Real-Time Painting with an Expressive Virtual Chinese Brush

Nelson S.H. Chu and Chiew-Lan Tai
Hong Kong University of Science and Technology

A 3D virtual brush's bristles bend and spread realistically like a real Eastern paint brush. The tool lets artists create digital brushwork with spontaneity and spirit.

Many painters today are going digital, because of the convenience and ease of experimentation that painting software gives. Existing software such as Corel Painter already effectively simulates the effects of various art media, including chalk, pencil, and oil. Add some after effects, such as embossing and lighting, and an untrained eye would find it difficult to distinguish between paintings created digitally and those created conventionally. Nevertheless, tools for creating digital Eastern brushwork are still lacking, as some artists and researchers realize,[1-4] primarily because existing brush models can't simulate the delicate real-time brush deformations needed for this traditional art.

Our goal is to effectively simulate the process of brush painting—particularly the dynamics of a brush tuft—so that artists can use a virtual brush to paint with spirit—that is, in a lively, dynamic, brisk way (see the "Eastern Brushwork" sidebar). We've designed an efficient deformable brush model with spreading bristles that works in real time with consumer-level hardware. (The preliminary results of this work are available elsewhere.[5]) Our system renders realistic brushwork in response to input data captured from a device with six degrees of freedom (DOFs). Our research thus far focuses on brush modeling and ink depositing from brush to paper. The simulation of ink diffusion on paper, another important feature of Eastern brushwork, is a future goal. Ultimately, we expect a complete painting system to lead to new graphics tools for

- creating digital brushwork with spontaneity and personal style,
- rendering high-resolution Eastern fonts with the aesthetic quality of real calligraphy, and
- rendering 3D objects in an Eastern painting style.

Why 3D?

It's possible to draw any shape using a 2D mark-making method by either dragging curve control points or painting with variable dab size. Moreover, simulating brush dynamics using a 3D brush model requires more computational power. So why should we use a 3D brush model rather than a 2D one?

In real-life brushwork, a stroke's formation largely depends on the constantly changing brush footprints generated from the artist's manipulation of the brush. A 3D brush model can generate realistic brush footprints automatically from the brush posture. Without such a

Eastern Brushwork

Eastern brushwork uses the brush to deliver harmonious rhythm with varying brush posture, speed, and pressure. The artist renders each brush stroke in a continuous, rhythmic movement so that the depicted subjects show vitality and spirit.[1] The artist always seeks spiritual depiction rather than the outward appearance of the painted subjects. Often, a few deft strokes suffice for this purpose, and each brush stroke can be appreciated individually. This contrasts sharply with the traditional Western painting style developed in the Renaissance, which emphasizes realism. In that style, artists carefully dab colors onto the canvas to give a realistic look. Modern Western art places less emphasis on realism, often employing more stylish brush strokes. However, this art still doesn't emphasize subtle shape formation as much as Eastern art does.

The tools and materials for Eastern brushwork, along with the philosophies behind the art, spread from China to neighboring countries, such as Korea and Japan. In this article, we use the term *Chinese brush*, and we use Chinese characters for the calligraphy demonstration. However, our brush model is also applicable to creating other Eastern artwork of the same origin.

Reference

1. D.W. Kwo, *Chinese Brushwork: Its History, Aesthetics, and Techniques*, George Prior, London, 1981.

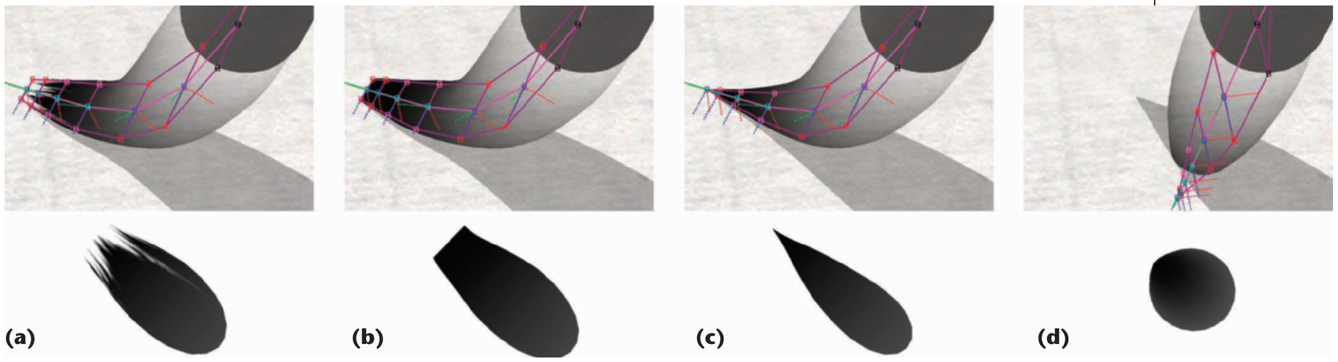
Related Work

Earlier efforts in brushwork simulation focused on stroke rendering. Strassmann swept a 1D texture to achieve varying shades in a stroke.[1] These strokes look artificial because of inadequate modeling of the natural spreading of brush bristles. Hsu and Lee proposed the skeletal stroke technique, which deforms some predefined 2D strokes to produce remarkable results.[2] However, for Chinese brushwork, this technique requires storing a large sample of stroke textures to avoid appearing repetitive. The strokes also appear unrealistic whenever there's a self-intersection or high curvature.

Later research efforts incorporated physical behaviors or physics theories into the brush models. Wong and Ip modeled a calligraphy brush as an inverted cone.[3] The bulk of the cone must penetrate the paper while stroking, and its footprint is the cone's intersection with the paper plane. Because this model ignores the brush tip while generating the strokes, it fails to produce the *biased-tip strokes* (whereby the tip travels along one side of a stroke rather than staying in the middle) common in Eastern painting and calligraphy. Based on the theory of elasticity, Lee modeled a brush as a collection of rods.[4] This model suffers from unnatural bending because it assumes homogeneous elasticity. Saito and Nakajima used a Bezier spine curve and a set of disks centered along the curve to model the brush.[5] However, this model doesn't consider brush flattening and spreading and thus fails to generate a realistic footprint. Baxter et al. modeled Western brushes as simple spring-mass systems, emphasizing the recreation of the painting process.[6] But they paid no attention to bristle spreading or splitting. Compared to Wong and Ip's cone model, the first model of Xu et al. has a more complex geometry and can split into smaller tufts,[7] but the same problem of bulk penetration still leads to unrealistic brush footprints. In their second design, Xu et al. attempted to improve the brush dynamics by querying a motion database prepared from real brush motion data,[8] but they didn't describe the nontrivial data acquisition.

References

1. S. Strassmann, "Hairy Brushes," *Proc. 13th Siggraph*, ACM Press, 1986, pp. 225-232.
2. S.C. Hsu and I.H.H. Lee, "Drawing and Animation Using Skeletal Strokes," *Proc. 21st Siggraph*, ACM Press, 1994, pp. 109-118.
3. H.T.F. Wong and H.H.S. Ip, "Virtual Brush: A Model-Based Synthesis of Chinese Calligraphy," *Computers and Graphics*, vol. 24, no. 1, Feb. 2000, pp. 99-113.
4. J. Lee, "Simulating Oriental Black-Ink Painting," *IEEE Computer Graphics and Applications*, vol. 19, no. 3, May/June 1999, pp. 74-81.
5. S. Saito and M. Nakajima, "3D Physics-Based Brush Model for Painting," *Proc. 26th Siggraph*, ACM Press, 1999, p. 226.
6. B. Baxter et al., "DAB: Interactive Haptic Painting with 3D Virtual Brushes," *Proc. 28th Siggraph*, ACM Press, 2001, pp. 461-468.
7. S. Xu et al., "A Solid Model Based Virtual Hairy Brush," *Computer Graphics Forum*, vol. 21, no. 3, Sept. 2002, pp. 299-308.
8. S. Xu et al., "Advanced Design for a Realistic Virtual Brush," *Computer Graphics Forum*, vol. 22, no. 3, Sept. 2003, pp. 533-542.

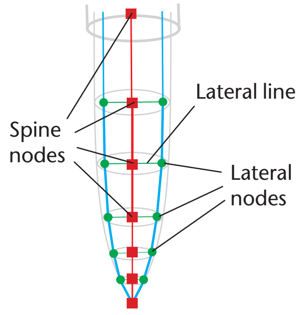


1 Comparison of tuft deformation effects and the corresponding footprints of our brush model (a) in full gear, (b) without split map, (c) without split map and lateral spreading, and (d) without brush–paper collision handling.

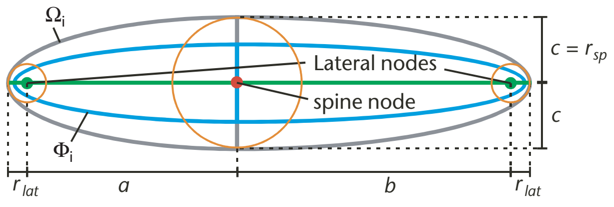
model, it's difficult to produce a stroke that looks like it was painted in a bouncing rhythm with an elastic brush. Rhythmic vitality, the essence of Eastern art, is lost in the process of adjusting control points or parameters. This is analogous to the need for a motion capture to provide lively character motions in computer-generated animation. When executing spontaneous strokes with a real brush, the brush's elasticity completes the stroke vibrancy. Thus, the elastic brush's response to the artist's motion plays an important part in producing the final appearance. This is essentially why strokes made with Chinese brushes can be so eloquent. Modeling the brush's physics lets us extend real elastic brushes' extensive power to the digital domain. Moreover, a 3D physical model allows visual feedback of the brush shape. Coupled with an appropriate input device, such a brush model makes a painting system intuitive to artists. In view of these benefits, we designed a physically based 3D brush model. (The "Related Work" sidebar discusses other brush models and related techniques.)

Brush modeling

Our challenge is to develop a model that can collectively simulate the bristles in real time yet is flexible enough to produce the effects that artists expect. Brush deformation, which causes the ever-changing footprints, plays a key role in producing convincing brushwork. Figure 1 compares the different brush deformations and the



2 Geometric model of a brush tuft.



3 Tuft cross section.

corresponding footprints produced from our prototype system when we turned off various modeling features. Figure 1a shows a fully functioning tuft pressed against the virtual paper. The bristles split and spread laterally to produce a wider footprint. Figure 1b is the result of turning off the texture-based bristle-splitting effect. Further removing lateral spreading gives the result in Figure 1c, which is similar to the results of previous physical models.[2,6] If the model doesn't handle brush–paper collision, the brush simply penetrates the paper (Figure 1d), and the footprint is a cross section of the brush geometry, which resembles the effect of other previous models.[3,4]

Our brush model includes four components: brush geometry, brush dynamics, ink loading, and ink depositing. We now describe how we model the brush as a single tuft. This single-tuft model also serves as a building block for a hierarchical representation of split brushes.

Brush geometry

The brush geometry closely determines the brush dynamics. We represent the geometry in two layers: the skeleton and the surface.

Brush skeleton

The skeleton includes a spine and some lateral nodes, as Figure 2 shows. The spine handles the general bending of the entire tuft; the lateral nodes model the tuft's lateral deformation and spreading. We represent the spine using a connected sequence of line segments that become progressively shorter toward the tip. We gave the tip higher resolution because it's softer, and usually an artist uses only the tip and the belly to paint. Each joint between two segments has two DOFs in a spherical coordinate system's latitude and longitude—called the bend angle and the turn angle at that joint.

We denote the position of spine node N_i as O_i, $i = 0, \ldots, n$, where O_0 is the brush root node's position. Two lateral nodes are attached to each spine node; these nodes can move only along the spine node's lateral line. The lateral line of N_i passes through O_i and is initially perpendicular to the two adjacent spine segments. It has one rotational DOF on the spine node's joint-bisecting plane (the plane passing through O_i and bisecting the angle between the two adjacent spine segments).

The two lateral nodes attached to a spine node represent two groups of bristles on both sides of the spine. This design can effectively capture the essence of tuft deformation. Because the brush interacts with only a planar painting surface, tuft flattening, controlled by the bending and lateral drag, largely determines the brush footprint. With the lateral lines having one DOF, the nonpenetration constraints in our dynamics model tend to keep the lateral lines of the spine nodes that touch the paper parallel to the paper. Consequently, when the artist presses the brush against the paper, the lateral nodes' loci lie on the painting surface, thus effectively modeling horizontal deformation and the lateral spreading of the bristles.

Brush surface

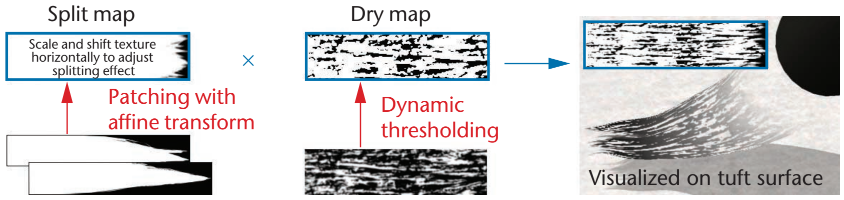
We represent the brush surface as a swept surface defined by the spine and a varying elliptic cross section. When the brushes are moistened and unbent, the cross sections are circles along the entire spine. We predefine these initial tuft radii for various types of brushes. In general, the cross section comprises two half ellipses, with possibly different major radii but a common minor radius, as Figure 3 shows. This simple representation is computationally efficient and doesn't differ much from real brush tufts. Cross section Ω_i at spine node N_i lies on the joint-bisecting plane, and its major axis coincides with the lateral line of N_i. Let a and b be the distances between N_i and its lateral nodes. The major radii for W_i are $(a + r_{\text{lat}})$ and $(b + r_{\text{lat}})$, where r_{lat} is the effective radii for the two lateral nodes. We then use the conservation of area to compute the common minor radius, c, as $c = r^2/[1/2(a + b) + r_{\text{lat}}]$, where r is the initial tuft radius at N_i.

Fine bristle effect

To obtain footprints with the bristle-level brush-splitting effect while using a single tuft, we define an alpha map, named the split map, to make part of the tuft surface transparent. We can either prepare the split map from a scanned image of real bristles or have the system automatically generate it by patching some mask primitives on the fly, as Figure 4 shows. A static split map applied on a single tuft, along with brush flattening, is sufficient to simulate convincing bristle splitting as long as the brush is not pressed too hard (see Figure 1a). To simulate the effect of wider spreading, we propose multiple tufts that work in a hierarchy, with split maps still applied for a fine-bristle effect. Split maps increase the modeling efficiency dramatically. They are an inexpensive feature supported by graphics hardware, and they produce ink streaks that look very natural.

Brush dynamics

Modeling the brush as a spring-mass system and solving its motion equations using a vectorial method is one possible way to simulate brush dynamics. However, to


Split map
Scale and shift texture horizontally to adjust splitting effect
Patching with affine transform
Dry map
Dynamic thresholding
Visualized on tuft surface

4 Texture-based ink-depositing effects.

avoid unrealistically bouncy or slack motion, the spring coefficients must be fairly large relative to the node masses. This makes it difficult to produce a tractable real-time system using this approach: The simulation time step must be very small or the stiff spring force will cause instability. Baxter et al. cope with instability by employing first-order dynamics and an approximated implicit integrator.[6] However, the approximated integrator has the drawback of modeling the brush's internal forces using only stretch springs rather than bend springs. Bend springs let us model bristles with nonstretchable constraints far more effectively.

We handle brush dynamics using a variational approach,[7] which lets us avoid dealing with stiff differential equations. Brush dynamics involves both conservative and dissipative forces. We apply the *minimum principle of incremental potential energy*, described by Pandolfi et al.,[8] to predict brush motion. At each time step (equal to the output refresh time), the system updates the brush state by choosing the incremental deformation of the brush skeleton from among those satisfying all external constraints. This update minimizes the sum of strain energy and incremental energy dissipation.

Energy minimization problems

We formulate the brush dynamics as a series of static constrained minimization problems. The objective functions are the dynamic system's current incremental potential energy, and the constraint is that the brush geometry must be above the paper. The set of variables Ψ in the minimization comprises all the joint angles and stretches of the lateral nodes. Our incremental potential energy has three components: strain energy, internal frictional energy, and external frictional energy. We don't include a component accounting for inertia because real brushes reach equilibrium almost instantaneously as an artist presses them against or lifts them from the paper. The strain energy accounts for the potential energy stored in the brush due to deformation from its unstrained state. The internal frictional energy refers to the work done against the internal friction between water and bristle molecules during tuft deformation. The external friction energy refers to the work done against the frictional force between the brush and the paper surface. We derive the brush's strain energy using a spring system. Imposing bend and twist springs on the bend angles and the turn angles at the spine nodes serves to model the tuft's general deformation. Stretch springs between the spine nodes and their lateral nodes model lateral deformation. Finally, bend springs between consecutive lateral nodes account for the stiffness of the bristles on the brush's sides. The "Energy Functions" sidebar (next page) presents the formulations of these spring energies as well as the frictional energy. Such behavior functions suffice for producing plausible brush movements.

We introduce inequality constraints in the minimization to account for the paper's normal reaction force on the brush. If the paper plane is at $y = 0$, we can express the constraints as $g_i(\Psi) = y_i > 0$, where y_i is the lowest point of an ellipsoid generated by rotating Ω_i about its major axis. However, to generate the brush footprint, we let part of the brush surface penetrate the paper; this penetration also makes the brush appear to have a flat bottom when pressed against the paper. Hence, for collision detection, we use a shrunked version of Ω_i, denoted by Φ_i (see Figure 3).

We solve energy minimization problems using local sequential quadratic programming (SQP) due to its fast convergence. There are a few points we need to take care of when applying gradient-based optimization techniques such as SQP. While a user drags a brush on paper, minimizing the external frictional energy means minimizing the number of nodes that are in contact with the paper, thus causing the brush to be lifted. To avoid this unphysical lifting, we don't allow the change of the touching state of the brush nodes during the search for the minimum; that is, throughout each time step, we use the touching state evaluated on the first iteration.

Minimization algorithms have difficulty evaluating gradients at the point where the initial joint angles are all zero (that is, a straight skeleton). We adopt a simple solution of assigning some minimal values to a few of the joint angles whenever the system or the users straighten the brush.

In general, we should also try to avoid discontinuity in the energy functions when formulating them so that the minimization can converge more easily.

Brush plasticity

When a wet brush is pressed and deformed, energy is lost due to the friction between the bristles and the water within them. When the artist releases the brush, the restoring spring force must overcome this friction's resistance to revert the brush to its original shape. Failing to do so makes the brush appear plastic.

We use a simple method to model this plasticity. The idea is to make the brush have a high tendency to remain in the skeleton configuration of the last time frame. To achieve this, we simply shift the spring energy functions so that the lowest energy is at the position corresponding to the last skeleton configuration. To avoid having the brush appear overly deformed, we also clamp the shifted distance against a user-adjustable plasticity parameter, α. Suppose the original bending energy function is $E(\theta) = k|\theta|^m$. If θ' is the bend-angle from the previous time frame, the new bending energy is then $E(\theta) = k|\theta - \rho|^m$, where $\rho = \text{clamp}(\theta', -\alpha, \alpha)$. This simple but effective method significantly improves the brush's realism, producing the plastic behavior that users expect

Energy Functions

Here, we describe the formulations of energy functions we use to derive the brush's incremental potential energy.

Strain energy

The energy stored in a *bend* or *twist* spring has the form,

$$E(\Psi) = \kappa_i |\theta_i|^m$$

where θ_i is the bend angle (or turn angle, for a twist spring) at the spine node, κ is the spring coefficient, and m is another parameter that controls the spring's strength. Our current implementation uses $m = 2$ or 3. In general, we can determine spring coefficients empirically, assuming the brush root is stiffer than the tip. We omit a twist spring at the root so that the tuft has no preference for bending in any particular direction.

We take the distance of a lateral node from its spine node as $r_i + \delta_i^2$, where r_i is the sum of the spine node's initial tuft radius and the lateral node's effective radius, and δ_i is the variable in the minimization that models the stretch degree of freedom. The energy function for a *stretch* spring attached to a lateral node has the form,

$$E(\Psi) = \kappa_i |\delta_i^2 - S(\theta_i, p_i)|^m$$

where p_i is the proportion of the associated tuft segment under the paper, and S is a heuristic function of θ_i and p_i that increases the spring's rest length as the artist presses the brush against the paper.

Internal frictional energy

We derive this energy by applying the same form of energy functions used for the bend energy to the joint angle and lateral displacement changes since the last frame.

External frictional energy

For every node contacting the paper, we calculate its frictional energy as

$$E(\Psi) = \mu R_i x_i$$

where x_i is the dragged distance, R_i is the normal reaction force, and μ is the friction coefficient. We consider a node as contacting the paper if its height minus its effective radius is below the paper. We denote a spine node's effective radius as r_{sp}, and its lateral nodes' effective radius as r_{lat}. We set r_{sp} equal to minor radius c of cross section Ω_i from the previous time frame, and we take r_{lat} as a fraction of r_{sp}.

Real wet bristles slightly adhere to the paper because of moisture. In our simulation, we also want the brush tip to experience some friction even when its normal force is zero. So, for each wet node, we also add to its R_i a small value proportional to its p_i.

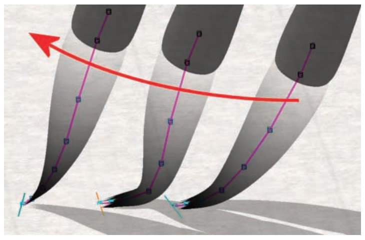
Because the hair on a brush is generally aligned, it tends to experience a larger resistance when dragged sideways. To control this anisotropic behavior, we modulate μ with the spine segment's direction.

from real brushes. Brush plasticity affects the rhythmic movement made by artists and thus indirectly determines the ink traces. In real-life Chinese painting, reshaping the brush tip to a sharp point is often necessary before drawing a new stroke. Users can set the plasticity to 0 so that tip reshaping is eliminated altogether. Our implementation enables reverting the brush to its original shape at any time through a user command.

Pore resistance

Most types of paper for painting are full of pores. When the tip of a slanted brush touches the paper, the pores act like a fence impeding sliding. If the artist pushes the brush toward the direction pointed to by the tip, these pores continue to exert considerable resistance, as Figure 5 shows, producing the rough texture seen in pushed strokes employed in many paintings.[9] To simulate such brush behavior and a painting effect, we also model paper pore resistance. We add an extra constraint in the minimization problem in the form of a blocking plane in front of the tip. This constraint is active only when the tip touches the paper. The blocking plane is vertical and normal to the projected spine segment of the brush tip onto the paper surface. To adjust the blocking effect, we increase the lead distance, L, between the plane and the brush tip, as $L = at + bu + c$. In this equation, a, b, and c are user-adjustable parameters; $t = \max(|\beta| - \gamma, 0)$, where β is the angle that the spine tip segment makes with the paper plane's normal vector, γ is the critical tilt angle within which the pores trap the tip; and u is the pressure experienced by the tip.

A soft brush might be too plastic to spring back to form a tip needed for making a bladelike stroke ending. Calligraphers cope with this by using pore resistance to straighten the brush. Mimicking pore resistance in our simulation lets us reproduce such deformation, which experienced artists expect.


5 Pore resistance deforming the brush. The red arrow indicates brush motion. The cyan line indicates the blocking constraint; this line turns orange when the constraint is active.

Ink loading

In real-life Chinese color painting, artists usually mix ink, water, and colors within the brush by loading different colors up to different lengths of the brush and then gently stroking to blend the colors. Allowing color loading and blending using similar brush motions in a virtual brush system would be the most intuitive. However, our current system employs a loading method that allows more precise control. Users define a color gradient by assigning colors ramps along the gradient, which the system then maps to the brush surface axially. Such an interface is common in existing illustration software. We intend to implement the intuitive color-loading interface as an option, along with the simulation of ink diffusion within the brush.

Ink depositing

Ink depositing involves determining which parts of the paper are receiving ink and how ink transfers from the brush to the paper. The first step is to determine the current brush footprint. Similar to Baxter et al.,[6] we let part of the brush surface intersect with the paper plane, and we consider the orthogonal projection of the penetrating portion onto the paper plane as the footprint. After obtaining the footprint, the system updates the ink work (stored as a texture for the paper) with the footprint's ink values. Users can either have the system automatically subtract the ink values from the brush or maintain the ink level to allow continuous painting without reloading. To apply transparent colors atop existing colors, the system uses alpha blending. The generation of the footprint and the blending of the deposited ink both occur on the graphics processing unit (GPU) using OpenGL, thus minimizing data transfer and leaving more of the CPU for the physics simulation. To produce more realistic ink traces, we model three ink-depositing effects: a dry brush, a soaked brush, and grain texture.

Dry brush

We can regard bristles as forming a height field on the brush surface. When a brush is rather dry, only the peaks deposit ink. In addition to the split map, we define another alpha map called the dry map for the tuft surface. The system dynamically generates the dry map by thresholding a predefined gray-scale image (see Figure 4), mimicking the gradual drying of the brush. We prepared this gray-scale image from real dry-brush prints to give a natural feel. The threshold varies across the entire map, reflecting the moisture level and the pressure at the brush nodes. Using the multitexturing function of OpenGL, the final alpha texture applied onto the tuft surface is the split map modulated by the dry map.

Soaked brush

The tip of a soaked brush expands a little because of excess moisture, so ink streaks don't appear even if the tip splits slightly. To mimic this effect, we simply expand the tip geometry slightly and dilate the opaque regions on the split map. As the excess moisture drains, the tip geometry shrinks back, and the opaque regions on the split map contract back to their original shapes. Our implementation effectively achieves dilation and contraction by scaling and shifting the split map down the length of the spine without dynamic texture generation.

Grain texture

Paper grain texture emerges only when the brush is rather dry. We use a gray-scale image to represent the paper height field. The system dynamically thresholds the height field according to the brush's current wetness and pressure to produce an alpha mask to apply on the resulting brush footprint.

Chinese Brushes

With a development history of 3,000 years, Chinese brushes are designed to make expressive lines.[1] Deft manipulation is necessary to effectively use them. A typical brush has layers of different animal hairs to provide balance between absorbency and springiness (see Figure A). Some artists regard the brush's tip as the part that gives it its spirit—that is, its character, or ability to produce artwork with vitality. The artist often makes sure the tip is sharp before approaching the paper. As the ink on the brush is gradually consumed, the brush breaks into smaller tufts. The artist can also split the brush tip deliberately to draw multiple lines with a single stroke.

Reference

1. S.K. Ng, *Brushstrokes: Styles and Techniques of Chinese Painting*, Asian Art Museum of San Francisco, 1992.

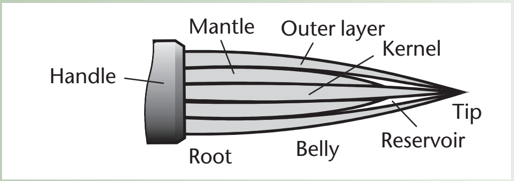
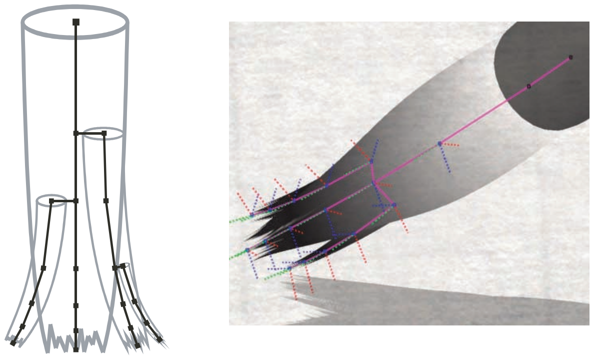


Figure A. Anatomy of a typical brush.

Simulation with multiple tufts

A thoroughly moistened brush forms a single tuft. The interaction of the brush with the paper surface and the reduced moisture, however, causes the brush tuft to break into smaller tufts during painting. When splitting begins, certain types of brushes tend to have some unruly bristles at the outer layer sprouted outward, forming thin tufts acting like satellites (see the "Chinese Brushes" sidebar). Other types of brushes with even bristle distribution tend to split into tufts of even sizes. All brushes finally become bushy when they are really dry. When a brush splits, it draws multiple lines simultaneously, and the small tufts can go through slightly different paths, producing subtle ink streaks.

To better simulate the splitting of real brushes, we introduce multiple tufts in a hierarchy so that a geometric split is feasible, as Figure 6 (next page) shows. Child tuft T_j is associated with a parent node, which is spine node N_{ij} of its parent tuft. The root of child tuft T_j lies on the joint-bisecting plane of N_{ij} and is confined within elliptic cross section Ω_{ij}. Its exact location is denoted by lateral displacement $\mathbf{r}_j = (r_{xj}, r_{yj})$ from O_{ij}, defined along and normalized by the lengths of the major and minor radii of Ω_{ij}. Several factors cause sprouting to occur: displacement of lateral nodes, pore resistance, and bending or drying of the brush. Applying simple heuristics on the parent tuft's friction, bend angles, and wetness, the system can decide when to



6 **Multiple tufts in a hierarchy: (a) schematic; (b) a brush with two child tufts sprouted.**

sprout child tufts and identify their parent nodes. The lateral displacements derive from the factors that cause the sprouting:

- displacement of a lateral node ($\mathbf{r}_j$ lies in the vicinity of that lateral node),
- pore resistance ($\mathbf{r}_j$ lies in the vicinity of the minor axis of Ω_{ij}), and
- bending or drying ($\mathbf{r}_j$ is randomized within Ω_{ij}).

For efficiency, we let the parent and its child tufts intersect. We don't enforce conservation of total tuft volume, because a split brush can become bushy. To model thin child tufts sprouted from the outer hair layer, we let the child tufts have default flattened shapes. To model brushes that split into evenly sized tufts, we replace the main tuft by several smaller tufts without any parent–child relationship.

A new tuft has the same spine structure as its parent, from the tip to its parent node, and the system initially assigns it the same bend and turn angles. Upon creation of a new tuft, its lateral displacement is fixed, but the tuft can split further upwards—the parent node can become N_{ij-1}. The child tuft then grows by one segment at its root. A simple reset reverts a split brush to a single tuft.

Currently, we solve for a child tuft's skeletal configuration using a separate energy minimization problem after finding its parent's configuration. We can reduce the computational demand of simulating many child tufts by eliminating lateral nodes of thin tufts. These thin tufts can still exhibit flattening, which only its spine's bend angles control. A few tufts (at most five) suffice for most painting purposes; more are necessary only when applying special painting effects such as rubbing with a bushy brush to produce rough texture. In such cases, it might not be worthwhile to simulate dozens of tufts with accurate physics, because the essence of such strokes lies in the controlled randomness rather than in the brush elasticity.

Real-time simulation

We solve energy minimization using SQP. There are several ways to speed up the simulation. SQP converges quickly if the initial estimate is close to the solution. In this application, the brush state from the last time frame serves as a good initial estimate. The first time an unbent brush touches the paper, the system gets a good initial estimate of the turn angle at the brush root, ϕ_0, by initializing ϕ_0 such that the lateral lines are horizontal. If the user holds the brush exactly vertical when it touches the paper (which is rare), the system simply sets ϕ_0 to 0, because the angle doesn't matter in this case.

The speedup largely depends on the correlation of successive minimization problems in our simulation. The SQP method has many variations; our current energy minimizer is based on a common implementation that iteratively updates a Hessian matrix, $\boldsymbol{L}_{xx}$, from which we derive the solution's search direction. We modified the basic algorithm to use the resulting $\boldsymbol{L}_{xx}$ from the previous time step, rather than the identity matrix, as the initial $\boldsymbol{L}_{xx}$. Exploiting the similarity of these matrices reduces the number of iterations needed by up to 50 percent. This speedup over our previous implementation,[5] along with improved code efficiency, has made multiple tufts practical in our current prototype.

We limit the number of iterations q in each search process to a maximum value, $q_{\max}$, to maintain a reasonable frame rate. In our current prototype, we set $q_{\max}$ to about 20 iterations for each tuft of six segments. If the prescribed accuracy is not reached when q exceeds $q_{\max}$ in the current frame, the search process effectively continues in the next time step because the Hessian $\boldsymbol{L}_{xx}$ is passed on. Table 1 compares the performance of our painting system with various dynamics simulation settings.

It's important not to waste computation on excessive accuracy. Unlike scientific applications, our brush simulation doesn't need to be very accurate for the brush to work well. Single-word precision is sufficient, and the

Table 1. Overall frame rates for various dynamics simulation settings based on experiments performed on a 2.26-GHz Pentium-4 PC with a Nvidia GeForce 4 graphics card, where q is the number of iterations in a search process.

No. of six-segment tufts	$q_{\max}$	Without continuous Hessian updates: Average q	Without continuous Hessian updates: Frames/s	With continuous Hessian updates: Average q	With continuous Hessian updates: Frames/s
1	20	14.6	60.5	7.4	62.6
3	20	14.6	30.4	7.4	44.6
3	25	15.4	28.8	8.2	43.3
3	30	16.7	28.0	9.7	42.6
5	20	14.6	18.9	7.4	28.5

SQP adopts empirically determined threshold values (for example, convergence criteria).

System prototype

We've built a prototype painting system based on our brush model (see Figure 7). The current version uses programmable shader units for ink depositing and uses floating-point textures to store the ink information (needed for ink diffusion, not discussed in this article). Therefore, this version requires an NV30-level graphics card. An earlier version that doesn't use the above hardware features can run on an NV20-level card. In both cases, the overall frame rate can be as high as 60 frames per second (see Table 1).

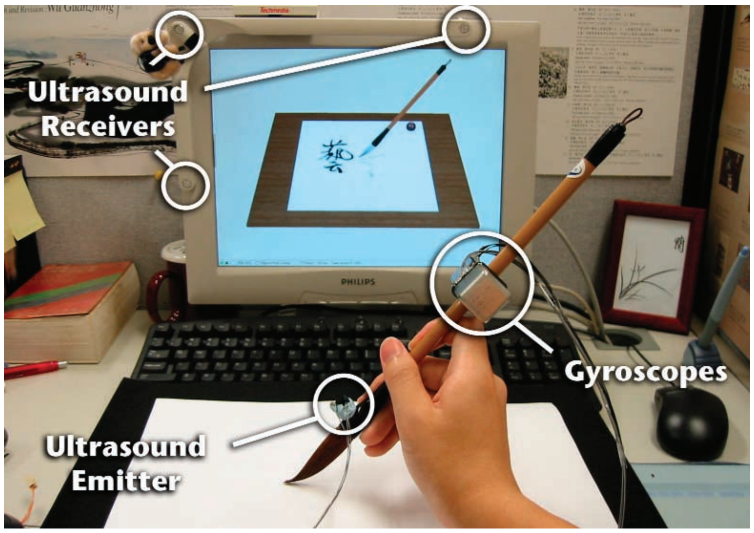
To drive the virtual brush, we built a 6-DOF input device by combining affordable sensor components. We use an ultrasonic device and miniature gyroscopes to detect the brush position and orientation, respectively. These sensors are attached to a real brush or a brush-like object, which the user manipulates to drive the virtual brush in real time. Users can calibrate the input device to map a real supporting surface to the virtual one, so that the real surface gives them some tangible feeling when they press the brush on the paper. However, one drawback of this approach is that current gyroscopes are still too heavy to allow a burden-free manipulation.

As an alternative, our system also supports pressure- and tilt-sensitive graphics tablets. The sensed pressure controls the brush's height above the paper, whereas the tilt controls its orientation. The best tablet model commonly available today is a 5-DOF device that doesn't track the rotation about the stylus' own axis. Although this approach doesn't control the brush height as directly as a true 3D positional device and doesn't sense brush twisting, the wide availability of tablet products makes our system more accessible to today's digital artists. The graphics tablet is also more convenient to use because a supporting ground already exists without calibration.

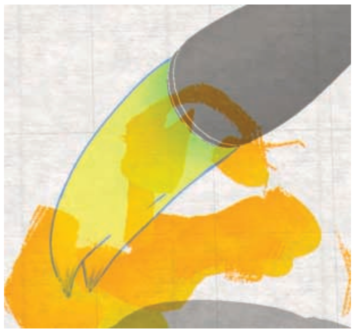
We've also experimented with an affordable 6-DOF glove-like controller that uses optical tracking technology. One drawback of this option is its bulkiness, which prevents us from delicately controlling the virtual brush with our fingers. Other alternatives include a robot-arm-based haptic device, which Baxter et al. discuss.[6] In our opinion, a suitable and affordable 6-DOF input device needs to be developed before common artists can enjoy a true-to-life digital painting experience. (The cheap ones aren't good enough, and the good ones are way too expensive.)

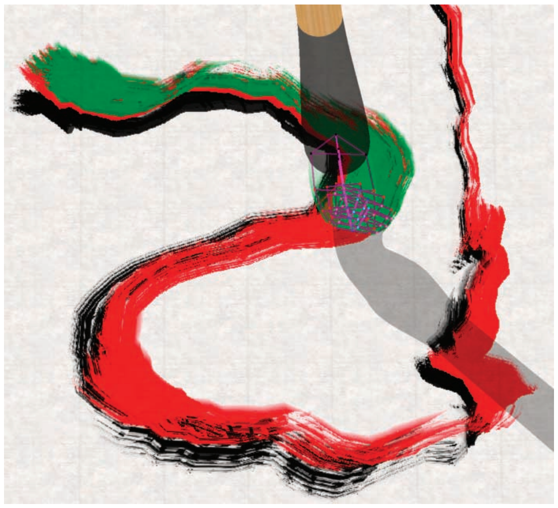
Visual feedback of the brush shape is important during the painting process. In our system, a user can choose either an orthogonal view or some other perspective of the 3D painting scene. Rendering the brush shadow lets the user judge the virtual brush's position. A common way to suggest the tuft's shape is to shade it. However, some users might prefer to see the ink color loaded on the tuft unmodulated. In this case, the system can outline the tuft for legibility, as Figure 8 shows. The contrasting outline also avoids camouflaging the tuft when it's held over a ground of the same color as the ink loading, which is often the case. Finally, users can render the brush as transparent to avoid blocking their view of the painting itself.

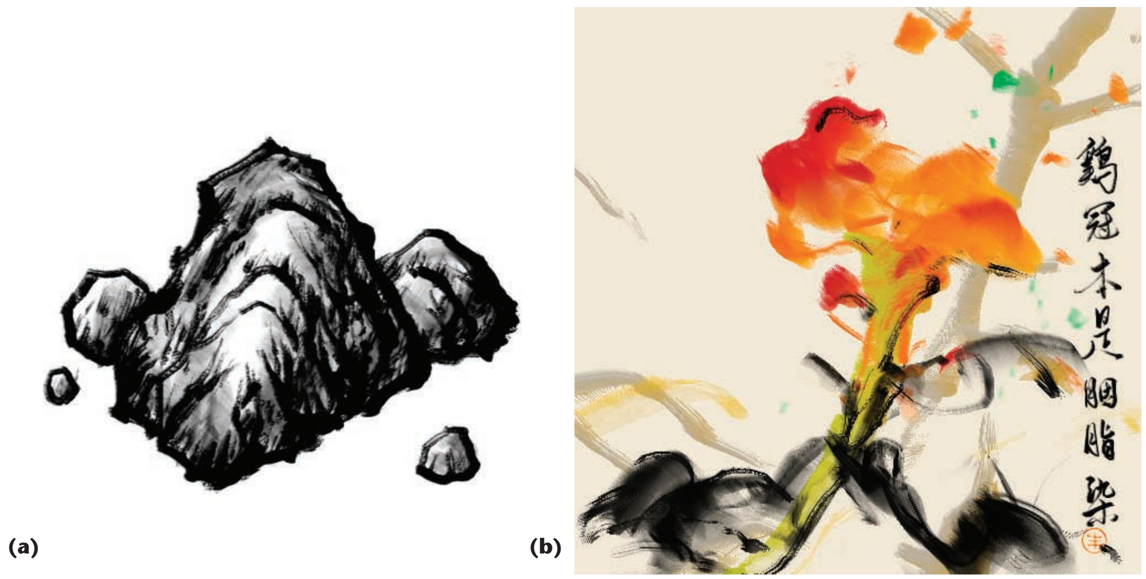
The system also allows brush movement to be recorded for later playback. Figure 9 shows a snapshot of playing back some recorded movement. This playback illustrates that we can produce slightly different strokes from the same brush movement by having a different initial brush skeletal state.



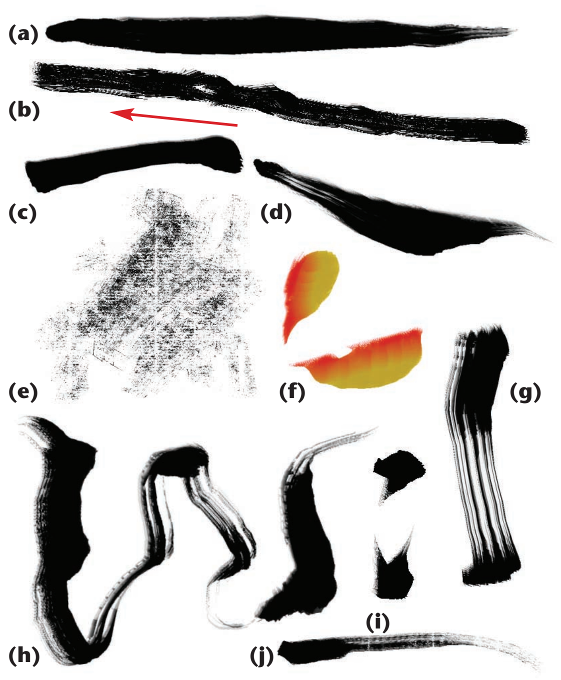
7 Using sensors to capture brush motion.


8 Tuft outlined by the system for legibility.

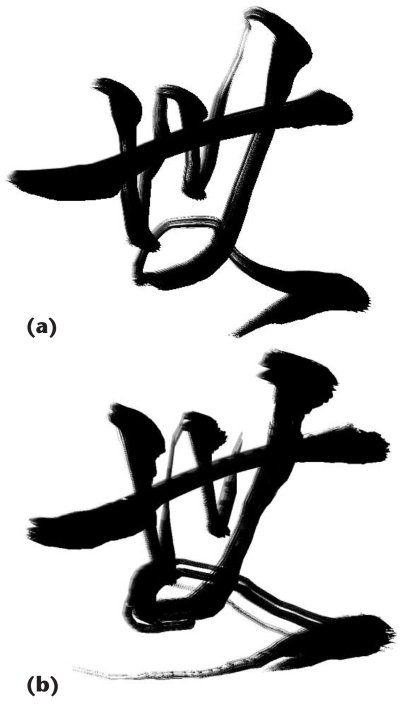

9 Playing back strokes created with multiple tufts. Original strokes are in black; played-back strokes are in red and green.



10 Sample artwork from our prototype system: (a) rocks painted with a single-tuft brush; (b) flower painted with color gradients using a single-tuft brush.



11 Sample strokes: (a) central tip; (b) pushed (the arrow indicates the direction of the stroke); (c) biased tip; (d) blade-like calligraphic; (e) rubbing with a dry brush; (f) petals made with gradient loaded; (h) made with a flattened brush; (i) calligraphic dots; (j) made with a drying brush.



12 Sample calligraphy done with (a) a single tuft, and (b) multiple tufts.

Our prototype also implements other features, such as stroke undo, image saving, and image loading. We have yet to incorporate ink diffusion on absorbent paper; thus, all simulated paintings act as if the paper is sized (treated with alum).

Sample results

Figures 10, 11, and 12 show some sample artwork, strokes, and calligraphy from our prototype system. In Figure 11b, a geometric brush split caused the rough edges. A quickly drying brush tip (belly still wet) made the stroke in Figure 11d, creating white streaks near the beginning of the stroke, which are not easy to get with a real brush. In Figure 11f, our brush dynamics naturally simulated the breakup at the top of the lower petal. We used a flattened singe-tuft brush to make the stroke in Figure 11g, which is similar to those used for painting bamboo trunks. (Video demos and additional color images are available at http://www.cs.ust.hk/~cpegnel/VCB/.)

Conclusion

Our brush model certainly can't mimic every way real brushes respond to artists' manipulations. However, one possible way to improve the modeling accuracy is to devise more subtle behavior functions. A good behavior function should include parameters that the user can adjust intuitively to derive brush variants. We could also improve modeling efficiency by using arcs rather than line segments for the spine; such a smooth spine needs fewer deformation parameters to attain comparable accuracy, thus reducing the optimization problem's size.

We are trying to couple ink diffusion simulation with our brush model.[10] To further improve speedup, we plan to investigate matrix-updating techniques. We will also try to better balance the CPU and GPU workloads for higher performance.

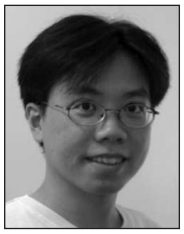
Although we've focused on modeling Eastern brushes, it should be easy to extend our brush model to represent other kinds of brushes as well. Since the introduction of the first natural-media painting system, digital artists have been mixing different simulated media to explore new effects. As computer technology advances, more real-time effects will become feasible. We believe that using computers can lead the way to new heights in the continual development of this ancient Eastern art form, as well as other art traditions. ■

Acknowledgments

We thank Kwan-Wah Ng for his generous help in building the input hardware, and Rui-Duo Yang for his help in hardware interfacing. The rapid software prototyping wouldn't be possible without various free software libraries or components, and we are grateful to their authors—particularly Eric Grange, Mike Lischke, Alex Denisov, Mattias Andersson, and Dejan Crnila. We also thank Xavier Granier and the anonymous reviewers for their constructive comments. Finally, Sino Software Research Institute grants SSRI00/01.EG08 and SSRI01/02.EG16 supported this work.

References

1. J. Lee, "Simulating Oriental Black-Ink Painting," *IEEE Computer Graphics and Applications*, vol. 19, no. 3, May/June 1999, pp. 74-81.
2. S. Saito and M. Nakajima, "3D Physics-Based Brush Model for Painting," *Proc. 26th Siggraph*, ACM Press, 1999, p. 226.
3. H.T.F. Wong and H.H.S. Ip, "Virtual Brush: A Model-Based Synthesis of Chinese Calligraphy," *Computers and Graphics*, vol. 24, no. 1, Feb. 2000, pp. 99-113.
4. S. Xu et al., "A Solid Model Based Virtual Hairy Brush," *Computer Graphics Forum*, vol. 21, no. 3, Sept. 2002, pp. 299-308.
5. N.S.-H. Chu and C.-L. Tai, "An Efficient Brush Model for Physically-Based 3D Painting," *Proc. 10th Pacific Conf. Computer Graphics* (PG 02), IEEE CS Press, 2002, pp. 413-421.
6. B. Baxter et al., "DAB: Interactive Haptic Painting with 3D Virtual Brushes," *Proc. 28th Siggraph*, ACM Press, 2001, pp. 461-468.
7. B. Tabarrok and F.P.J. Rimrott, *Variational Methods and Complementary Formulations in Dynamics*, Kluwer Academic, 1994.
8. A. Pandolfi et al., "Time-Discretized Variational Formulation of Non-Smooth Frictional Contact," *Int'l J. Numerical Methods in Eng.*, vol. 53, 2002, pp. 1801-1829.
9. D.W. Kwo, *Chinese Brushwork: Its History, Aesthetics, and Techniques*, George Prior, London, 1981.
10. T.L. Kunii, G.V. Nosovskij, and T. Hayashi, "A Diffusion Model for Computer Animation of Diffuse Ink Painting," *Computer Animation* (CA 95), IEEE CS Press, 1995, pp. 98-102.



Nelson S.H. Chu *is a research assistant in the Computer Vision and Graphics Group at the Hong Kong University of Science and Technology. His research interests include nonphotorealistic rendering, image-based rendering, and the development of new tools for digital artists. Chu has a BEng in computer engineering and an MPhil in computer science, both from the Hong Kong University of Science and Technology.*

Chiew-Lan Tai *is an assistant professor of computer science at the Hong Kong University of Science and Technology. Her research interests include geometric modeling and nonphotorealistic rendering. Tai has a BSc and an MSc in mathematics from the University of Malaya, an MSc in computer and information sciences from the National University of Singapore, and a DSc in information science from the University of Tokyo. She is a member of the ACM.*

Readers may contact Nelson Chu or Chiew-Lan Tai at Computer Vision and Graphics Group, Dept. of Computer Science, Hong Kong Univ. of Science and Technology, Clear Water Bay, Hong Kong; cpegnel@ust.hk or taicl@ust.hk.