Simulating Oriental Black-Ink Painting

Black-ink painting began as a traditional art form in Asia more than three thousand years ago. Black-ink painting uses four main tools, commonly referred to as the “four treasures”: the ink brush, inkstick, inkstone, and paper (Figure 1).

Oriental black-ink and Western watercolor painting resemble each other in that both use water and brushes. From the standpoint of technique and media, however, they differ significantly. In black-ink painting, the artist doesn’t care about producing an exact or literal reproduction, expressing only the essence of the considered object(s). In other words, paintings typically consist of just a few simple strokes intended to convey the artist’s “deep feelings” regarding the painted object. Simplicity is therefore a key concept. Feelings are expressed by the speed, placement, pressure, and movement of the brush, as well as by the resultant degree of shading produced by the brush strokes.

Attracted by such simple yet expressive beauty, numerous researchers have attempted to numerically model black-ink painting using computer simulations. They have employed methods generally classifiable as outline-based or flat-brush-based.

In the outline-based approach, the user describes the outline of brush strokes as a sequence of connected Bézier or B-spline curves developed using 2D drafting software in conjunction with mouse or pen input and scaling/editing control points. The strokes produced are filled with image patterns incorporating a set of shade variation effects. The user can modify them by reapplying the same 2D drafting and filling functions.

The outline-based approach gives precise, smooth outlines with variations in shading. However, the drawing process employed limits the mode of access, since stroke editing is restricted to altering knot points that determine stroke boundaries.

The flat-brush-based approach uses 2D brushes of various shapes, sizes, and patterns, including square, round, diagonal lines, dotted, or faded airbrush. The average person finds these brushes much easier to use than real paintbrushes. Many commercially available products use the flat-brush approach because brush strokes result from brush movement rather than boundary editing. Subsequent modification of strokes also becomes simpler using path-based commands.

A remarkable brush model of this type, developed by Strassmann, considers the brush a 1D array of bristles. The brush always moves perpendicular to the stroke path defined by a set of position and pressure parameters, where pressure determines stroke width.

While this approach produces brush strokes by brush movement, the brush model could still use improvement, especially with regard to black-ink painting. Specifically, since the brushes are simply 2D patterns like rubber stamps, users cannot see how the bristle shape transforms as bristles are pulled, turned, pressed down, or lifted up—important visual information to better control brush movement. Moreover, because such imaginary bristles do not bend, the timing of exerting pressure on the brush differs noticeably from that of a real brush. Hence, users cannot apply painting skills acquired over a lifetime.

The inherent drawbacks in both approaches led to the present study (an extension of previous work), which introduces the concept of “soft” brushes—3D brushes in which the bristle shape varies dynamically in response to the forces imposed on it by the paper. The interactive modeling algorithm for soft brushes lets users resize brushes and choose a particular color, texture, or type of bristles.

With this unique approach, a user produces brush strokes by brush movement, eliminating the unnatural spline curve editing required by the outline-based approach. Unlike the flat-brush-based approach, here 3D visual information comes from the elastic deformation of bristles on the paper—crucial information in black-ink painting.

Jintae Lee
University of Aizu
es, then at how to effectively apply them in black-ink painting.

**Oriental brush**

Both the shape and handling of the brush used in oriental black-ink painting differ from brushes used in Western watercolor painting. The bristles—made from the soft hair of animals such as rabbits, martins, badgers, horses, or deer—measure from 5 to 6 centimeters long and are starched to form a pointed tip. If they do not have the proper elasticity and shape, the brush cannot provide stroke lines with adequate quality.

Figure 2 shows the structure of the oriental brush. In the normal or “up” state, the whole bristle bundle (b-bundle) narrows to a pointed tip. The overall shape looks like an inverted cone. Consider the center of the base of the b-bundle as the origin of the coordinate system in which the y-axis is vertical. The elasticity of the bristles is considered ideal. When inverted with light pressure applied, the bristles should neither collapse to one side (too soft) nor feel too hard or springy, like a hairbrush. This ideal material and shape permits expressing all the nuances of nature’s colors through simple, delicate stroke lines, shades of black, and variations of ink diffusion.

When modeling an oriental brush, the designer should consider the following features:

- When drawing a line, as the pressure exerted on the brush increases, the elastic bristles spread out such that their area of contact widens to maintain a constant total volume. As a result, the brush draws a thicker line on the paper.
- When the position or direction of the brush changes, the brush does not reach its equilibrium shape for some time interval \( \Delta t \) because of the bristles’ elasticity.
- If during a brush stroke the ink is not uniformly distributed in the bristles, the shade will vary along and across the stroke.
- When using absorbent paper, the effect of diffusion is apparent around the boundary of a brush stroke.
- If the bristles are not saturated with ink, the brush stroke will show “scratchiness” and the outline of the stroke will not be distinct.

**Modeling the elastic bristles**

Real bristles possess elastic properties such that application of an external force will deform them, though they nearly resume their initial form after removal of the force. This property allows modeling the bristles using Hooke’s law. Considering bristle elasticity is an important aspect in creating a visually and physically realistic model because it

Accordingly, we simulate bristles as long, thin, elastic rods and apply the theory of elasticity to model their deformation. Note that, hereafter, all quantities in equations are vectors.

**General deflection of a single bristle**

Under Hooke’s law, an elastic body is assumed homogeneous. This means infinitesimal elements possess the same specific physical properties as the body, and forces
producing internal stresses are “near-action” forces that act from any point only to neighboring points. Accordingly, forces exerted on any part of the body by surrounding parts act only on that part’s surface.

Consider an infinitesimal element of a bristle as bounded by two adjoining cross-sections. By considering equilibrium conditions of the total force and the moment acting on it, we can write the equation for pure bending of a circular, elastic rod as

$$\frac{EI}{dl} \frac{d^2r}{dl^2} = F \times \frac{dr}{dl},$$  

(1)

where $E$ is the modulus of elasticity or Young’s modulus, $f$ the principal moment of inertia, and $dl$ and $dr$ the differentials of the central axis along the rod and radius vector, respectively.

We can obtain the equation for a bristle with one end fixed in space and the other under a force exerted at any angle $\alpha$ by using boundary conditions and applying a rotational transformation to Equation 1. Let the bending plane be the $xy$-plane such that force $f$ acts at angle $\alpha$ to the $x$-axis (see Figure 3). If $\theta$ represents the angle between a line tangent to the bristle and the $x$-axis, then we can obtain arc length $l$ from the base of the rod to the point considered by substituting $dr/dl = (\sin \theta, \cos \theta)$ and integrating Equation 1. That is,

$$l = \int_{\theta_0}^{\theta} \sqrt{\frac{EI}{2f} \int_{\theta-a}^{\theta} \frac{d\theta}{\cos(\theta_0-\alpha)\cos \theta}} d\theta$$  

(2)

The coordinates $x, y$ of a point on the bristle and its general deflection shape can be now described using the following algorithm:

**Algorithm 1 (bristle-general-deflection)**

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \left[ \begin{bmatrix} l \\ \sin \theta \ d\theta \end{bmatrix} - \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \right] \left[ F(f) \frac{1}{2} \cos \theta / G(\theta_0 - \alpha, \theta) \right] d\theta$$

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \left[ 4F(f) \left( G(\theta_0 - \alpha, \pi / 2 - \alpha) - G(\theta_0 - \alpha, \theta - \alpha) \right) \right]$$

(3)

where

$$F(f) = \sqrt{\frac{EI}{2f}}$$

and $G(\alpha, \beta) = \sqrt{\cos \alpha - \cos \beta}$

**Small deflection of a single bristle**

A solution to Algorithm 1, modeling the general shape of a bristle, exists only when sufficiently large forces are applied, such as $f \geq \pi^2 EI/4L^2$. In other words, the vector tangential to the rod ($t = dr/dl$ in Equation 1) varies significantly along its length. Accordingly, we need another algorithm for forces producing small deflections of bristles. One simple solution would neglect $dt/dl$ in Equation 1, which should be small, but then continuity is lost between Equations 2 and 3 at the critical compression force value ($\pi^2 EI/4L^2$). To obtain a solution for small deflections, we therefore employ differential interpolation, in which the following algorithm determines the bristle’s shape under a small force $f$.

**Algorithm 2 (bristle-small-deflection).** Let $P_0, P_1, \ldots, P_n$ be a sequence of points of an elastic bristle (rod) under force $f$, where each interval $|P_i - P_{(i-1)}|$ equals $\delta$; and let the “curvature angle” be defined as $\theta_i = \cos^{-1} (P_{(i+1)}, P_i)$.

1. Using Algorithm 1, for $f = c$ (the critical force value) calculate points $P_0, P_1, \ldots, P_n$ and curvature angles $\theta_0, \theta_1, \ldots, \theta_{(n-1)}$. Also, for $f = 0$, correspondingly calculate points $P_0, P_1, \ldots, P_n$ and curvature angles $\theta_0 = \theta_1 = \cdots = \theta_{(n-1)} = 0$.
2. Interpolate the curvature angles $\theta_i = (1-f/c)\theta_0 + \left(f/c\right)\theta_n$, for $i = 0, \ldots, n-1$. 

---

3 The modeled shape of a bristle undergoing bending with one end (the brush shaft) fixed and the other (the brush tip) subject to force $f$ at the surface of the paper.
3. Set \( P_0 = P_{00} = P_{0} \).
4. Set \( P_i \) for \( i = 1, \cdots, n \) so that \( \theta_{(i-1)} = \cos^{-1}(P_{ii}, P_{(i-1)}) \) and \( |P_{(i)} - P_{(i-1)}| = \delta \).

Using Algorithms 1 and 2 permits describing the shape of bristles as “large” (greater than the critical force value) or “small” (less than the critical force value) forces, respectively.

**Modeling the b-bundle**

Modeling the b-bundle first requires determining the relevant geometry and then exploring deformations of the b-bundle in response to force applied.

**Geometry of the b-bundle**

The geometry of the b-bundle relates naturally to that of the bristles (length and radius of base) as well as the distribution of bristles in the bundle (number and placement). In fact, b-bundle geometry determines computation efficiency, shading effects of the stroke, and applicable algorithms. Here I denote the number of bristles as \( N_{bt} \) and each bristle as \( \text{bristle-0}, \cdots, \text{bristle-} (N_{bt} - 1) \). Implicitly, we assume that \( \text{bristle-0} \) is the bristle located at the center of the b-bundle’s base—the “central bristle.”

As a simple means of modeling the distribution of bristles, we could assume random or uniform distribution of the bristles within the circular base and maintain the shape of the whole b-bundle. Such a brush shape—termed here a “full brush”—is simple to model, but requires exceptionally high computational time because of the large number of bristles.

An alternative model involves disregarding individual bristles situated between the central bristle and the surface of the b-bundle cone—we consider only the central bristle and those bristles at the surface of the cone. I refer to such a brush as a “shell brush.” We still consider the central bristle because we use it to compute b-bundle deformation, the central line of strokes, and other parameters. This approach has merit in that the painted trajectories of the inner bristles are always completely contained within the painted trajectories of the outside bristles. Hence, we need to consider only the region covered by the outer bristles to obtain the boundary of the brush stroke at any instant in time.

Denote the radius of the b-bundle base as \( R_{base} \) and its height as \( H_{bb} \). Accordingly, the bristles (noncentral) can be located relative to the base using the following parameters:

- Extremity of b-bundle = \( E_{bb} = (0, -H_{bb}, 0) \)
- Root of bristle = \( b_i(O_i) = (R_{base} \cos(2\pi/(N_{bt} - 1)), 0, R_{base} \sin(2\pi/(N_{bt} - 1))) \)
- Tip of bristle = \( b_i(T_i) = E_{bb}, \) for \( i = 1, \cdots, N_{bt} - 1 \)

**Deformation of the b-bundle**

The force exerted on the brush while pulled across the paper will naturally deform the b-bundle’s cone shape. Unlike the case of a single bristle, however, the bristles now interact not only with the paper, but also with neighboring bristles and the fluid ink between them. Consider a side view of the b-bundle (see Figure 4): the bristles are pulled toward the central bristle due to the adhesion produced among them by the ink. From the front, bristles appear to spread out from the central bristle; as the b-bundle flattens, the bristles push each other aside in order to maintain a constant volume.

My approach first calculates the shape of the bristles using bristle-deflection Algorithms 1 and 2, after which the brush shape is adjusted using the deformation produced inside the b-bundle. That is, the deformation of any bristle-\( n \) whose root lies at \( O_n = (x_n, 0, z_n) \) is computed by the following algorithm.

**Algorithm 3 (bundle-shape).** When the central line of the b-bundle is projected on the \( xz \)-plane, assume it’s oriented in the +x-direction.

1. Determine the shape of bristle-0 and bristle-\( n \) using the bristle-deflection algorithms.
2. To adjust the side view of bristle-\( n \): if \( x_n \) is negative (meaning bristle-\( n \) belongs to the “back-group”), pull it toward bristle-0 by executing \( \text{Rot}(z, z\text{-axis}) \) so that its tip comes close to that of bristle-0; if \( x_n \) is positive (meaning bristle-\( n \) belongs to the “front-group”) and bristle-0 is in a “down”-state, push the bristle by executing \( \text{Rot}(-z, z\text{-axis}) \) so that it lies in front of bristle-0 and its tip comes close to that of bristle-0.

3. To adjust the front view of bristle-\( n \), calculate

\[
\text{spreading angle} = k \min(\arctan(-z_n/(R + x_n)), \omega)
\]

where \( \omega \) is the maximum spreading angle chosen. Spread the bristle out by executing \( \text{Rot} \) (spreading angle, \( y \)-axis).

Figure 4 illustrates the elastic behavior of the bristles on the paper’s surface as the user presses the brush down slightly. Note that the bristles bend smoothly in the direction opposite to the stroke line, spreading out from the center of the stroke line in response to mutually applied pressure.

**Strokes**

The brush path and rendering of the brush stroke both affect the appearance and position of the brush-stroked ink on the paper. We’ll consider each in turn.

**Brush path**

The brush path is defined interactively by the user or as a list of control points comprised of parameter conditions (time, position, pressure). When control points specify the path, they’re interpolated using a Catmull-Rom spline curve\(^9\) to yield a series of nodes sufficiently close to each other that they appear as a smooth curve. I selected this type of spline curve because it interpolates the given points such that the tangent vector at a control point \( P_t \) parallels the neighboring line points \( P_{t-1} \) and \( P_{t+1} \). Once the brush path is chosen, the brush moves along it such that the brush’s orientation always remains tangent to the path. Due to friction with the paper and inertia of the bristles, however, the orientation of the b-bundle relative to the paper reluctantly changes as the orientation of the brush changes on the brush path. In other words, when the brush changes directions, the b-bundle orientation changes over some time interval \( t \) such that the unit vector \( O \) representing the orientation of the b-bundle at a point on the brush path \( P(t) \) can be described by

\[
O(t) = O(t - 1) + (1 - r) \int_{t-1}^{t} (-P''(t)) \, dt
\]

where \( r \) is the reluctance rate. To reduce computation time, approximate Equation 5 by

\[
O(t) = O(t - 1) + (1 - r) (P(t - 1) - P(t))
\]

**Rendering strokes**

In Strassmann’s novel brush model, a 1D array of bristles represents the brush, each with its own fixed position relative to the brush handle. The stroke is simply the “footprint” of all the bristles, which I classify as “simple-trajectory rendering.” This method proves advantageous in that the brush’s structure is analogous to that of a real brush and realistically creates “stroke scratchiness.” By scratchiness I mean the white breakup texture that appears when the amount of ink on some bristles drains to zero or when some bristles fail to contact the paper during a speedy brush stroke.

On the other hand, it’s difficult for users to assign the amount of ink to each and every bristle, and the resul-
tant shading will not be smooth without a sufficient number of bristles. Moreover, limitations arise with regard to controlling global shading variations within the stroke boundary or including various patterns at a desirable position within the boundary.

An alternative method I call “boundary-shading rendering” calculates the boundary line for each stroke segment and subsequently smooths the area within the boundary. Theoretically, we could apply a variety of polygon-shading methods to obtain this type of shading. Moreover, because we can determine the density of boundary line control points from the ink density of the corresponding bristles, the shading may not be independent of the brush’s ink state, as in conventional brush strokes created by CAD systems.

Given the merits of each type of rendering, my method lets users choose either simple-trajectory or boundary-shading rendering. With the former, rendering of strokes is straightforward in that the paper pixels in contact with each bristle are simply painted according to the ink quantity remaining in the bristle. With the latter method, the boundary of the stroke segment and the amount of ink within are both calculated iteratively. The “brush mark” at any instant in time is defined as the area of the paper to be painted by the bristles, represented by the convex hull of the paper points in contact with bristles. The outline of a brush mark is the “stroke segment,” later used to compute the “stroke.” The “stroke” or “boundary of the stroke” is the trail left by the brush as it moves along a path, with the resultant image computed as the connection or union of all stroke segments at every node along the brush path. Simultaneously with computing the boundary of a stroke, shading within the stroke is applied in a left-to-right linear shading pattern, across each shading segment. See Figure 5.

**Application to black-ink painting**

Applying my soft-brush method to black-ink painting requires modeling both the special paper used and the effect of the diffusion of ink. We’ll consider each in turn.

**Modeling mesh-structured fibrous paper**

The paper used in black-ink painting differs from that typically used in watercolor painting. It’s much thinner and more textured, has little sheen, and is quite absorbent, so painted fluid flows easily along its fibers. This occurs because the paper is actually a mesh of fibers, and the spaces between them act as thin capillary tubes to transport water away from the initial position of contact with the paper.

Guo and Kunii\(^\text{10}\) first proposed a model for representing a fiber mesh. They used microscopic observations to confirm that the structure of the fiber mesh consisted of randomly distributed fibers. To make the local property of fiber distribution random and the global property uniform, they divided a large field into small regions and distributed the fibers according to a rule: for each region the average fiber distribution is the same, but within each region the fiber distribution varies randomly.

Their model considers fibers randomly distributed but doesn’t take the texture of the paper into account. Consequently, it has a drawback in that the stroke’s shape and fluid diffusion that reflect characteristics of the paper material cannot be simulated. Therefore, I extended the model to simulate different paper textures (see Figure 6). This improvement lets users select the paper that best suits their creativity.

**Ink diffusion**

Diffusion of the painted fluid is an important feature in black-ink painting and, in fact, is one of the most admired features of this unique art form. A type of halo appearing around the original stroke, which adds a mysterious touch, marks the occurrence of diffusion. To simulate this effect, I constructed the following ink diffusion algorithm based on paper having a fiber mesh structure.

**Algorithm 4 (ink-diffusion).** To simulate ink diffusion in paper,

1. Find the points on the original stroke boundary...
2. For each point of the source queue, determine its “next” points, where the next points of a source point are neighboring points connected by one or more capillary tubes and having an intensity value smaller than that of the source point.

3. Insert all the next points into the “next queue.”

4. Decide the ink density at each point of the next queue according to the original intensity value of the point, the value of its source point, and a step counter (amount of remaining ink).

5. If the next queue is empty or all the points in it have no remaining ink, terminate the process; otherwise, let the next queue be the source queue and repeat the procedure from Step 2.

Discussion

Figure 7 shows two simulated black-ink paintings, one drawn with a combination of simple-trajectory rendering and boundary-shading rendering, and the other with only boundary-shading rendering varying the inside shadings. Both were generated using my soft-brush method with the number of bristles reduced to 21 to simulate fast, real-time drawing. Note the different shading effects caused by changing the brush’s rendering and shading mode.

Because the system swiftly transfers from Algorithm 1 (for large force) to Algorithm 2 (for small force) across the critical force value, as the brush is gradually lifted,
the stroke converges to a point with continuity. Figure 8 shows simulated black-ink paintings of an orchid plant drawn on paper with different textures. The effect on ink diffusion caused by the paper’s absorbent characteristics is easily seen.

The soft-brush method has several advantages over conventional techniques\(^\text{1,2,10,11}\) in that

- A more realistic drawing process results from directly manipulating the brush and by implementing dynamic movement of bristles.
- The complexity in drawing is substantially reduced, so users can focus more on drawing rather than editing control points. For example, the orchid plant shown in Figure 8, consisting of 14 leaves and branches, requires only 69 brush guiding points; to draw the same object using the conventional Bézier brush stroke technique takes about 200 spline curves.\(^2\)
- The user can select a wide variety of paper textures to obtain the desired image.

I implemented the model on a conventional Silicon Graphics O2 workstation using OpenGL. Drawing occurs either interactively or automatically using a predefined file. That is, in interactive drawing the user can specify sample stroke points using a mouse or stylus pen, with strokes drawn using information about the sequence of points and the shading pattern. If the stroke isn’t suitable, the user can change the memorized point sequence or shading pattern and create a new stroke.

Speed trades off against smoothness in interactive drawing: limiting the number of bristles makes for speedy visual feedback of the brush, but sparse bristles cause a jagged-looking stroke boundary. In contrast, densely structured bristles increase the time interval needed to complete rendering, making it difficult to maintain a smooth brush stroke. My system achieves realistic and timely visual feedback for 20 to 30 bristles, while with 50 or more bristles the system responds slower than actual brush movement.

Up to now, my research has focused mainly on modeling a soft brush drawing on absorbent paper, with results indicating the effectiveness and potential usefulness of my approach. Subtle shading effects caused by interaction of the bristles and ink offer subjects for further investigation. Presently, only an upright (vertical) brush works with the system, leaving for future study its adaptation for use with stylus pens that sense slanted angles. Nevertheless, I consider the algorithms employed here flexible enough for application to “slanted” brushes.

Acknowledgments

I thank my colleagues at the University of Aizu for their valuable support and encouragement, especially Karol Myszkowski for locating so many key references.

References


Jintae Lee is an associate professor at the University of Aizu. His research interests include virtual reality, visual communication, multimedia, and computer-aided translation. He received a BS in computer science and statistics from Seoul National University in 1981, an MS in computer science from Korea Advanced Institute of Science and Technology in 1983, and a PhD in information science from the University of Tokyo in 1993. He is a member of the IEEE Computer Society, Information Processing Society of Japan, and Korea Information Science Society.

Readers may contact Lee at the School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu City, Fukushima 965-8580, Japan, e-mail j-lee@u-aizu.ac.jp.