.

[6] Nishita, T., Sederberg, T.W., Kakimoto, M., "Ray Tracing Rational Trimmed Surface Patches," Computer Graphics, Vol.24, No.4,(1990), pp.337-345.

[7] Nishita, T., Nakamae, E., "Half-Tone Representation of 3D Objects with Smooth Edge by Using a Multi-Scanning Method,"J.Information Processing(in Japanese), Vol.25, No.5,(1984), pp.703-711.

[8] Pavlidis, T., "Scan Conversion of Regions Bounded by Parabolic Splines," IEEE CG & A, 1985,pp.47-53.

[9] Pahm, B.,"Expressive Brush Strokes," Graphical Models and Image Processing, Vol.53, No.1,(1991), pp.1-6.

[10] Saitoh, T., Hosaka, M., "High Quality Outline Fonts by the Extended Rational Quadratic Bézier Curve," J.Information Processing(in Japanese), Vol.31, No.4,pp.562-570.

[11] Strassmann, S., "Hairy Brushes," Computer Graphics, Vol.20, No.4,(1986), pp.225-232.

[12] Yun-Jie, P., Hui-Xiang, Z., "Drawing Chinese Traditional Painting by Computer," Modeling in Computer Graphics (Proc. of the IFIP WG5.10), Springer-Verlag(1991), pp.321-328.

Tomoyuki Nishita is a professor in the department of Electronic and Electrical Engineering at Fukuyama University, Japan. He was on the research staff at Mazda from 1973 to 1979 and worked on design and development of computer-controlled vehicle system. He joined Fukuyama University in 1979. He was an associate researcher in the Engineering Computer Graphics Laboratory at Brigham Young University from 1988 to the end of March, 1989. His research interests involve computer graphics including lighting model, hidden-surface removal, and antialiasing.
Nishita received his BE, ME and Ph. D in Engineering in 1971, 1973, and 1985, respectively, from Hiroshima University. He is a member of ACM, IPS of Japan and IEE of Japan.
Address: Faculty of Engineering, Fukuyama University, Sanzo, Higashimura-cho, Fukuyama, 729-02 Japan.
E-mail: nis@eml.hiroshima-u.ac.jp

Shinichi Takita is a professor in the Department of Education at Kagawa University, Japan. His research interests include computer graphics and CAI.
Takita received his BE and ME degrees in electrical engineering form Hiroshima University in 1964 and 1966, respectively. He is a member of the IEE of Japan, IPS of Japan and the Japan Society of Industrial and Technical Education.
Address: Faculty of Education, Kagawa University,
1-1, Saiwai-cho, Takamatsu, 760 Japan.
E-mail: takita@ed.kagawa-u.ac.jp

Eihachiro Nakamae is a professor at Hiroshima Prefectural University. Previously he worked at Hiroshima University from 1956 to 1992, where he was appointed as research associate in 1956 and a professor in 1968. He joined Hiroshima Prefectural University in 1992. He was an associate researcher at Clarkson College of Technology, Potsdam, N. Y., from 1973 to 1974. His research interests include computer graphics and electric machinery.
Nakamae received the BE, ME, and DE degrees in 1954, 1956, and 1967 from Waseda University. He is a member of IEEE, IEE of Japan, IPS of Japan and IEICE of Japan.
Address: Faculty of Information Science, Hiroshima Prefectural University,
Nanatuka-cho, Shoubara City, Hiroshima Prefecture, 727 Japan.
E-mail: naka@eml.hiroshima-u.ac.jp